Department of Electrical & Computer Engineering

Second Semester (2022/2023)

ENCS3130 Linux Laboratory

Shell Scripting Project – Statistics of Running Processes on Linux Machine

---

Prepared by:

Katya Kobari 1201478

Hidaya Mustafa 1201910

Instructor: Dr. Mohammad Jubran

Teaching Assistant: Eng. Ibrahim Injas

Date:13/6/2023

Section: 3

Introduction:

This project is a simulation of a menu-driven script designed to perform statistical calculations on a "top" output file. It offers users the ability to analyze CPU usage and network packet data, enabling them to identify resource-intensive commands and monitor the overall performance of a system.

Contents

Table of figures:

Procedure:

1. The program should print on the screen the main menu and ask the user to select an option

For this required we write a function to print the menu and call it in the while loop.

```
print_menu() {
  echo "Select an option to run the top statistics project :"
  echo "r - Read top output file"
  echo "c - Calculate average, minimum,and maximum CPU usage"
  echo "i - Calculate average, minimum,and maximum  of packets received"
  echo "o - Calculate average, minimum,and maximum  of packets sent"
  echo "u - Find commands with the maximum CPU usage"
  echo "a - Find commands with the maximum average memory usage"
  echo "b - Find commands with the minimum average memory usage"
  echo "e - Exit"
}

while true ; do
print_menu
read -p "Enter your selection: " selection
case "$selection" in

```

Figure 1. First required Code.

```
katya@katya-VirtualBox: ~
katya@katya-VirtualBox:~$ ./input.sh
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 2. First required Output.

2. If the user enters ‘r’:

Here we ask user to enter the name of input file and check wither is existing or not using –e.

```
case "$selection" in

  r)
  #Read top output file
    read -p ""Please enter the name of the file: " name
    if [ -e "$name" ]
      then
      echo "File exists."
    else
      echo "File does not exist."
    fi
    ;;
```

Figure 3. selection r Code.

```
Enter your selection: r
“Please enter the name of the file: in.txt
File exists.
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 4. Selection r Output

3. If the user enters ‘c’

Here we read the input file and searches for lines that indicate CPU usage. It extracts the CPU usage values, calculates the sum, and keeps track of the count, minimum, and maximum values. Finally, it outputs the average, minimum, and maximum CPU usage if data is found, or notifies that no CPU usage data was found in the file.

```
c)
  # Calculate the average, minimum, and maximum CPU usage
 if [ -z "$name" ]; then
   echo "please must do case (r) to read file "
 else
 cpu_count=0
 sum=0
 min=9999999
 max=0

while IFS= read -r line; do
 if grep -qE "^CPU usage: " <<< "$line"; then
     cpu_usage=$(echo "$line" | awk -F "CPU usage: " '{print $2}' | awk '{printf "%.3f", $1}')
     if [[ "$cpu_usage" =~ ^[0-9]+\.[0-9]+$ ]]; then
       sum=$(awk "BEGIN {print $sum + $cpu_usage}")
       cpu_count=$((cpu_count + 1))

       if (( $(echo "$cpu_usage < $min" | bc -l) )); then
         min=$cpu_usage
       fi

       if (( $(echo "$cpu_usage > $max" | bc -l) )); then
         max=$cpu_usage
       fi
     fi
   fi
done < "$name"

 if [ "$cpu_count" -gt 0 ]; then
   avg=$(awk "BEGIN {print $sum / $cpu_count}")
   echo "Average CPU usage: $avg"
   echo "Minimum CPU usage: $min"
   echo "Maximum CPU usage: $max"
 else
   echo "No CPU usage data found in the file"
 fi
fi
;;
```

Figure 5. Selection c Code

```
e - Exit
Enter your selection: c
Average CPU usage: 2.8238$
Minimum CPU usage: 1.980
Maximum CPU usage: 3.930
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 6. Selection c Output

4. If the user enters 'i'

Here we calculate the average, minimum, and maximum number of packets received from input file. If the file is not provided, it prompts the user to read the file first. Else will reads the contents of the file and searches for lines that indicate packet information. It extracts the number of received packets, calculates the sum, and keeps track of the count, minimum, and maximum values. Finally, it outputs the average, minimum, and maximum packets received if data is found, or notifies that no packet data was found in the file.

```
i)
# Calculate the average, minimum, and maximum number of packets received
    packets_sum=0
    packets_count=0
    packets_minimum=9999999
    packets_maximum=0
    if [ -z "$name" ]; then
      echo "please must do case (r) to read file "
    else
    while IFS= read -r line; do
        if grep -qE "^Networks: packets: " <<< "$line"; then

          received_packets=$(echo "$line" | awk -F 'Networks: packets: ' '{print $2}' | awk -F '/' '{print $1}')

          if [[ "$received_packets" =~ ^[0-9]+$ ]]; then
            packets_sum=$((packets_sum + received_packets))
            packets_count=$((packets_count + 1))

            if ((received_packets < packets_minimum)); then
              packets_minimum=$received_packets
            fi

            if ((received_packets > packets_maximum)); then
              packets_maximum=$received_packets
            fi
          fi
        fi
      done < "$name"

      if [ $packets_count -gt 0 ]; then
        packets_average=$((packets_sum / packets_count))

        echo "Average packets received: $packets_average"
        echo "Minimum packets received: $packets_minimum"
        echo "Maximum packets received: $packets_maximum"
      else
        echo "No packet data found in the file"
      fi

    fi
;;
```

Figure 7. Selection i Code

```
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection: i
Average packets received: 3760726
Minimum packets received: 3760418
Maximum packets received: 3760827
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 8. Selection i Output.

5. If the user enters 'o':

Here we calculate the average, minimum, and maximum number of packets sent. It reads the contents of input file and searches for lines that contain packet information. then extracts the number of sent packets, calculates the sum, and keeps track of the count, minimum, and maximum values. Finally, it outputs the average, minimum, and maximum packets sent if data is found, or notifies that no packet data was found in the file.

```
o)
    packets_sum=0
packets_count=0
packets_minimum=9999999
packets_maximum=0


if [ -z "$name" ]; then
echo "please must do case (r) to read file "
else
  while IFS= read -r line; do
    if [[ $line =~ packets:[[:space:]]+[0-9]+/[0-9]+M[[:space:]]+in,[[:space:]]+([0-9]+)/[0-9]+M[[:space:]]+out ]]; then
      sent_packets=${BASH_REMATCH[1]}
      if [[ "$sent_packets" =~ ^[0-9]+$ ]]; then
        packets_sum=$((packets_sum + sent_packets))
        packets_count=$((packets_count + 1))
        if ((sent_packets < packets_minimum)); then
          packets_minimum=$sent_packets
        fi
        if ((sent_packets > packets_maximum)); then
          packets_maximum=$sent_packets
        fi
      fi
    fi
  done < "$name"

  if [ $packets_count -gt 0 ]; then
    packets_average=$((packets_sum / packets_count))
    echo "Average packets sent: $packets_average"
    echo "Minimum packets sent: $packets_minimum"
    echo "Maximum packets sent: $packets_maximum"
  else
    echo "No packet data found in the file"
  fi
fi
;;
```

Figure 9. Selection o Code.

```
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection: o
Average packets sent: 1967681
Minimum packets sent: 1967584
Maximum packets sent: 1967736
```

Figure 10. Selection o Output.

6. If the user enters 'u'

Here we read a file and prompts the user to enter an integer number. It then finds commands with the maximum CPU usage from the file. It calculates the average CPU usage for each command and prints the m commands with the highest maximum average CPU usage. Additionally, it displays the average CPU usage for each command found in the file.

```
u)
# Find commands with the maximun CPU usage
  if [ -z "$name" ]; then # check if the user entered a file name
 echo "File not found. Please make sure to read the file by using the case (r)."
 echo ""
      else
read -p "Enter an integer number: " m
while ! [[ $m =~ ^[0-9]+$ ]]; do # check if the m is an integer
    echo "Error, Entered value is not an integer. Please try again."
    read -p "Enter an integer number: " m
done
#cut data from file
start="COMMAND"
end="Processes:"
output=$(awk "/$start/,/$end/" "$name")
out=$(echo "$output" | grep -v "$end"| grep -v "$start")
cpu_usage=$(echo "$out" | sed 's/^[^ ]* //')
declare -A cpu_sum
declare -A cpu_counts

while read -r line; do
        s=0
        e=0

        while [ $s -lt ${#line} ]; do
            current_char="${line:s:1}"

            if [[ $current_char =~ [0-9] ]]; then
                e=$s
                break
            fi

            ((s++))
        done

        ((e--))
        process="${line:0:e}"

        g=$((e+1))
        i=0

        while [ $g -lt ${#line} ]; do
            current_char="${line:g:1}"
               if [[ $current_char == " " ]]; then
```

Figure 11. Selection u Code p1

```
            current_char=${line:g:1}
            if [[ $current_char == " " ]]; then
                i=$g
                break
            fi

            ((g++))
        done

        cpu=`${line:e:t-e}`

        # Sum the total CPU usage for each process and insert it into the array
        if [[ -n "${cpu_sum[$process]}" ]]; then
            cpu_sum[$process]=$(echo "${cpu_sum[$process]} + $cpu" | bc -l)
        else
            cpu_sum[$process]=$cpu
        fi

        # Update the count for each process
        if [[ -n "${cpu_counts[$process]}" ]]; then
            cpu_counts[$process]=$((cpu_counts[$process] + 1))
        else
            cpu_counts[$process]=1
        fi
    done <<< "$cpu_usage"

declare -A avg_cpu_totals

# Calculate average CPU usage for each command
for process in "${!cpu_sum[@]}"; do
    cpu_sum="${cpu_sum["$process"]}"
    count="${cpu_counts["$process"]}"
    avg_cpu=0

    if (( count > 0 )); then
        avg_cpu=$(awk "BEGIN {printf \"%.1f\", $cpu_sum / $count}")
    fi
    avg_cpu_totals["$process"]=$avg_cpu
done
#sort command according avg cpu
sorted_entries1=()
while IFS== read -r key1 value1; do
  sorted_entries1+=("$key1=$value1")
done < <(for key1 in "${!avg_cpu_totals[@]}"; do echo "$key1=${avg_cpu_totals[$key1]}"; done | sort -t= -k2nr)
echo ""
```

Figure 12. Selection u Code p2

```
echo ""
echo "Top $m commands with the maximum average cpu usage:"
echo ""
for ((i=0; i<m && i<${#sorted_entries1[@]}; i++)); do
  entry=${sorted_entries1[i]}
  key="${entry%%=*}"
  value="${entry#*=}"
   echo "Command: $key, AVG CPU usage: ($value)"
done
echo ""
echo "*********"
echo ""
fi
;;
```

Figure 13. Selection u Code p3

```
Enter your selection: u
Enter an integer number: 3

Top 3 commands with the maximum average cpu usage:

Command: Google Chrome    , AVG CPU usage: (10.35)
Command: Finder           , AVG CPU usage: (10.33)
Command: WindowServer     , AVG CPU usage: (10.28)

*********

             ******
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 14. Selection u Output

7. If the user enters 'a'

Here we read the input file and prompts the user to enter an integer number. Then finds commands with the highest average memory (MEM) usage from the file. and calculates the average memory usage for each command and prints the m commands with the highest average memory usage. Additionally, it displays the average memory usage for each command found in the file. If the file is not provided, it prompts the user to read the file first.

```
a)
# Find commands with the maximum mem usage
 if [ -z "$name" ]; then # check if the user entered a file name
echo "File not found. Please make sure to read the file by using the case (r)."
echo ""
    else
  read -p "Enter an integer number: " m

  while ! [[ $m =~ ^[0-9]+$ ]]; do
      echo "Error: Entered value is not an integer. Please try again."
      read -p "Enter an integer number: " m
  done
# cut data from input file
  start="COMMAND"
  end="Processes:"
  output=$(awk "/$start/,/$end/" "$name")
  out=$(echo "$output" | grep -v "$end" | grep -v "$start")
  mem_usage=$(echo "$out" | sed 's/^[^ ]* //')

  declare -A mem_sum
  declare -A mem_counts

  while read -r line; do
      s=0
      e=0

      while [ $s -lt ${#line} ]; do
          current_char="${line:s:1}"
          if [[ $current_char =~ [0-9] ]]; then
              e=$s
              break
          fi
          ((s++))
      done
      ((e--))
      process="${line:0:e}"
      process=$(echo "$process" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
      mem_i=$(awk -v FS="[ ]+" '{if (NF > 0) print $7}' <<< "${line:e}")
      mem_i=$(echo "$mem_i" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
      a=0
      q=0
      while [ $a -lt ${#mem_i} ]; do
          current="${mem_i:a:1}"
         if [[ $current == " " || $current =~ [[:alpha:]] ]]; then
```

Figure 15. Selection a Code p1.

```
310            if [[ $current == " " || $current == [[:alpha:]] ]]; then
311                q=$a
312                break
313            fi
314            ((a++))
315        done
316 # convert K-1024 , M-1024*1024
317        mem="${mem_l:0:q}"
318        if [[ $current == [K] ]]; then
319            mem=$((mem*1024))
320        elif [[ $current == [M] ]]; then
321            mem=$((mem*1024*1024))
322        fi
323        if [[ -n "${mem_sun[$process]}" ]]; then
324            mem_sun[$process]=$(echo "${mem_sun[$process]} + $mem" | bc -l)
325        else
326            mem_sun[$process]=$mem
327        fi
328
329        # Update the count for each process
330        if [[ -n "${mem_counts[$process]}" ]]; then
331            mem_counts[$process]=$((mem_counts[$process] + 1))
332        else
333            mem_counts[$process]=1
334        fi
335    done <<< "$mem_usage"
336
337    declare -A avg_mem_totals
338
339    # Calculate average mem usage for each command
340    for process in "${!mem_sun[@]}"; do
341        sum=${mem_sun["$process"]}
342        count=${mem_counts["$process"]}
343        avg_mem=0
344        if (( count > 0 )); then
345            avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
346        fi
347            avg_mem_totals["$process"]=$avg_mem
348    done
349 #sort command according avg
350 sorted_entries=()
351 while IFS== read -r key value; do
352   sorted_entries+=("$key=$value")
353 done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -kZnr)
```

Figure 16. Selection a Code p2.


```
310            if [[ $current == " " || $current == [[:alpha:]] ]]; then
311                q=$a
312                break
313            fi
314            ((a++))
315        done
316 # convert K-1024 , M-1024*1024
317        mem="${mem_l:0:q}"
318        if [[ $current == [K] ]]; then
319            mem=$((mem*1024))
320        elif [[ $current == [M] ]]; then
321            mem=$((mem*1024*1024))
322        fi
323        if [[ -n "${mem_sun[$process]}" ]]; then
324            mem_sun[$process]=$(echo "${mem_sun[$process]} + $mem" | bc -l)
325        else
326            mem_sun[$process]=$mem
327        fi
328
329        # Update the count for each process
330        if [[ -n "${mem_counts[$process]}" ]]; then
331            mem_counts[$process]=$((mem_counts[$process] + 1))
332        else
333            mem_counts[$process]=1
334        fi
335    done <<< "$mem_usage"
336
337    declare -A avg_mem_totals
338
339    # Calculate average mem usage for each command
340    for process in "${!mem_sun[@]}"; do
341        sum=${mem_sun["$process"]}
342        count=${mem_counts["$process"]}
343        avg_mem=0
344        if (( count > 0 )); then
345            avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
346        fi
347            avg_mem_totals["$process"]=$avg_mem
348    done
349 #sort command according avg
350 sorted_entries=()
351 while IFS== read -r key value; do
352   sorted_entries+=("$key=$value")
353 done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -kZnr)
```


Figure 17. Selection a Code p3.

```
Enter your selection: a
Enter an integer number: 3
*********
Top 3 commands with the maximum average memory usage:

Command: Google Drive    , AVG MEM usage: (2856321024.00)
Command: WindowServer    , AVG MEM usage: (1795791257.60)
Command: Microsoft Outloo, AVG MEM usage: (488636416.00)
*********

              ******
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection:
```

Figure 18. Selection a Output.

8. If the user enters 'b'

Here we find commands with the maximum memory usage from a specified file. It calculates the average memory usage for each command and displays the top commands with the minimum average memory usage based on a user-provided integer value (m).

```
b)
 # Find commands with the maximum mem usage
 if [ -z "$name" ]; then # check if the user entered a file name
echo "File not found. Please make sure to read the file by using the case (r)."
echo ""
    else
  read -p "Enter an integer number: " m

  while ! [[ $m =~ ^[0-9]+$ ]]; do
      echo "Error: Entered value is not an integer. Please try again."
      read -p "Enter an integer number: " m
  done

  start="COMMAND"
  end="Processes:"

  output=$(awk "/$start/,/$end/" "$name")
  out=$(echo "$output" | grep -v "$end" | grep -v "$start")
  mem_usage=$(echo "$out" | sed 's/^[^ ]* //')

  declare -A mem_sum
  declare -A mem_counts

  while read -r line; do
      s=0
      e=0

      while [ $s -lt ${#line} ]; do
          current_char="${line:s:1}"
          if [[ $current_char =~ [0-9] ]]; then
              e=$s
              break
          fi
          ((s++))
      done
      ((e--))
      process="${line:0:e}"
      process=$(echo "$process" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
      mem_t=$(awk -v FS='[ ]+' '{if (NF > 6) print $7}' <<< "${line:e}")
      mem_t=$(echo "$mem_t" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
      a=0
      q=0
      while [ $a -lt ${#mem_t} ]; do
          current="${mem_t:a:1}"
         if [[ $current == " " || $current =~ [[:alpha:]] ]]; then
```

Figure 19. Selectin b code p1

```
            if [[ $current == " " || $current =~ [[:alpha:]] ]]; then
                q=$a
                break
            fi
            ((a++))
        done
        mem="${mem_l:0:q}"
        if [[ $current == [K] ]]; then
            mem=$((mem*1024))
        elif [[ $current == [M] ]]; then
            mem=$((mem*1024*1024))
        fi

        if [[ -n "${mem_sum[$process]}" ]]; then
            mem_sum[$process]=$(echo "${mem_sum[$process]} + $mem" | bc -l)
        else
            mem_sum[$process]=$mem
        fi

        # Update the count for each process
        if [[ -n "${mem_counts[$process]}" ]]; then
            mem_counts[$process]=$((mem_counts[$process] + 1))
        else
            mem_counts[$process]=1
        fi
    done <<< "$mem_usage"

    declare -A avg_mem_totals

    # Calculate average mem usage for each command
    for process in "${!mem_sum[@]}"; do
        sum=${mem_sum["$process"]}
        count=${mem_counts["$process"]}
        avg_mem=0
        if (( count > 0 )); then
            avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
        fi
            avg_mem_totals["$process"]=$avg_mem
    done

sorted_entries=()
while IFS== read -r key value; do
  sorted_entries+=("$key=$value")
done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -k2n)
```

Figure 20. Selectin b code p2

```
done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -k2n)
echo ""
echo "*********"
echo "Top $n commands with the minimum average memory usage:"
echo ""
for ((i=0; i<n && i<${#sorted_entries[@]}; i++)); do
  entry=${sorted_entries[i]}
  key="${entry%%=*}"
  value="${entry#*=}"
   echo "Command: $key, AVG MEM usage: ($value)"
done
echo ""
    echo "*********"

fi
    ;;
```

Figure 21. Selectin b code p3

```
Enter your selection: b
Enter an integer number: 3

*********
Top 3 commands with the minimum average memory usage:

Command: com.apple.ColorS, AVG MEM usage: (626688.00)
Command: mdworker_shared, AVG MEM usage: (1352135.11)
Command: exchangesyncd  , AVG MEM usage: (1830912.00)

*********
            ******
```

Figure 22. Selection b output

9. If the user enters ‘e’

prompts the user with a confirmation message to exit the program. If the user responds with "yes", the script displays a message indicating that it is exiting and terminates. If the user enters any other response, an error message is displayed, and the program continues to prompt the user for input until a valid selection is made.

```
  e)
    read -p "Are you sure you want to exit? " ch
if [[ $ch == "yes" ]]; then
 echo " Exiting... "
  exit 0
fi
    ;;
  *)
    echo "Invalid selection ! ! , try again "
    ;;
esac
done
```

Figure 23. Selection e code

```
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum,and maximum CPU usage
i - Calculate average, minimum,and maximum  of packets received
o - Calculate average, minimum,and maximum  of packets sent
u - Find commands with the maximum CPU usage
a - Find commands with the maximum average memory usage
b - Find commands with the minimum average memory usage
e - Exit
Enter your selection: e
Are you sure you want to exit? yes
 Exiting...
katya@katya-VirtualBox:~$
```

Figure 24. Selection e output

Code:

```
print_menu() {
  echo "            ******                    "
  echo "Select an option to run the top statistics project :"
  echo "r - Read top output file"
  echo "c - Calculate average, minimum,and maximum CPU usage"
  echo "i - Calculate average, minimum,and maximum  of packets received"
  echo "o - Calculate average, minimum,and maximum  of packets sent"
  echo "u - Find commands with the maximum CPU usage"
  echo "a - Find commands with the maximum average memory usage"
  echo "b - Find commands with the minimum average memory usage"
  echo "e - Exit"
}

while true ; do
print_menu
read -p "Enter your selection: " selection
case "$selection" in

  r)
  #Read top output file
    read -p "“Please enter the name of the file: " name
    if [ -e "$name" ]
      then
      echo "File exists."
    else
      echo "File does not exist."
    fi
    ;;



  c)
```

```
# Calculate the average, minimum, and maximum CPU usage

if [ -z "$name" ]; then # check if the user entered a file name
  echo "File not found. Please make sure to read the file by using the case (r)."
  echo ""
else

  cpu_count=0
  sum=0
  min=9999999
  max=0

  # read data from the input file, compute avg, max, min
  while read -r line; do

    if grep -qE "^CPU usage: " <<< "$line"; then
      cpu_usage=$(echo "$line" | awk -F 'CPU usage: ' '{print $2}' | awk '{printf "%.3f", $1}')

      if [[ "$cpu_usage" =~ ^[0-9]+\.[0-9]+$ ]]; then
        sum=$(awk "BEGIN {print $sum + $cpu_usage}")
        cpu_count=$((cpu_count + 1))

        if (( $(echo "$cpu_usage < $min" | bc -l) )); then
          min=$cpu_usage
        fi

        if (( $(echo "$cpu_usage > $max" | bc -l) )); then
          max=$cpu_usage
        fi
      fi
    fi
```

```
  done < "$name"

  # print results
  if [ "$cpu_count" -gt 0 ]; then
    avg=$(awk "BEGIN {print $sum / $cpu_count}")
    echo ""
    echo "Average CPU usage :  $avg"
    echo "Minimum CPU usage :  $min"
    echo "Maximum CPU usage :  $max"
    echo ""
  else
    echo "No CPU usage data found in the file"
  fi
fi
;;

  i)
  # Calculate the average, minimum, and maximum number of packets received
   if [ -z "$name" ]; then # check if the user entered a file name
  echo "File not found. Please make sure to read the file by using the case (r)."
  echo ""
      else
      packets_sum=0
      packets_count=0
      packets_minimum=9999999
      packets_maximum=0
      # read data from the input file, compute avg, max, min of Networks packets received
      while IFS= read -r line; do
         if grep -qE "^Networks: packets: " <<< "$line"; then
           received_packets=$(echo "$line" | awk -F 'Networks: packets: ' '{print $2}' | awk -F '/' '{print $1}')
```

```
          if [[ "$received_packets" =~ ^[0-9]+$ ]]; then
            packets_sum=$((packets_sum + received_packets))
            packets_count=$((packets_count + 1))

            if ((received_packets < packets_minimum)); then
              packets_minimum=$received_packets
            fi

            if ((received_packets > packets_maximum)); then
              packets_maximum=$received_packets
            fi
          fi
        fi
      done < "$name"
# print results
      if [ $packets_count -gt 0 ]; then
        packets_average=$((packets_sum / packets_count))
        echo ""
        echo "Average packets received :  $packets_average"
        echo "Minimum packets received :  $packets_minimum"
        echo "Maximum packets received :  $packets_maximum"
        echo ""

      else
        echo "No packet data found in the file"
      fi
    fi
  ;;



  o)
   #Calculate the average, minimum, and maximum number of packets sent
```

```
 if [ -z "$name" ]; then # check if the user entered a file name
echo "File not found. Please make sure to read the file by using the case (r)."
echo ""
    else
    packets_sum=0
    packets_count=0
    packets_minimum=9999999
    packets_maximum=0
    # read data from the input file, compute avg, max, min of Networks packets sent
while IFS= read -r line; do
  if [[ $line =~ packets:[[:space:]]+[0-9]+/[0-9]+M[[:space:]]+in,[[:space:]]+([0-9]+)/[0-9]+M[[:space:]]+out ]];
then
    sent_packets=${BASH_REMATCH[1]}
    if [[ "$sent_packets" =~ ^[0-9]+$ ]]; then
      packets_sum=$((packets_sum + sent_packets))
      packets_count=$((packets_count + 1))
      if ((sent_packets < packets_minimum)); then
        packets_minimum=$sent_packets
      fi
      if ((sent_packets > packets_maximum)); then
        packets_maximum=$sent_packets
      fi
    fi
  fi
done < "$name"

if [ $packets_count -gt 0 ]; then
  packets_average=$((packets_sum / packets_count))
  echo ""
  echo "Average packets sent :  $packets_average"
  echo "Minimum packets sent :  $packets_minimum"
  echo "Maximum packets sent :  $packets_maximum"
```

```
    echo ""
  else
    echo "No packet data found in the file"
  fi
fi
;;

 u)
# Find commands with the maximum CPU usage
   if [ -z "$name" ]; then # check if the user entered a file name
  echo "File not found. Please make sure to read the file by using the case (r)."
  echo ""
      else
read -p "Enter an integer number: " m
while ! [[ $m =~ ^[0-9]+$ ]]; do # check if the m is an integer
    echo "Error, Entered value is not an integer. Please try again."
    read -p "Enter an integer number: " m
done
#cut data from file
start="COMMAND"
end="Processes:"
output=$(awk "/$start/,/$end/" "$name")
out=$(echo "$output" | grep -v "$end"| grep -v "$start")
cpu_usage=$(echo "$out" | sed 's/^[^ ]* //')
declare -A cpu_sum
declare -A cpu_counts

while read -r line; do
        s=0
        e=0

        while [ $s -lt ${#line} ]; do
```

```
    current_char="${line:s:1}"

    if [[ $current_char =~ [0-9] ]]; then
        e=$s
        break
    fi

    ((s++))
done

((e--))
process="${line:0:e}"

g=$((e+1))
i=0

while [ $g -lt ${#line} ]; do
    current_char="${line:g:1}"
       if [[ $current_char == " " ]]; then
        i=$g
        break
    fi

    ((g++))
done

cpu="${line:e:i-e}"

# Sum the total CPU usage for each process and insert it into the array
if [[ -n "${cpu_sum[$process]}" ]]; then
    cpu_sum[$process]=$(echo "${cpu_sum[$process]} + $cpu" | bc -l)
else
```

```
        cpu_sum[$process]=$cpu
    fi

    # Update the count for each process
    if [[ -n "${cpu_counts[$process]}" ]]; then
        cpu_counts[$process]=$((cpu_counts[$process] + 1))
    else
        cpu_counts[$process]=1
    fi
  done <<< "$cpu_usage"

declare -A avg_cpu_totals

# Calculate average CPU usage for each command
for process in "${!cpu_sum[@]}"; do
  cpu_sum=${cpu_sum["$process"]}
  count=${cpu_counts["$process"]}
  avg_cpu=0

  if (( count > 0 )); then
    avg_cpu=$(awk "BEGIN {printf \"%.2f\", $cpu_sum / $count}")
  fi
  avg_cpu_totals["$process"]=$avg_cpu
done
#sort command according avg cpu
sorted_entries1=()
while IFS== read -r key1 value1; do
 sorted_entries1+=("$key1=$value1")
done < <(for key1 in "${!avg_cpu_totals[@]}"; do echo "$key1=${avg_cpu_totals[$key1]}"; done | sort -t= -k2nr)
echo ""
echo "Top $m commands with the maximum average cpu usage:"
```

```
echo ""
for ((i=0; i<m && i<${#sorted_entries1[@]}; i++)); do
  entry=${sorted_entries1[i]}
  key="${entry%%=*}"
  value="${entry#*=}"
   echo "Command: $key, AVG CPU usage: ($value)"
done
echo ""
echo "*********"
echo ""
fi
;;

  a)
  # Find commands with the maximum mem usage
   if [ -z "$name" ]; then # check if the user entered a file name
  echo "File not found. Please make sure to read the file by using the case (r)."
  echo ""
     else
    read -p "Enter an integer number: " m

    while ! [[ $m =~ ^[0-9]+$ ]]; do
        echo "Error: Entered value is not an integer. Please try again."
        read -p "Enter an integer number: " m
    done
# cut data from input file
    start="COMMAND"
    end="Processes:"
    output=$(awk "/$start/,/$end/" "$name")
    out=$(echo "$output" | grep -v "$end" | grep -v "$start")
    mem_usage=$(echo "$out" | sed 's/^[^ ]* //')
```

```
    declare -A mem_sum
    declare -A mem_counts

    while read -r line; do
        s=0
        e=0

        while [ $s -lt ${#line} ]; do
            current_char="${line:s:1}"
            if [[ $current_char =~ [0-9] ]]; then
                e=$s
                break
            fi
            ((s++))
        done
        ((e--))
        process="${line:0:e}"
        process=$(echo "$process" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
        mem_i=$(awk -v FS="[ ]+" '{if (NF > 6) print $7}' <<< "${line:e}")
        mem_i=$(echo "$mem_i" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
        a=0
        q=0
        while [ $a -lt ${#mem_i} ]; do
            current="${mem_i:a:1}"
           if [[ $current == " " || $current =~ [[:alpha:]] ]]; then
                q=$a
                break
            fi
            ((a++))
        done
# convert K-1024 , M-1024*1024
        mem="${mem_i:0:q}"
```

```
        if [[ $current == [K] ]]; then
            mem=$((mem*1024))
        elif [[ $current == [M] ]]; then
            mem=$((mem*1024*1024))
        fi
        if [[ -n "${mem_sum[$process]}" ]]; then
            mem_sum[$process]=$(echo "${mem_sum[$process]} + $mem" | bc -l)
        else
            mem_sum[$process]=$mem
        fi

        # Update the count for each process
        if [[ -n "${mem_counts[$process]}" ]]; then
            mem_counts[$process]=$((mem_counts[$process] + 1))
        else
            mem_counts[$process]=1
        fi
    done <<< "$mem_usage"

    declare -A avg_mem_totals

    # Calculate average mem usage for each command
    for process in "${!mem_sum[@]}"; do
        sum=${mem_sum["$process"]}
        count=${mem_counts["$process"]}
        avg_mem=0
        if (( count > 0 )); then
            avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
        fi
        avg_mem_totals["$process"]=$avg_mem
    done
#sort command according avg
```

```
sorted_entries=()
while IFS== read -r key value; do
  sorted_entries+=("$key=$value")
done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -
k2nr)
echo "*********"
echo "Top $m commands with the maximum average memory usage:"
 echo " "
for ((i=0; i<m && i<${#sorted_entries[@]}; i++)); do
  entry=${sorted_entries[i]}
  key="${entry%%=*}"
  value="${entry#*=}"
   echo "Command: $key, AVG MEM usage: ($value)"
done
echo "*********"
    echo " "
fi
    ;;
  b)
   # Find commands with the maximum mem usage
   if [ -z "$name" ]; then # check if the user entered a file name
  echo "File not found. Please make sure to read the file by using the case (r)."
  echo ""
      else
   read -p "Enter an integer number: " m

   while ! [[ $m =~ ^[0-9]+$ ]]; do
       echo "Error: Entered value is not an integer. Please try again."
       read -p "Enter an integer number: " m
   done

   start="COMMAND"
```

```
end="Processes:"

output=$(awk "/$start/,/$end/" "$name")
out=$(echo "$output" | grep -v "$end" | grep -v "$start")
mem_usage=$(echo "$out" | sed 's/^[^ ]* //')

declare -A mem_sum
declare -A mem_counts

while read -r line; do
    s=0
    e=0

    while [ $s -lt ${#line} ]; do
        current_char="${line:s:1}"
        if [[ $current_char =~ [0-9] ]]; then
            e=$s
            break
        fi
        ((s++))
    done
    ((e--))
    process="${line:0:e}"
    process=$(echo "$process" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
    mem_i=$(awk -v FS="[ ]+" '{if (NF > 6) print $7}' <<< "${line:e}")
    mem_i=$(echo "$mem_i" | sed 's/^[[:space:]]//; s/[[:space:]]$//')
    a=0
    q=0
    while [ $a -lt ${#mem_i} ]; do
        current="${mem_i:a:1}"
       if [[ $current == " " || $current =~ [[:alpha:]] ]]; then
            q=$a
```

```
            break
        fi
        ((a++))
    done
    mem="${mem_i:0:q}"
    if [[ $current == [K] ]]; then
        mem=$((mem*1024))
    elif [[ $current == [M] ]]; then
        mem=$((mem*1024*1024))
    fi

    if [[ -n "${mem_sum[$process]}" ]]; then
        mem_sum[$process]=$(echo "${mem_sum[$process]} + $mem" | bc -l)
    else
        mem_sum[$process]=$mem
    fi

    # Update the count for each process
    if [[ -n "${mem_counts[$process]}" ]]; then
        mem_counts[$process]=$((mem_counts[$process] + 1))
    else
        mem_counts[$process]=1
    fi
done <<< "$mem_usage"

declare -A avg_mem_totals

# Calculate average mem usage for each command
for process in "${!mem_sum[@]}"; do
    sum=${mem_sum["$process"]}
    count=${mem_counts["$process"]}
    avg_mem=0
```

```
        if (( count > 0 )); then
            avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
        fi
          avg_mem_totals["$process"]=$avg_mem
    done

sorted_entries=()
while IFS== read -r key value; do
  sorted_entries+=("$key=$value")
done < <(for key in "${!avg_mem_totals[@]}"; do echo "$key=${avg_mem_totals[$key]}"; done | sort -t= -
k2n)
echo ""
echo "*********"
echo "Top $m commands with the minimum average memory usage:"
echo ""
for ((i=0; i<m && i<${#sorted_entries[@]}; i++)); do
  entry=${sorted_entries[i]}
  key="${entry%%=*}"
  value="${entry#*=}"
   echo "Command: $key, AVG MEM usage: ($value)"
done
echo ""
    echo "*********"

fi
    ;;
  e)
    read -p "Are you sure you want to exit? " ch
if [[ $ch == "yes" ]]; then
 echo " Exiting... "
  exit 0
fi
```

```
    ;;
  *)
    echo "Invalid selection ! ! , try again "
    ;;
esac
done
```