



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

Introduction:	2
Procedure:	5
1. The program should print on the screen the main menu and ask the user to select an option	5
2. If the user enters 'r':	5
3. If the user enters 'c'	6
4. If the user enters 'i'	7
5. If the user enters 'o':	8
6. If the user enters 'u'	9
7. If the user enters 'a'	11
8. If the user enters 'b'	13
9. If the user enters 'e'	15
Code:	16

Table of figures:

Figure 1. First required Code.	5
Figure 2. First required Output.....	5
Figure 3. selection r Code.....	5
Figure 4. Selection r Output.....	6
Figure 5. Selection c Code	6
Figure 6. Selection c Output.....	7
Figure 7. Selection i Code.....	7
Figure 8. Selection i Output.	8
Figure 9. Selection o Code.....	8
Figure 10. Selection o Output.	9
Figure 11. Selection u Code p1.....	9
Figure 12. Selection u Code p2.....	10
Figure 13. Selection u Code p3.....	10
Figure 14. Selection u Output	11
Figure 15. Selection a Code p1.....	11
Figure 16. Selection a Code p2.....	12
Figure 17. Selection a Code p3.....	12
Figure 18. Selection a Output.	13
Figure 19. Selectin b code p1.....	13
Figure 20. Selectin b code p2.....	14
Figure 21. Selectin b code p3.....	14
Figure 22. Selection b output	15
Figure 23. Selection e code	15
Figure 24. Selection e output.....	15

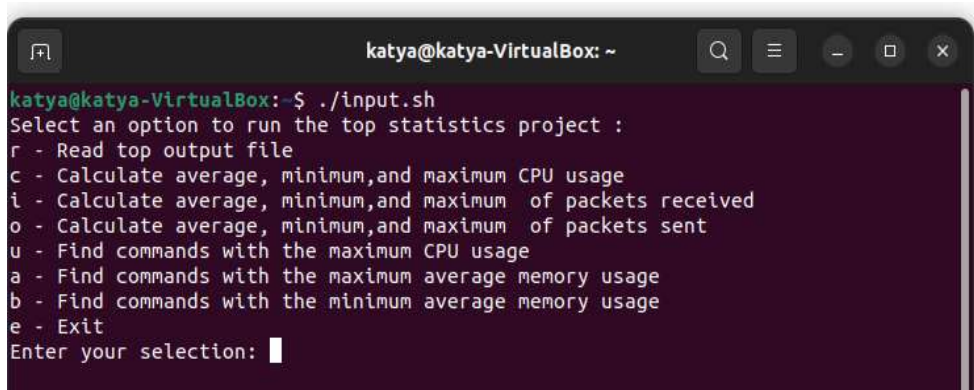
Procedure:

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.

```
1 print_menu() {
2   echo "Select an option to run the top statistics project :"
3   echo "r - Read top output file"
4   echo "c - Calculate average, minimum, and maximum CPU usage"
5   echo "i - Calculate average, minimum, and maximum of packets received"
6   echo "o - Calculate average, minimum, and maximum of packets sent"
7   echo "u - Find commands with the maximum CPU usage"
8   echo "a - Find commands with the maximum average memory usage"
9   echo "b - Find commands with the minimum average memory usage"
10  echo "e - Exit"
11 }
12
13 while true ; do
14   print_menu
15   read -p "Enter your selection: " selection
16   case "$selection" in
17     r)
```

Figure 1. First required Code.



```
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 whether it is existing or not using `-e`.

```
16 case "$selection" in
17   r)
18     #Read top output file
19     read -p "Please enter the name of the file: " name
20     if [ -e "$name" ]
21     then
22       echo "File exists."
23     else
24       echo "File does not exist."
25     fi
26   ;;
27   ;;
28   ;;
29
```

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.

```

29
30 c)
31 # Calculate the average, minimum, and maximum CPU usage
32 if [ -z "$name" ]; then
33     echo "please must do case (r) to read file"
34 else
35     cpu_count=0
36     sum=0
37     min=9999999
38     max=0
39
40 while I= read -r line; do
41     if grep -qE "CPU usage:" <<< "$line"; then
42         cpu_usage=$(echo "$line" | awk -F "CPU usage:" '{print $2}' | awk '{printf "%.3f", $1}')
43         if [ $(echo "$cpu_usage" | awk '{print ($0>0)?($0-0):0}') ]; then
44             sum=$(awk 'BEGIN {print $sum + $cpu_usage}')
45             cpu_count=$((cpu_count + 1))
46
47             if (( $(echo "$cpu_usage < $min" | bc -l) )); then
48                 min=$cpu_usage
49             fi
50
51             if (( $(echo "$cpu_usage > $max" | bc -l) )); then
52                 max=$cpu_usage
53             fi
54         fi
55     fi
56 done < "$name"
57
58 if [ "$cpu_count" -gt 0 ]; then
59     avg=$(awk 'BEGIN {print $sum / $cpu_count}')
60     echo "Average CPU usage: $avg"
61     echo "Minimum CPU usage: $min"
62     echo "Maximum CPU usage: $max"
63 else
64     echo "no CPU usage data found in the file"
65 fi
66 fi
67

```

Figure 5. Selection c Code

```

e - Exit
Enter your selection: c
Average CPU usage: 2.82385
Minimum CPU usage: 1.9880
Maximum CPU usage: 3.9380
Select an option to run the top statistics project :
r - Read top output file
c - Calculate average, minimum, and maximum CPU usage
l - 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.

```

69 l)
70 # Calculate the average, minimum, and maximum number of packets received
71 packets_sum=0
72 packets_count=0
73 packets_minimum=9999999
74 packets_maximum=0
75 if [ -z "$name" ]; then
76     echo "Please must do case (r) to read file"
77 else
78     while IFS= read -r line; do
79         if grep -qE "Networks: packets: " <<< "$line"; then
80             received_packets=$(echo "$line" | awk -F 'Networks: packets: ' '{print $2}' | awk -F '/' '{print $1}')
81             if [[ "$received_packets" =~ ^[0-9]+$ ]]; then
82                 packets_sum=$((packets_sum + received_packets))
83                 packets_count=$((packets_count + 1))
84                 if ((received_packets < packets_minimum)); then
85                     packets_minimum=$received_packets
86                 fi
87                 if ((received_packets > packets_maximum)); then
88                     packets_maximum=$received_packets
89                 fi
90             fi
91         fi
92     done < "$name"
93     if [ $packets_count -gt 0 ]; then
94         packets_average=$((packets_sum / packets_count))
95         echo "Average packets received: $packets_average"
96         echo "Minimum packets received: $packets_minimum"
97         echo "Maximum packets received: $packets_maximum"
98     else
99         echo "No packet data found in the file"
100     fi
101 fi
102 ;;

```

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.

```

112
113 o)
114     packets_sum=0
115     packets_count=0
116     packets_minimum=9999999
117     packets_maximum=0
118
119
120 if [ -z "$name" ]; then
121     echo "please must do case (r) to read file"
122 else
123     while IFS= read -r line; do
124         if [[ $line =~ packets:[[:space:]]+[0-9]+/[0-9]+M[[:space:]]+in,[[:space:]]+[0-9]+/[0-9]+M[[:space:]]+out ]]; then
125             sent_packets=${BASH_REMATCH[1]}
126             if [[ "$sent_packets" =~ ^[0-9]+$ ]]; then
127                 packets_sum=$((packets_sum + sent_packets))
128                 packets_count=$((packets_count + 1))
129                 if ((sent_packets < packets_minimum)); then
130                     packets_minimum=sent_packets
131                 fi
132                 if ((sent_packets > packets_maximum)); then
133                     packets_maximum=sent_packets
134                 fi
135             fi
136         fi
137     done < "$name"
138
139 if [ $packets_count -gt 0 ]; then
140     packets_average=$((packets_sum / packets_count))
141     echo "Average packets sent: $packets_average"
142     echo "Minimum packets sent: $packets_minimum"
143     echo "Maximum packets sent: $packets_maximum"
144 else
145     echo "No packet data found in the file"
146 fi
147 fi
148 ;;

```

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.

```

166 u)
167 # Find commands with the maximum CPU usage
168 if [ -z "$name" ]; then # check if the user entered a file name
169 echo "File not found. Please make sure to read the file by using the case {r}."
170 echo ""
171 else
172 read -p "Enter an integer number: " n
173 while ! [[ $n == ^[0-9]+$ ]]; do # check if the n is an integer
174 echo "Error, Entered value is not an integer. Please try again."
175 read -p "Enter an integer number: " n
176 done
177 #cut data from file
178 start="COMMAND"
179 end="@processes:"
180 output=$(awk "/$start/,/$end/" "$name")
181 out=$(echo "$output" | grep -v "Send" | grep -v "$start")
182 cpu_usage=$(echo "$out" | sed 's/^[\t]* //')
183 declare -A cpu_sum
184 declare -A cpu_counts
185
186 while read -r line; do
187 s=0
188 e=0
189
190 while [ $s -lt ${#line} ]; do
191 current_char="${line:$s:1}"
192
193 if [[ $current_char == [0-9] ]]; then
194 e=$((e+$current_char))
195 break
196 fi
197
198 ((s++))
199 done
200
201 ((e--))
202 process="${line:0:e}"
203
204 g=$((e+1))
205 l=0
206
207 while [ $g -lt ${#line} ]; do
208 current_char="${line:$g:1}"
209 if [[ $current_char == " " ]]; then

```

Figure 11. Selection u Code p1

```

208 correct_char=$((time-g+1))
209 if [[ $current_char == "" ]] then
210     break
211 fi
212
213 ((g++))
214 done
215
216 cpu=$((time-ent-e))
217
218 # Sum the total CPU usage for each process and insert it into the array
219 if [[ -n "${cpu_sum[$process]}" ]]; then
220     cpu_sum[$process]=$((echo "${cpu_sum[$process]} + $cpu" | bc -l))
221 else
222     cpu_sum[$process]=$cpu
223 fi
224
225 # Update the count for each process
226 if [[ -n "${cpu_counts[$process]}" ]]; then
227     cpu_counts[$process]=$((cpu_counts[$process] + 1))
228 else
229     cpu_counts[$process]=1
230 fi
231 done <<< $cpu_usage
232
233 declare -A avg_cpu_totals
234
235 # Calculate average CPU usage for each command
236 for process in "${cpu_sum[@]"; do
237     cpu_sum=${cpu_sum[$process]}
238     count=${cpu_counts[$process]}
239     avg_cpu=$((cpu_sum / count))
240
241     if (( count > 0 )); then
242         avg_cpu=$((awk 'BEGIN {printf "%d.%d", $cpu_sum / $count}'))
243     fi
244     avg_cpu_totals["$process"]=$avg_cpu
245 done
246
247 # sort command according avg cpu
248 sorted_entries1=()
249 while IFS= read -r key1 value1; do
250     sorted_entries1+=("$key1-$value1")
251 done < <{(for key1 in "${!avg_cpu_totals[@]"; do echo "$key1-${avg_cpu_totals[$key1]"; done | sort -t= -k2nr)}
252 echo ""

```

Figure 12. Selection u Code p2

```

252 echo ""
253 echo "Top $m commands with the maximum average cpu usage:"
254 echo ""
255 for ((i=0; i<m && i<${#sorted_entries1[@]}; i++)); do
256     entry=${sorted_entries1[i]}
257     key=${entry%*=*}
258     value=${entry#*=*}
259     echo "Command: $key, AVG CPU usage: ($value)"
260 done
261 echo ""
262 echo "*****"
263 echo ""
264 fi
265 ;;
266

```

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
t - 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.

```

267 a)
268 # Find commands with the maximum mem usage
269 if [ -z "$name" ]; then # check if the user entered a file name
270 echo "File not found. Please make sure to read the file by using the case (r)."
```

```

271 echo ""
272 else
273 read -p "Enter an integer number: " n
274
275 while ! [[ $n == ^[0-9]+$ ]]; do
276 echo "Error: Entered value is not an integer. Please try again."
277 read -p "Enter an integer number: " n
278 done
279 # cut data from input file
280 start="COMMAND"
281 end="Processes:"
282 output=$(awk '/$start/,/$end/' "$name")
283 out=$(echo "$output" | grep -v "$end" | grep -v "$start")
284 mem_usage=$(echo "$out" | sed 's/^[^ ]* //')
285
286 declare -A mem_sum
287 declare -A mem_counts
288
289 while read -r line; do
290 s=0
291 e=0
292
293 while [ $s -lt ${#line} ]; do
294 current_char=${line:$s:1}
295 if [[ $current_char == [0-9] ]]; then
296 s=$((s+1))
297 break
298 fi
299 s=$((s+1))
300 done
301 e=$((e+1))
302 process=${line:0:$e}
303 process=$(echo "$process" | sed 's/^[^ ]* //')
304 mem_l=$(awk -v FS="|" '{ if (NF > 0) print $7 }' <<< "$line:e")
305 mem_l=$(echo "$mem_l" | sed 's/^[^ ]* //')
306 a=0
307 q=0
308 while [ $a -lt ${#mem_l} ]; do
309 current=${mem_l:$a:1}
310 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]=$mem
327 fi
328 # Update the count for each process
329 if [[ -n "${mem_counts[$process]}" ]]; then
330     mem_counts[$process]=$((mem_counts[$process] + 1))
331 else
332     mem_counts[$process]=1
333 fi
334 done <<< "$mem_usage"
335
336 declare -A avg_mem_totals
337
338 # Calculate average mem usage for each command
339 for process in "${!mem_sun[@]"; do
340     sum=${mem_sun["$process"]}
341     count=${mem_counts["$process"]}
342     avg_mem=$((sum / count))
343     if ((count > 0 )); then
344         avg_mem=$(awk 'BEGIN {printf "%.2f\n", $sum / $count}')
345     fi
346     avg_mem_totals["$process"]=$avg_mem
347 done
348 #sort command according avg
349 sorted_entries=()
350 while IFS= read -r key value; do
351     sorted_entries+=("${key}-${value}")
352 done <<< (for key in "${!avg_mem_totals[@]"; do echo "${key}-${avg_mem_totals[$key]}"; done | sort -t- -k2nr)

```

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]=$mem
327 fi
328 # Update the count for each process
329 if [[ -n "${mem_counts[$process]}" ]]; then
330     mem_counts[$process]=$((mem_counts[$process] + 1))
331 else
332     mem_counts[$process]=1
333 fi
334 done <<< "$mem_usage"
335
336 declare -A avg_mem_totals
337
338 # Calculate average mem usage for each command
339 for process in "${!mem_sun[@]"; do
340     sum=${mem_sun["$process"]}
341     count=${mem_counts["$process"]}
342     avg_mem=$((sum / count))
343     if ((count > 0 )); then
344         avg_mem=$(awk 'BEGIN {printf "%.2f\n", $sum / $count}')
345     fi
346     avg_mem_totals["$process"]=$avg_mem
347 done
348 #sort command according avg
349 sorted_entries=()
350 while IFS= read -r key value; do
351     sorted_entries+=("${key}-${value}")
352 done <<< (for key in "${!avg_mem_totals[@]"; do echo "${key}-${avg_mem_totals[$key]}"; done | sort -t- -k2nr)

```

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
t - 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).

```

367 b)
368 # Find commands with the maximum mem usage
369 if [ -z "$name" ]; then # check if the user entered a file name
370 echo "File not found. Please make sure to read the file by using the case (r)."
```

```

371 echo ""
372 else
373 read -p "Enter an integer number: " m
374
375 while ! [[ $m == ^[0-9]+$ ]]; do
376 echo "Error: Entered value is not an integer. Please try again."
377 read -p "Enter an integer number: " m
378 done
379
380 start="COMMAND"
381 end="Processes:"
382
383 output=$(awk "/$start/,/$end/" "$name")
384 out=$(echo "$output" | grep -v "$end" | grep -v "$start")
385 mem_usage=$(echo "$out" | sed 's/^[\t]* //')
386
387 declare -A mem_sum
388 declare -A mem_counts
389
390 while read -r line; do
391 s=0
392 c=0
393
394 while [ $s -lt ${#line} ]; do
395 current_char=${line:$s:1}
396 if [[ $current_char == [0-9] ]]; then
397 s=$((s+1))
398 break
399 fi
400 s=$((s+1))
401 done
402 c=$((c+1))
403 process=${line:0:$s}
404 process=$(echo "$process" | sed 's/^[\t]* //; s/[[\t]]//; s/[[\t]]//')
405 mem_s=$(awk -v FS="[" |> " '{if (NF > 0) print $7}' <<< "$line")
406 mem_t=$(echo "$mem_s" | sed 's/^[\t]* //; s/[[\t]]//; s/[[\t]]//')
407 s=0
408 c=0
409 while [ $s -lt ${#mem_t} ]; do
410 current=${mem_t:$s:1}
411 if [[ $current == " " || $current == "[alpha:] " ]]; then

```

Figure 19. Selectin b code p1

```

411         if [[ $current == " " ]] || $current == [[:alpha:]]; then
412             q=$a
413             break
414         fi
415         ((a++))
416     done
417     mem=${mem_i:0:q}
418     if [[ $current == [K] ]]; then
419         mem=$((mem*1024))
420     elif [[ $current == [M] ]]; then
421         mem=$((mem*1024*1024))
422     fi
423
424     if [[ -n "${mem_sum[$process]}" ]]; then
425         mem_sum[$process]=$((echo "${mem_sum[$process]} + $mem" | bc -l))
426     else
427         mem_sum[$process]=$mem
428     fi
429
430     # Update the count for each process
431     if [[ -n "${mem_counts[$process]}" ]]; then
432         mem_counts[$process]=$((mem_counts[$process] + 1))
433     else
434         mem_counts[$process]=1
435     fi
436 done <<< "$mem_usage"
437
438 declare -A avg_mem_totals
439
440 # Calculate average mem usage for each command
441 for process in "${!mem_sum[@]"; do
442     sum=${mem_sum[$process]}
443     count=${mem_counts[$process]}
444     avg_mem=0
445     if (( count > 0 )); then
446         avg_mem=$(awk "BEGIN {printf \"%.2f\", $sum / $count}")
447     fi
448     avg_mem_totals["$process"]=$avg_mem
449 done
450
451 sorted_entries=()
452 while IFS= read -r key value; do
453     sorted_entries+=("${key=$value}")
454 done < <((for key in "${!avg_mem_totals[@]"; do echo "key=${avg_mem_totals[$key]}"; done | sort -t= -k2n)

```

Figure 20. Selectin b code p2

```

454 done < <((for key in "${!avg_mem_totals[@]"; do echo "key=${avg_mem_totals[$key]}"; done | sort -t= -k2n)
455 echo ""
456 echo "*****"
457 echo "Top $n commands with the minimum average memory usage:"
458 echo ""
459 for ((i=0; i<n && i<${#sorted_entries[@]}; i++)); do
460     entry=${sorted_entries[i]}
461     key=${entry%%=*}
462     value=${entry#*=}
463     echo "Command: $key, AVG MEM usage: ($value)"
464 done
465 echo ""
466 echo "*****"
467 fi
468 fi
469 ::

```

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
t - 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
    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

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
```