CPU resource monitoring using Bash script

1. Create and enter project folder by using 'cd ~' then 'mkdir cpu_monitor' to creates a new directory called cpu_monitor.

```
limjinzhao@osvm:~$ cd ~
limjinzhao@osvm:~$ mkdir cpu_monitor
```

2. Change into the new directory by using 'cd cpu_monitor'.

```
limjinzhao@osvm:~$ cd cpu_monitor
limjinzhao@osvm:~/cpu_monitor$
```

3. Start to write the monitoring script. Create a new file name: CPU.sh with nano text editor by the command 'nano CPU.sh'.

```
osvm@osvm-virtualbox:~/cpu_monitor$ nano CPU.sh
osvm@osvm-virtualbox:~/cpu_monitor$
```

4. CPU.sh script is created to monitor the cpu usage and record the information into a CSV file (will create later). Below are the code needed in CPU.sh script

```
  GNU nano 6.2                                  CPU.sh *
#!/bin/bash
INTERVAL=${1:-60}
>CPU.csv
while true;do
TIMESTAMP=$(date '+%H:%M:%S')
read CPU_USAGE CMD <<< $(ps -eo pcpu,comm --no-headers | sort -k1 -nr | head -n1)
echo "$TIMESTAMP;$CMD;$CPU_USAGE" >> CPU.csv
sleep "INTERVAL"
done
```

- **#!/bin/bash**
  use to indicate that it should be executed using the bash shell

- **INTERVAL=${1:-60}**
  use to specify an interval in second. For the defauls is 60 seconds

- **>CPU.csv**
  use to create and clear the CPU.csv file before data collection started

- **while true;do**
  use to while true loop to continually sample CPU usage at regular intervals

- **TIMESTAMP=$(date '+%H:%M:S')**
  use to ensure the system time format in hour:minute:second

- **read CPU_USAGE CMD <<< $(ps -eo pcpu,comm –no-headers|sort -k1-nr|head -n1)**

use to list the cpu usage percentage and command name, sort the list in descending order based on the cpu usage and select the process with the highest cpu consumption.

- **echo "$TIMESTAMP;$CMD;$CPU_USAGE" >> CPU.csv**
  use to write CPU.csv with the format timestamp;command;cpu_usage

- **sleep "$INTERVAL"**
  use to wait for the specified interval before taking the next measurement

5. Next to run the CPU.sh script, use the command 'chmod +x CPU.sh' to assign the permission. Then use './CPU.sh 4' to run the script with 4s interval. To stop it, press Crtl+C.

```
limjinzhao@osvm:~/cpu_monitor$ chmod +x CPU.sh
limjinzhao@osvm:~/cpu_monitor$ ./CPU.sh 4
^C
limjinzhao@osvm:~/cpu_monitor$
```

6. Record the output into CPU.csv by using ‘cat CPU.csv’.

```
limjinzhao@osvm:~/cpu_monitor$ cat CPU.csv
02:33:59;firefox;23.7
02:34:03;firefox;23.8
02:34:07;firefox;23.8
02:34:11;firefox;23.8
02:34:15;firefox;23.8
02:34:19;firefox;23.8
02:34:23;firefox;23.8
02:34:27;firefox;23.8
02:34:31;firefox;23.8
02:34:35;firefox;23.8
02:34:39;firefox;23.8
02:34:43;firefox;23.8
02:34:47;firefox;23.8
02:34:51;firefox;23.7
02:34:55;firefox;23.7
02:34:59;firefox;23.7
02:35:03;firefox;23.7
02:35:07;firefox;23.7
02:35:11;firefox;23.7
02:35:15;firefox;23.7
02:35:19;firefox;23.7
02:35:23;firefox;23.6
02:35:27;firefox;23.6
02:35:31;firefox;23.6
02:35:35;2048-qt;23.9
02:35:39;2048-qt;24.7
02:35:43;2048-qt;27.0
02:35:47;2048-qt;30.3
02:35:51;2048-qt;32.6
02:35:55;2048-qt;33.9
02:35:59;2048-qt;34.8
02:36:03;2048-qt;35.2
```

```
02:36:07;2048-qt;34.2
02:36:11;2048-qt;32.8
02:36:15;2048-qt;31.5
limjinzhao@osvm:~/cpu_monitor$
```

7. Run the script in the background by the command ‘ ./CPU.sh 4 &’ .

```
limjinzhao@osvm:~/cpu_monitor$ ./CPU.sh 4 &
[1] 50317
```

8. To verify the background process, type the command ‘ ps -ef|grep CPU.sh|grep -v grep’. If there is output means that the script is active.

```
limjinzhao@osvm:~/cpu_monitor$ ps -ef | grep CPU.sh | grep -v grep
limjinz+   50317   39546  0 02:50 pts/0    00:00:00 /bin/bash ./CPU.sh 4
```

9. Now we create a new script CPU-stop.sh script to stop the background process created just now. Below are the code needed in the script.

```
limjinzhao@osvm: ~/cpu_monitor
  GNU nano 6.2                                    CPU-stop.sh
#!/bin/bash
PIDS=$(ps -ef | grep '[C]PU.sh' | awk '{print $2}')
if [ -z "$PIDS"]; then
echo "No running CPU.sh process found."
exit 1
fi
kill $PIDS
echo "CPU.sh process has been stopped: $PIDS"
```

- ✓ #!/bin/bash
  use to indicate that it should be executed using the bash shell
- ✓ PIDS=$(ps -ef | grep '[C]PU.sh' | awk '{print $2}')
  use to find the process id of CPU.sh
- ✓ if [-z "$PIDS"]; then
  check that if the process id list is empty
- ✓ echo "No running CPU.sh process found."
  return the message
- ✓ kill $PIDS
  stops the process
- ✓ echo "CPU.sh process has been stopped: $PIDS"
  return the message

10. Run the stop script by give permission 'chmod +x CPU-stop.sh', then run by ' ./CPU-stop.sh'

```
limjinzhao@osvm:~/cpu_monitor$ chmod +x CPU-stop.sh
limjinzhao@osvm:~/cpu_monitor$ ./CPU-stop.sh
./CPU-stop.sh: line 3: [: missing `]'
CPU.sh process has been stopped: 50317
```

11. Use command ' ps -ef|grep CPU.sh|grep -v grep' to verify. If no result display means it works.

```
limjinzhao@osvm:~/cpu_monitor$ ./CPU-stop.sh
No running CPU.sh process found.
limjinzhao@osvm:~/cpu_monitor$ ps -ef | grep CPU.sh | grep -v grep
limjinzhao@osvm:~/cpu_monitor$
```

RAM resource monitoring program

For steps (a) to (g), show the content of RAM.sh and RAM.csv in your report.

1. Commands in QTerminal

```
stanleychewjunxian@osvm:~$ chmod +x RAM.sh
stanleychewjunxian@osvm:~$ ./RAM.sh 2
^C
```

- use the command 'chmod +x RAM.sh' to assign the required permissions for the RAM.sh script. Then use './RAM.sh 2' to run the script with 2s interval. To stop it, press Crtl+C.

2. Content of RAM.sh

/home/stanleychewjunxian/RAM.sh

File Edit Options Search Help

RAM.sh

```
#!/bin/bash

# Clear previous content in RAM.csv
> RAM.csv

# Set interval from input or default to 60 seconds
interval=${1:-60}

# Infinite loop to monitor RAM usage
while true; do
    # Get current time in HH:MM:SS format
    current_time=$(date +%T)

    # Extract the process with the highest %MEM usage using comm
    read -r p_name mem_usage <<<$(ps -eo comm,%mem --no-headers --sort=-%mem | head -n1)

    # Append data to RAM.csv
    echo "$current_time;$p_name;$mem_usage" >> RAM.csv

    # Sleep for the specified interval
    sleep $interval
done
```

Search...

Encoding: UTF-8 Syntax: sh Lines: 23 Sel. Chars: 0 Words:

- Script was written as blank file (text file) and saved with .sh extension
- Text form:

```
#!/bin/bash

# Clear previous content in RAM.csv
> RAM.csv

# Set interval from input or default to 60 seconds
interval=${1:-60}

# Infinite loop to monitor RAM usage
while true; do
    # Get current time in HH:MM:SS format
    current_time=$(date +%T)
```

```
    # Extract the process with the highest %MEM usage (corrected version)
    read -r p_name mem_usage <<<$(ps -eo comm,%mem --no-headers --sort=-%mem | head
-n1)

    # Append properly formatted data to RAM.csv
    echo "$current_time;$p_name;$mem_usage" >> RAM.csv

    # Sleep for the specified interval
    sleep $interval
done
```

3. Content of RAM.csv

```
stanleychewjunxian@osvm:~$ cat RAM.csv
01:11:58;firefox;23.4
01:12:00;firefox;23.4
01:12:02;firefox;23.4
01:12:04;firefox;23.4
01:12:06;firefox;23.3
01:12:08;firefox;23.3
01:12:10;firefox;23.3
01:12:12;firefox;23.3
01:12:14;firefox;23.3
01:12:16;firefox;23.3
01:12:18;firefox;23.3
```

- view the contents of RAM.csv using ‘cat RAM.csv’

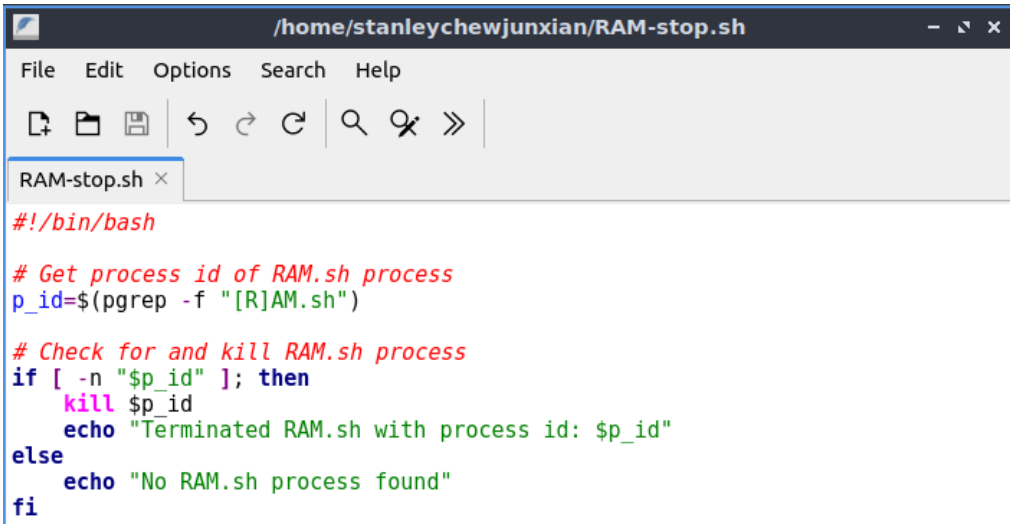
For step (h), show the screenshot in your report. For step (i), show the content of RAM-stop.sh and the verification screenshot in your report.

1. Commands in QTerminal

```
stanleychewjunxian@osvm:~$ ./RAM.sh 2 &
[1] 9039
stanleychewjunxian@osvm:~$ ./RAM-stop.sh
```

- use the command ‘chmod +x RAM-stop.sh’ to assign the required permissions for the RAM-stop.sh script. Then use ‘./RAM.sh 2 &’ to run the script with 2s interval in the background. To stop it, use './RAM-stop.sh'

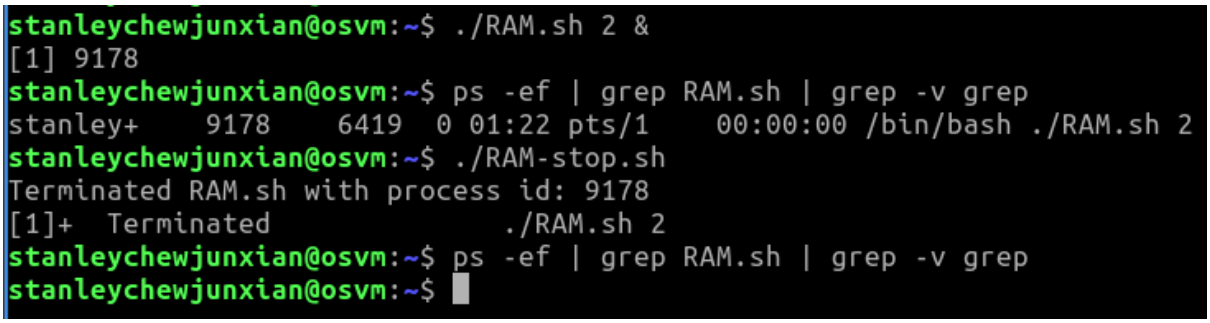
2. Content of RAM-stop.sh



- Script was written as blank file (text file) and saved with the .sh extension.
- Text form:

```
#!/bin/bash

# Get process id of RAM.sh processes
p_ids=$(pgrep -f "[R]AM.sh")

# Check for and kill RAM.sh processes
if [ -n "$p_ids" ]; then
   kill $p_ids
   echo "Terminated RAM.sh processes: $p_ids"
else
   echo "No RAM.sh processes found."
fi
```

3. Verification Screenshot



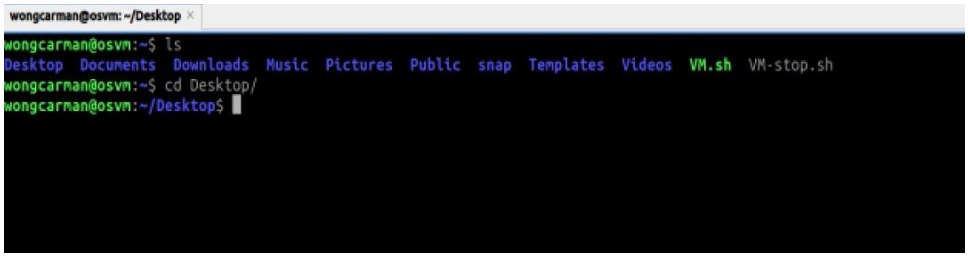
- Use ‘ps -ef | grep RAM.sh | grep -v grep’ after execution and termination to confirm whether RAM.sh is running or not running.
- When RAM.sh is running, the command will return a result.
- When RAM.sh is not running, the command will not return any result.
- Since there is no returned result after using ‘./RAM-stop.sh’, this shows that RAM-stop.sh is working as intended.

Virtual Memory resource monitoring program

Objective :Develop a Bash script (VM.sh) to track processes with the highest virtual memory (VSZ) consumption at configurable intervals.

1. To display all files and directories in the home directory use the command: ***ls***

2. To navigate into the `Desktop` directory, which contains the relevant shell scripts command: cd Desktop/



3.Command: ***vim VM.sh***

- This invokes the `vim` text editor to either create or modify the `VM.sh` shell script. The editing step is typically used to insert or update command sequences within the script.

4. Once editing is complete, execute permissions are assigned using the command: ***chmod +x VM.sh***

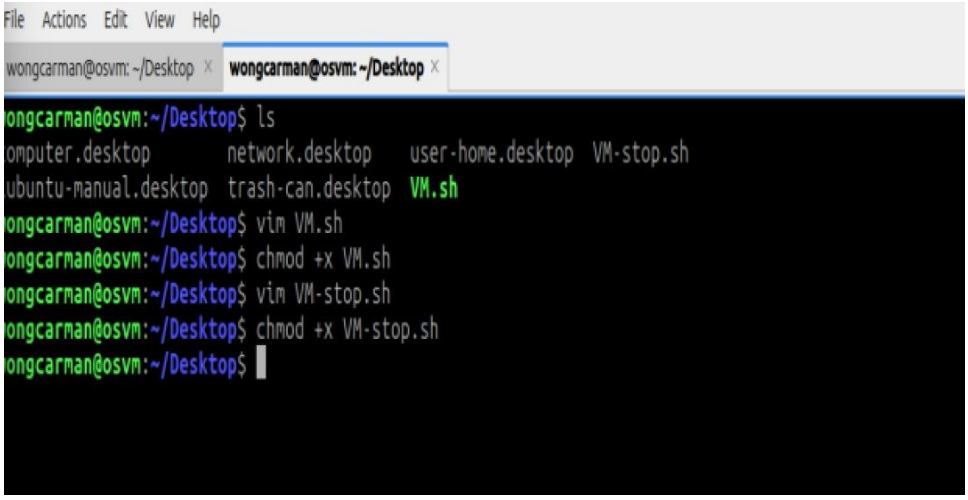
- This action enables the script to be run as a standalone executable program.
- `chmod` is the utility for changing file mode (permissions),
- `+x` adds execute permission,
- `VM.sh` is the target script file.

5. To create or modify the second shell script, the following command is executed: ***vim VM-stop.sh***

- This command opens the `VM-stop.sh` file in the `vim` text editor. The script can be written or edited within this interface, typically to include instructions for stopping or managing a virtual machine or service.

6. After editing is completed and the script is saved, execute permissions are granted using the command: chmod +x VM-stop.sh

- This step allows the script to be executed directly from the terminal using the `./VM-stop.sh` command.



Key Features

- **Monitors Virtual Memory Usage**: Utilizes `ps -eo comm,vsz` to list all active processes along with their virtual memory size (VSZ).
- **Ranks by Memory Consumption**: Sorts the processes in descending order using `--sort=-vsz` to identify the top memory-consuming process.
- **Structured Logging Format**: Appends output to `VM.csv` in a clean, comma-separated format: `HH:MM:SS,process_name,VSZ_bytes`.
- **Customizable Monitoring Interval**: Supports optional time interval input (e.g., `./VM.sh 10` for updates every 10 seconds); defaults to 60 seconds if not specified.

/home/wongcarman/Desktop/VM.sh

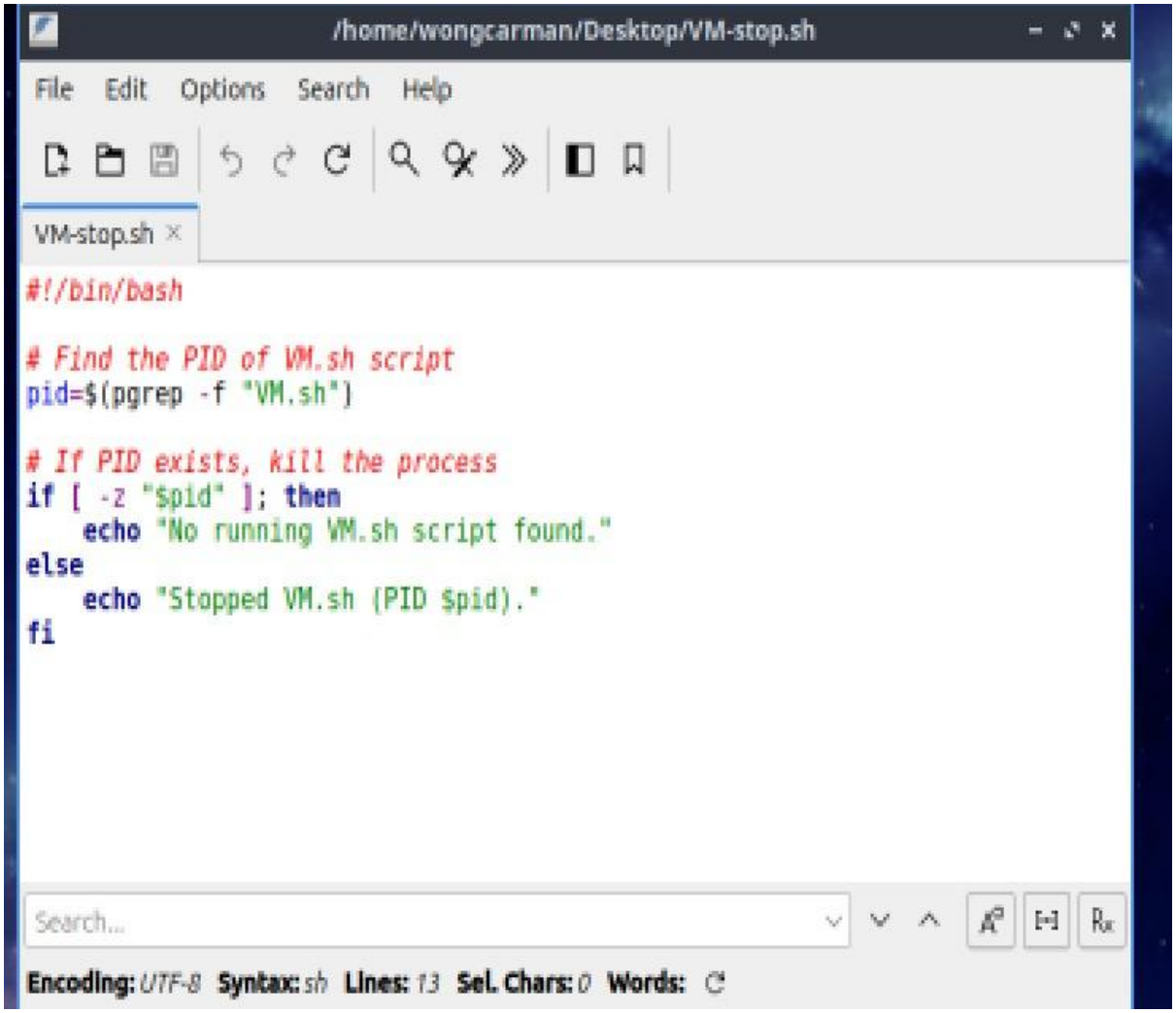
File Edit Options Search Help

VM.sh

```bash
#!/bin/bash

# Check if user provided a sleep time
if [ -z "$1" ]; then
    interval=60
else
    interval=$1
fi

# Clear VM.csv file
> VM.csv

# Infinite loop
while true
do
    # Get current time
    current_time=$(date +"%H:%M:%S")

    # Get the process with the largest VSZ
    process_info=$(ps --no-headers -eo comm,vsz --sort=-vsz | head -n 1)
    process_name=$(echo "$process_info" | awk '{print $1}')
    process_vsz=$(echo "$process_info" | awk '{print $2}')

    # Append the information to VM.csv
    echo "$current_time,$process_name,$process_vsz" >> VM.csv

    # Sleep for the interval
    sleep "$interval"
done
```

Key Features

- **Automatic Script Detection**: Uses `pgrep -f "VM.sh"` to search for the process ID (PID) of the running `VM.sh` script by name.
- **Safe Termination Check**: Verifies whether the script is running before attempting to terminate it, preventing unnecessary errors.
- **Graceful Shutdown**: If found, terminates the `VM.sh` process using the `kill` command and confirms it via console message.
- **User Feedback**: Prints a clear message whether the script was found and stopped, or if it was not running at all.



7. To initiate the memory logging script with a 4-second update interval while keeping the terminal free for other tasks, use the following command: ***./VM.sh 4 &***

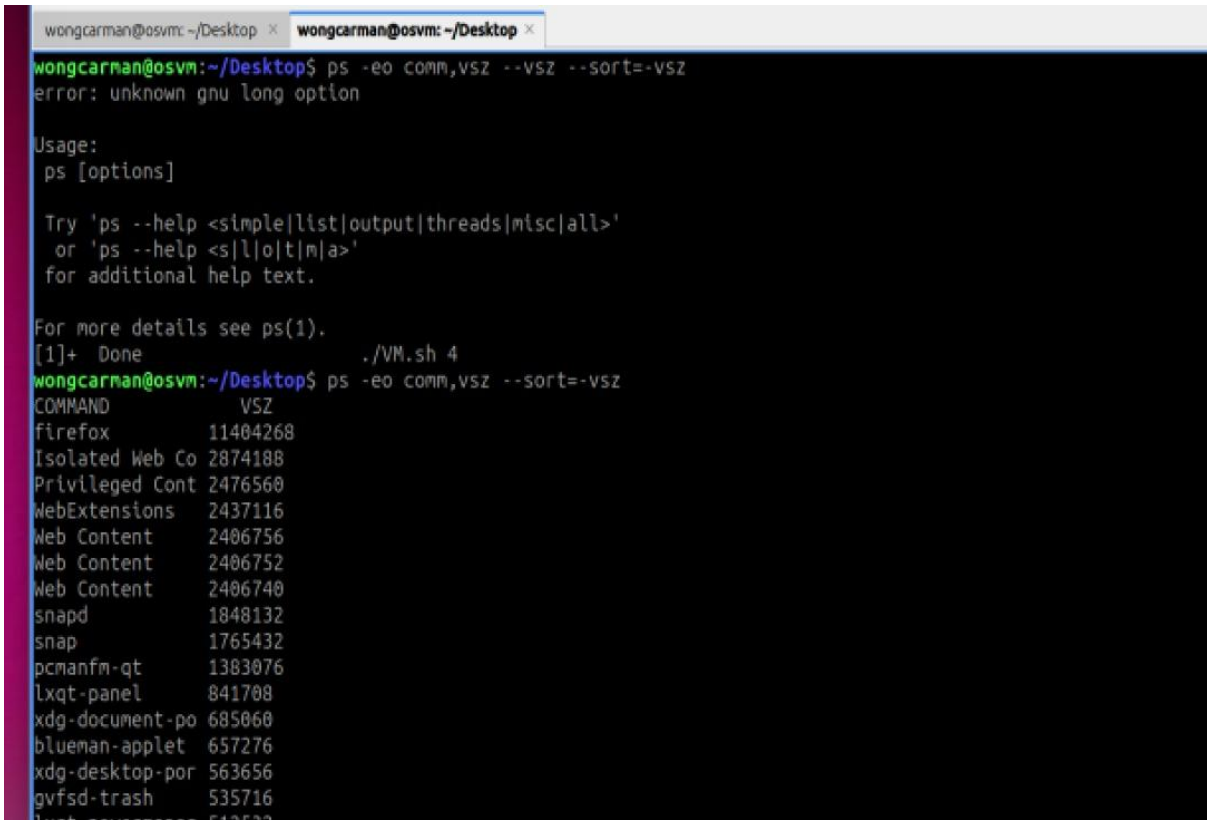
- `./VM.sh` launches the script from the current directory.
- 4 sets the logging interval to **4 seconds**.
- & runs the script **in the background**.
- Upon execution, the shell returns output like:[1] 10032
- [1] is the background job number assigned by the shell.
- 10032 is the PID (Process ID) of the VM.sh script instance.
- This setup allows for high-frequency virtual memory monitoring with minimal terminal disruption, ideal for short-interval analysis or testing.

8. An attempt was made to execute the following command: ***ps -eo comm,vsz -vsz --sort=-vsz***

- This resulted in the error
- The cause of the error is the use of an invalid or mistyped option `-vsz`, which is not recognized by the `ps` command.
- The user accidentally types the wrong command

9. The proper command is then insert to list processes and sort them by virtual memory size (VSZ) in descending order is: ***ps -eo comm,vsz --sort=-vsz***

- ***-eo comm,vsz: Outputs the command name and VSZ memory field.***
- --sort=-vsz: Sorts the result by VSZ in descending order (- indicates reverse sort).



10. To terminate the VM.sh background script command: ***./VM-stop.sh***

- This command runs the VM-stop.sh shell script, which searches for the process ID (PID) of the running VM.sh instance using pgrep, and then attempts to terminate it.

11. To display the contents of the log file command: ***cat VM.csv***

- This command outputs the collected data that was continuously appended by the VM.sh script during execution.
- Process Name: `firefox` is consistently identified as the top consumer of virtual memory.
- Memory Usage (VSZ): The recorded VSZ value fluctuates slightly over time, indicating dynamic memory consumption.

- Time Stamps: Entries are logged at 4-second intervals, which aligns with the user-defined interval set during the script execution (`./VM.sh 4 &`).
- This log provides a time series dataset for memory usage analysis, useful for identifying trends or performance bottlenecks over time.

```
wongcarman@osvm:~/Desktop$ cat VM.csv
19:26:00,firefox,11461644
19:26:04,firefox,11461644
19:26:08,firefox,11461644
19:26:12,firefox,11461644
19:26:16,firefox,11461644
19:26:20,firefox,11461644
19:26:24,firefox,11461644
19:26:28,firefox,11461692
19:26:32,firefox,11461692
19:26:36,firefox,11461692
19:26:40,firefox,11461692
19:26:44,firefox,11461692
19:26:48,firefox,11461692
19:26:52,firefox,11461692
19:26:57,firefox,11461676
19:27:01,firefox,11461644
wongcarman@osvm:~/Desktop$
```

The contents of `VM.csv`, which were generated by the `VM.sh` monitoring script, are shown here loaded into a data viewer or spreadsheet application.

- This view confirms that the data logged by `VM.sh` is properly structured and suitable for further analysis, visualization, or import into tools like Excel, LibreOffice Calc, or data analytics software.

Fields

Column type:

	Standard	Standard	Standard
1	19:26:00	firefox	11461644
2	19:26:04	firefox	11461644
3	19:26:08	firefox	11461644
4	19:26:12	firefox	11461644
5	19:26:16	firefox	11461644
6	19:26:20	firefox	11461644
7	19:26:24	firefox	11461644
8	19:26:28	firefox	11461692
9	19:26:32	firefox	11461692
10	19:26:36	firefox	11461692
11	19:26:40	firefox	11461692

Help

Network usage monitoring program

1.Create a file called network.sh to get total network consumption in kilobytes (KB) and record it into a file called network.csv

```
chanzhunseang@osvm:~$ nano network-stop.sh
```

```
chanzhunseang@osvm: ~
  GNU nano 7.2                          network.sh
#!/bin/bash

> network.csv
INTERVAL=${1:-60}
INTERFACE="ens33"

while true; do

        CURRENT_TIME=$(date +"%H:%M:%S" 2>/dev/null || echo "00:00:00")

        RX_BYTES=$(ip -s link show ens33 | awk '/RX.*bytes/{getline; print $1;}')
        TX_BYTES=$(ip -s link show ens33 | awk '/TX.*bytes/{getline; print $1;}')

        RX_KB=$((RX_BYTES / 1024))
        TX_KB=$((TX_BYTES / 1024))

        echo "$CURRENT_TIME;$RX_KB;$TX_KB" >> network.csv

        sleep $INTERVAL
done
```

First, we will clear previous content in network.csv file.

The loop can use period between 1-60 second, if the user does not enter any input parameter, it will make the period set to a default value 60 seconds.

It will use default network interface ens33.

In the loop, it will capture date in hr:min:sec format, total received data (RX) size and transmitted data (TX) size.

And then will store in network.csv file in data:RX data:Tx data format

2. I design a shell script, called network-stop.sh to stop the background script because cannot stop it using Ctrl + x.

```
chanzhunseang@osvm:~$ nano network-stop.sh
```

```
chanzhunseang@osvm: ~
  GNU nano 7.2                          network-stop.sh
#!/bin/bash

pkill -f "network.sh"

if pgrep -f "network.sh" > /dev/null; then
        echo "Failed to stop network monitoring"
else
        echo "Succefully stop network monitoring"
fi
```

3. After creating network.sh and network-stop.sh file, i will add permissions to these two files

```
chanzhunseang@osvm:~$ chmod +x network.sh network-stop.sh
```

4.I will run it in terminal with 5 second period, here is the output that I run the script (./network.sh 5) in terminal with 5 second period and end it by using Crlt + c.

```
chanzhunseang@osvm:~$ ./network.sh 5
^C
chanzhunseang@osvm:~$ cat network.csv
18:58:16;7350;289
18:58:21;8011;314
18:58:26;8013;316
18:58:31;8676;331
18:58:36;9325;336
18:58:41;10153;344
18:58:46;10808;355
18:58:51;11478;362
18:58:56;12105;370
18:59:01;12111;376
18:59:06;12751;385
```

5. After successful run the network.sh script file in terminal and end it.
   The script sile will test it in backgroud (./network.sh 4 &) with 4 second period

```
chanzhunseang@osvm: ~
File  Actions  Edit  View  Help
chanzhunseang@osvm: ~
chanzhunseang@osvm:~$ ./network.sh 4 &
[1] 15171
chanzhunseang@osvm:~$ cat network.csv
19:51:15;21385;918
19:51:19;21387;919
19:51:23;21388;920
19:51:27;21390;921
19:51:31;21393;923
19:51:35;21396;924
19:51:39;21396;925
19:51:43;21398;926
19:51:47;21399;927
19:51:51;21401;928
19:51:55;21403;930
19:51:59;21406;932
19:52:03;21406;933
19:52:07;21407;933
19:52:11;21407;934
19:52:15;21408;935
19:52:19;21409;935
19:52:23;21410;936
19:52:27;21411;936
```

6. If the script run it in background, it cannot end it with Crlt + c, it will keep running. So, i will run the network-stop.sh script file (./network-stop.sh) to stop it

```
chanzhunseang@osvm:~$ ps -all
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME
CMD
0 S  1000   15171    1799  0  80   0 -  2486 do_wai pts/0    00:00:00
network.sh
0 S  1000   15596   15171  0  80   0 -  2073 hrtime pts/0    00:00:00
sleep
4 R  1000   15597    1799 99  80   0 -  3447 -      pts/0    00:00:00
ps
chanzhunseang@osvm:~$ ./network-stop.sh
Succefully stop network monitoring
[1]+  Terminated              ./network.sh 4
chanzhunseang@osvm:~$ ps -all
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME
CMD
4 R  1000   15610    1799  0  80   0 -  3447 -      pts/0    00:00:00
ps
```