UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

LAB 2 : PROCESS

SECR2043 - OPERATING SYSTEMS

Semester 2 2024/2025

Section 01

NAME	MATRIC NUMBER
AMELIA ADLINA BINTI AZRUL	A23CS0043

Lecturer: **DR. MUHAMMAD ZAFRAN BIN MUHAMMAD ZALY SHAH**

Date: 15th May 2025

SCR2043 OPERATING SYSTEMS

Name	: AMELIA ADLINA BINTI AZRUL	Marks
Student ID	: A23CS0043	
Section	: 01	

This lab assessment is designed to test your understanding and skills on some basic concepts and tools related to process monitoring and management in operating system. Please follow the instructions carefully and submit your answers in this word document and rename the file as **os-lab-assessment02-studentname-matricno.docx**.

Essential Steps Before Starting Lab Assessment 2:

1. **Download necessary source codes:**

   Use the `wget` command to retrieve the following source code files to your Linux (or WSL or MacOS) environment:

```
wget -O mainprocess.c https://rebrand.ly/mainprocess_c
wget -O subprocess1.c https://rebrand.ly/subprocess1_c
wget -O subprocess2.c https://rebrand.ly/subprocess2_c
```

2. **Compile the source files:**

   Use the `gcc` compiler to create executable files from the source code.

```
gcc mainprocess.c -o mainprocess
gcc subprocess1.c -o subprocess1
gcc subprocess2.c -o subprocess2
```

3. **Execute the dummy processes:**

   Run all the dummy processes

```
./mainprocess &
```

   Press **enter** two times.

4. The dummy processes are running for 2 hours. If you took longer than 2 hours on questions 1-9, please restart the main process with `./mainprocess &`.

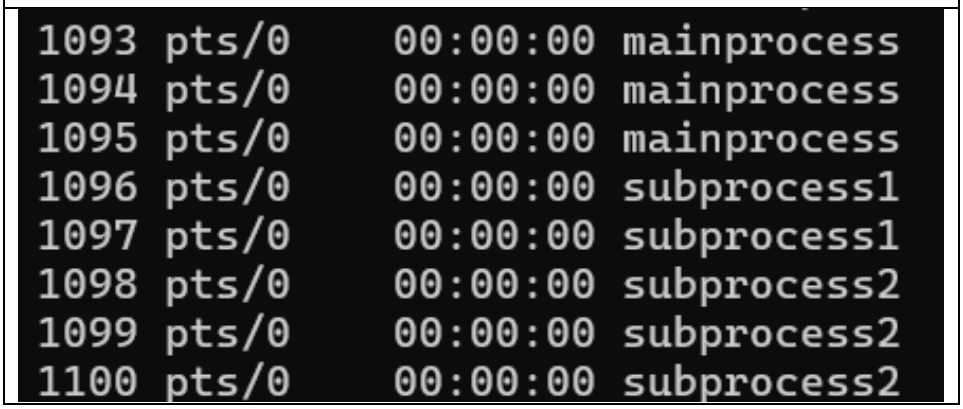
Lab Assessment 2 : Linux Process Monitoring and Management

Instructions:

1. Carefully execute each command as instructed in the questions.
2. Write down the exact command used for each task.
3. Capture a screenshot of the command's output.

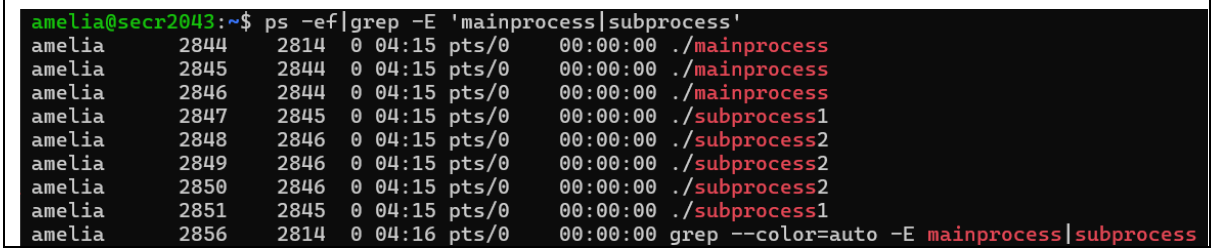
Question 1

Use the `ps` command with the appropriate option to display a complete list of all running processes within the Linux operating system.

Command
`ps -e`
Output


Question 2

Employ the `ps` command with necessary options to unveil comprehensive details about each running process.

Command
`ps -ef|grep -E 'mainprocess|subprocess'`
Output


Question 3

Use the `ps` command with some tools to only list processes named "subprocess" and show some info about them.

Command
`ps -ef|grep -E 'subprocess'`

Output

```
amelia@secr2043:~$ ps -ef|grep -E 'subprocess'
amelia      2847    2845  0 04:15 pts/0    00:00:00 ./subprocess1
amelia      2848    2846  0 04:15 pts/0    00:00:00 ./subprocess2
amelia      2849    2846  0 04:15 pts/0    00:00:00 ./subprocess2
amelia      2850    2846  0 04:15 pts/0    00:00:00 ./subprocess2
amelia      2851    2845  0 04:15 pts/0    00:00:00 ./subprocess1
amelia      2858    2814  0 04:16 pts/0    00:00:00 grep --color=auto -E subprocess
```

Question 4

Execute the `ps` command, specifying options that reveal only the following columns:

- Process ID (`pid`)
- Owner of the process (`user`)
- CPU percentage (`pcpu`)
- Memory percentage (`pmem`)
- Command (`cmd`)

Command
`ps -eo pid,user,pcpu,pmem,cmd | grep -E 'subprocess'`

Output

```
amelia@secr2043:~$ ps -eo pid,user,pcpu,pmem,cmd | grep -E 'subprocess
   2847 amelia    0.0  0.1 ./subprocess1
   2848 amelia    0.0  0.1 ./subprocess2
   2849 amelia    0.0  0.1 ./subprocess2
   2850 amelia    0.0  0.1 ./subprocess2
   2851 amelia    0.0  0.1 ./subprocess1
   2863 amelia    0.0  0.1 grep --color=auto -E subprocess
```

Question 5

Building on the ps command used in Question 4, can you add an option to sort the listed processes by their memory usage (`pmem`)?

Command
`ps -eo pid,user,pcpu,pmem,cmd --sort=pmem | grep -E 'subprocess'`
Output

```
amelia@secr2043:~$ ps -eo pid,user,pcpu,pmem,cmd --sort=pmem | grep -E 'subprocess
   2847 amelia    0.0  0.1 ./subprocess1
   2848 amelia    0.0  0.1 ./subprocess2
   2849 amelia    0.0  0.1 ./subprocess2
   2850 amelia    0.0  0.1 ./subprocess2
   2851 amelia    0.0  0.1 ./subprocess1
   2865 amelia    0.0  0.1 grep --color=auto -E subprocess
```

Question 6

Construct a command using `ps`, suitable options, and any additional tools to visualize the hierarchical structure (tree-like) of the following processes:

- "mainprocess"
- "subprocess1"
- "subprocess2"

Command
`ps --forest -C mainprocess -C subprocess1 -C subprocess2`
Output

```
amelia@secr2043:~$ ps --forest -C mainprocess -C subprocess1 -C subprocess2
    PID TTY          TIME CMD
   2844 pts/0    00:00:00 mainprocess
   2845 pts/0    00:00:00  \_ mainprocess
   2847 pts/0    00:00:00  |   \_ subprocess1
   2851 pts/0    00:00:00  |   \_ subprocess1
   2846 pts/0    00:00:00  \_ mainprocess
   2848 pts/0    00:00:00      \_ subprocess2
   2849 pts/0    00:00:00      \_ subprocess2
   2850 pts/0    00:00:00      \_ subprocess2
```

Question 7

Use `pstree` command with option that show the number of threads to each process.

Command
`pstree -c -s 2231`
Output

```
amelia@secr2043:~$ pstree -c -s 2844
systemd---sshd---sshd---sshd---bash---mainprocess-+-mainprocess-+-subprocess1
                                                  |             `-subprocess1
                                                  `-mainprocess-+-subprocess2
                                                                |-subprocess2
                                                                `-subprocess2
```

Question 8

Use `renice` command to change priority level of one of process “subprocess1”.

Command
`sudo renice -5 2847`
Output

```
amelia@secr2043:~$ ps -o pid,nice,comm -C subprocess1
    PID  NI COMMAND
   2847   0 subprocess1
   2851   0 subprocess1
amelia@secr2043:~$ sudo renice -5 2847
[sudo] password for amelia:
Sorry, try again.
[sudo] password for amelia:
2847 (process ID) old priority 0, new priority -5
```

Question 9

Terminate all running processes with the name “mainprocess”.

Command
`killall -15 mainprocess`
Output

```
amelia@secr2043:~$ killall -15 mainprocess
Main process (ID: 2844) received signal: 15. Terminating...
Main process (ID: 2845) received signal: 15. Terminating...
Main process (ID: 2846) received signal: 15. Terminating...
[1]+  Done                    ./mainprocess
```

Question 10

Write a short C or Python code (choose only one language) demonstrating multiprocessing with `fork()` and `wait()`. Compile and/or run the code. Show the output.

Source Code:

```
nano example.c
gcc example.c -o example
gcc./example

example.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <time.h>

void child_process() {
 printf("Child process with PID: %d\n", getpid());
 int sleep_time = rand() % 5 + 1;
 printf("Child process sleeping for %d seconds\n", sleep_time);
 sleep(sleep_time);
 printf("Child process exiting\n");
}

int main() {
 printf("Parent process with PID: %d\n", getpid());

 // Fork a child process
 pid_t pid = fork();

 if (pid == 0) {
 // This is the child process
 child_process();
 exit(0);
 } else if (pid > 0) {
 // This is the parent process
 printf("Parent process waiting for child process to finish\n");
 // Wait for the child process to finish
 wait(NULL);
 printf("Parent process exiting\n");
 } else {
 // Error occurred while forking
 perror("fork");
 return 1;
 }

 return 0;
}
```

Output:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <time.h>
void child_process() {
 printf("Child process with PID: %d\n", getpid());
 int sleep_time = rand() % 5 + 1;
 printf("Child process sleeping for %d seconds\n", sleep_time);
 sleep(sleep_time);
 printf("Child process exiting\n");
}
int main() {
 printf("Parent process with PID: %d\n", getpid());
 // Fork a child process
 pid_t pid = fork();
 if (pid == 0) {
 // This is the child process
 child_process();
 exit(0);
 } else if (pid > 0) {
 // This is the parent process
 printf("Parent process waiting for child process to finish\n");
 // Wait for the child process to finish
 wait(NULL);
 printf("Parent process exiting\n");
 } else {
 // Error occurred while forking
 perror("fork");
 return 1;
 }
 return 0;
}
```

Output for example.c

```
amelia@secr2043:~$ gcc example.c -o example
amelia@secr2043:~$ ./example
Parent process with PID: 2898
Parent process waiting for child process to finish
Child process with PID: 2899
Child process sleeping for 4 seconds
Child process exiting
Parent process exiting
```