

Name: Hồ Hồng Hà

ID: 20520480

Class: IT007.M13.1

OPERATING SYSTEM LAB 3'S REPORT

SUMMARY

Task		Status	Page
Section 3.5	Bài tập 1	Hoàn thành	2
	Bài tập 2	Hoàn thành	3
	Bài tập 3	Hoàn thành	4
	Bài tập 4	Hoàn thành	5

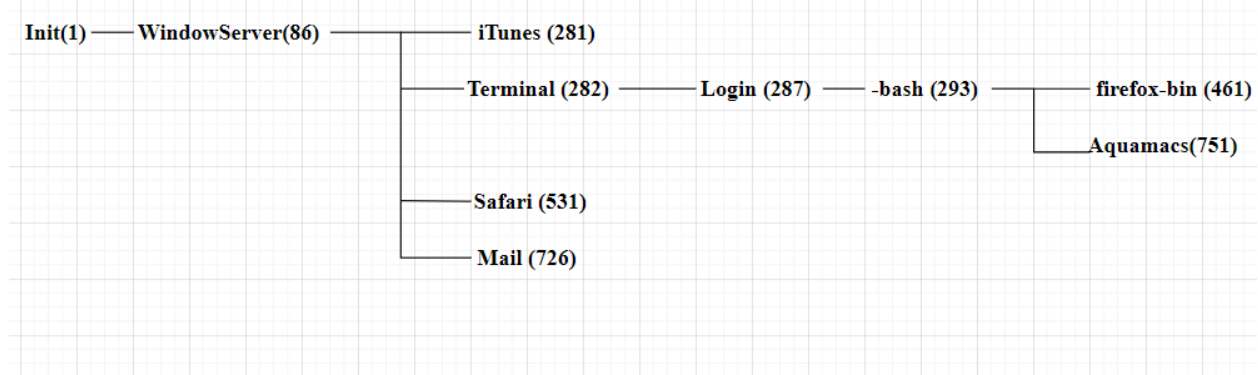
Self-scores: 6

Note: Export file to **PDF and name the file by following format:
LAB X – <Student ID>.pdf*

Section 3.5

Bài 1: Mối quan hệ cha-con giữa các tiến trình

a. Vẽ cây quan hệ parent-child của các tiến trình



b. Trình bày cách sử dụng lệnh ps để tìm tiến trình cha của một tiến trình dựa vào PID của nó.

-Sử dụng lệnh ps -f <PID của tiến trình> để tìm PID của tiến trình cha(PPID). Sau đó tiếp tục thực hiện lệnh ps -f <PID của tiến trình cha(PPID)> để hiển thị thông tin của tiến trình cha.

```
hongha@hongha-VirtualBox:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
hongha      15698    15686  0   08:40 pts/0    00:00:00 bash
hongha      15734    15698  0   08:45 pts/0    00:00:00 ps -f
hongha@hongha-VirtualBox:~$ ps -f 15698
UID          PID     PPID  C  STIME TTY          TIME CMD
hongha      15698    15686  0   08:40 pts/0    00:00:00 bash
hongha@hongha-VirtualBox:~$ ps -f 15686
UID          PID     PPID  C  STIME TTY          TIME CMD
hongha      15686       900  0   08:40 ?        00:00:01 /usr/libexec/gnome-termin
```

c. Tìm hiểu và cài đặt lệnh pstree (nếu chưa được cài đặt), sau đó trình bày cách sử dụng lệnh này để tìm tiến trình cha của một tiến trình dựa vào PID của nó.

-Sử dụng lệnh pstree -p -s <PID của tiến trình> để hiển thị cây tiến trình. Trong đó -p là để xuất ra PID của tiến trình, -s để xuất ra tiến trình cha của tiến trình được chỉ định.

```
hongha@hongha-VirtualBox:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
hongha      15698    15686  0   08:40 pts/0    00:00:00 bash
hongha      15745    15698  0   08:53 pts/0    00:00:00 ps -f
hongha@hongha-VirtualBox:~$ pstree -p -s 15698
systemd(1)---systemd(900)---gnome-terminal-(15686)---bash(15698)---pstree(1574+
```

Bài 2: Chương trình bên dưới in ra kết quả gì? Giải thích tại sao?

*Chương trình được sửa lại như sau:

```
#include<stdio.h>
#include<wait.h>
#include<stdlib.h>
#include<unistd.h>

int main(){
    pid_t pid;
    int num_coconuts = 17;
    pid = fork();
    if(pid == 0) {
        num_coconuts = 42;
        exit(0);
    } else {
        wait(NULL); /*wait until the child terminates */
    }
    printf("I see %d coconuts!\n", num_coconuts);
    exit(0);
}
```

*Chương trình in ra kết quả là dòng chữ: I see 17 coconuts!

Vì tiến trình con đã kết thúc khi gặp lệnh exit(0) trước khi thực thi lệnh printf, do đó tiến trình cha sẽ thực hiện lệnh printf. Do tiến trình cha và tiến trình con có bộ nhớ riêng nên giá trị của num_coconuts vẫn giữ giá trị là 17.

*Kết quả thực thi chương trình trên ubuntu

```
hongha@hongha-VirtualBox:~/LAB3$ gcc exercise_2.c -o ex
hongha@hongha-VirtualBox:~/LAB3$ ./ex
I see 17 coconuts!
```

Bài 3: Liệt kê 10 hàm pthread và chức năng

STT	Tên	Chức năng
1	pthread_attr_init()	Initialize a thread-attribute object
2	pthread_attr_destroy()	Destroy a thread-attribute object
3	pthread_attr_getdetachstate()	Get thread detach state attribute
4	pthread_attr_getscope()	Get thread contention scope attribute
5	pthread_attr_setschedpolicy()	Set the thread scheduling policy attribute
6	pthread_attr_setstacksize()	Set the thread stack-size attribute
7	pthread_attr_getstacksize()	Get the thread stack-size attribute
8	pthread_attr_setguardsize()	Set the size of the thread's guard area
9	pthread_attr_setstackaddr()	Set the thread stack address attribute
10	pthread_attr_getstackprealloc()	Get the amount of memory to preallocate for a thread's <i>MAP_LAZY</i> stack

Bài 4: Viết chương trình làm các công việc sau theo thứ tự:

- In ra dòng chữ: "Welcome to IT007, I am <your_Student_ID>!"
- Mở tệp abcd.txt bằng vim editor
- Tắt vim editor khi người dùng nhấn CTRL+C
- Khi người dùng nhấn CTRL+C thì in ra dòng chữ: "You are pressed CTRL+C! Goodbye!"

*Sử dụng lệnh `gnome-terminal --tab -- vim abcd.txt` để mở tệp abcd.txt trong cửa sổ mới

*Sử dụng lệnh `killall vim` để tắt vim editor bằng tên

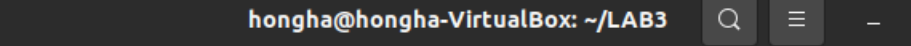
```
hongha@hongha-VirtualBox: ~/LAB3
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>
#include<unistd.h>

int close_vim=0;

void openVim()
{
    system("gnome-terminal --tab -- vim abcd.txt");
}

void on_sigint()
{
    system("killall vim");
    close_vim=1;
}

int main()
{
    signal(SIGINT,on_sigint);
    printf("Welcome to IT007,I am 20520480\n");
    openVim();
    while(close_vim!=1){}
    printf("\nYou are pressed CTRL+C!Goodbye!\n");
    return 0;
}
```



The screenshot shows a terminal window with a dark background. The title bar at the top reads "hongha@hongha-VirtualBox: ~/LAB3". Below the title bar, the terminal prompt is "hongha@hongha-VirtualBox: ~/LAB3". The user enters the command "gcc exercise4.c -o ex4", followed by a new line where they enter "./ex4". The output of the program is "Welcome to IT007,I am 20520480".

```
hongha@hongha-VirtualBox: ~/LAB3
hongha@hongha-VirtualBox:~/LAB3$ gcc exercise4.c -o ex4
hongha@hongha-VirtualBox:~/LAB3$ ./ex4
Welcome to IT007,I am 20520480
```

Kết quả khi mở tệp abcd.txt bằng vim trong cửa sổ mới

Kết quả sau khi nhấn CTRL+C

