

Name: Phạm Mai Dung

ID: 19520477

Class: IT007.L21.1

**OPERATING SYSTEM
LAB 3'S REPORT**

SUMMARY

Task		Status	Page
Section 1.5	Task name 1	Done	2 – 4
	Task name 2	Done	4 – 5
	Task name 3	Done	5 – 7
	Task name 4	Done	7 – 10
...	...		
	...		

Self-scores: 9

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

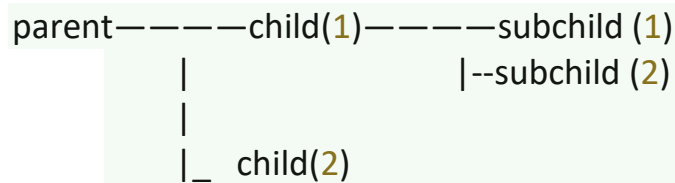
Section 1.5

1. Task name 1

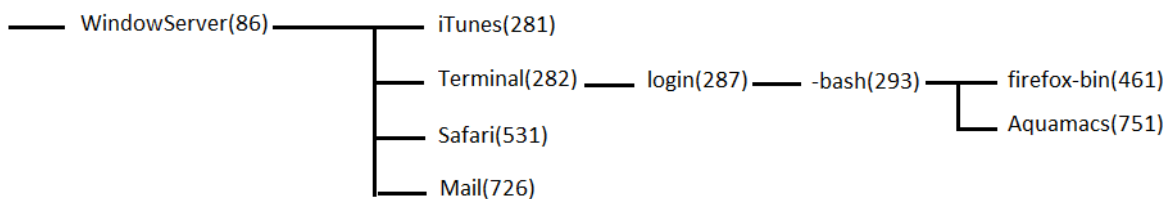
a. Vẽ cây quan hệ parent-child của các tiến trình bên dưới:

UID	PID	PPID	COMMAND
88	86	1	WindowServer
501	281	86	iTunes
501	282	86	Terminal
0	287	282	login
501	461	293	firefox-bin
501	531	86	Safari
501	726	86	Mail
501	751	293	Aquamacs
501	293	287	-bash

- Cấu trúc cơ bản cây tiến trình:



- Thực hiện:



b. Cách sử dụng lệnh ps để tìm tiến trình cha của một tiến trình dựa vào PID của nó:

- Để xem thông tin shell hiện tại, ta gõ lệnh ps. Trong ví dụ Hình 1.b dưới đây, có 2 tiến trình đang được chạy và có PID (mã tiến trình) là 3088 và 9083.

- Để xem PPID (mã tiến trình cha của tiến trình hiện tại), ta gõ lệnh **ps -o ppid <pid_num>**. Ví dụ: **ps -o ppid 3088**. Hình 1.b hiển thị một vài ví dụ cho lệnh này.

```
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps
  PID TTY          TIME CMD
 3088 pts/0        00:00:00 bash
 9083 pts/0        00:00:00 ps
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 3088
PPID
3077
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 9083
PPID
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 3077
PPID
1430
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 1430
PPID
1
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 1
PPID
0
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ps -o ppid 0
error: process ID out of range

Usage:
ps [options]

Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s|l|o|t|m|a>'
for additional help text.

For more details see ps(1).
```

Hình 1.b _ lệnh ps tìm tiến trình cha dựa vào PID đã biết

- c. Cách sử dụng lệnh pstree để tìm tiến trình cha của một tiến trình dựa vào PID của nó:
 - Lệnh ps hiển thị tiến trình có PID là 2221 (Hình 1.c). Để tìm tiến trình cha khi đã biết PID và thể hiện quan hệ parent-child, ta có thể sử dụng lệnh **pstree -s <PID đã biết>**. Lệnh này sẽ hiển thị một cây tiến trình tương tự như cấu trúc trên phần 1.a mô tả. Hình 1.c là một ví dụ.
 - Note: nếu ta muốn xem cả PID của các tiến trình trên cây, ta có thể sử dụng lệnh **pstree -sp <PID đã biết>**.

- Ví dụ: trong hình 1.c, ta tìm tiến trình cha của bash có PID là 2221, dùng lệnh `ps` để hiển thị cây tiến trình, tiến trình cha là tiến trình nằm trước nó: `gnome-terminal-(2210)`.

```
dung-19520477@dung19520477-VirtualBox:~$ ps
  PID TTY          TIME CMD
 2221 pts/0        00:00:00 bash
 2245 pts/0        00:00:00 ps
dung-19520477@dung19520477-VirtualBox:~$ pstree -s 2221
systemd---systemd---gnome-terminal---bash---pstree
dung-19520477@dung19520477-VirtualBox:~$ pstree -sp 2221
systemd(1)---systemd(1319)---gnome-terminal-(2210)---bash(2221)---pstree(2252)
dung-19520477@dung19520477-VirtualBox:~$
```

Hình 1.c _ Lệnh `pstree` tìm tiến trình cha dựa trên PID đã biết.

2. Task name 2

- Chương trình đã cho:

```
/*#####
#University of Information Technology
#IT007 Operating System
#Dung Pham, 19520477
#File: bai02.c
#####*/

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

int main()
{
    pid_t pid;
    int num_coconuts = 17;
    pid=fork();
    if(pid==0)
    {
        num_coconuts = 42;
        exit(0);
    }
    else
    {
        wait(NULL);
    }
    printf("I see %d coconuts!\n",num_coconuts);
    exit(0);
}
"bai02.c" 29L, 542C                                     5,12                                     Top
```

Hình 2.a _ Chương trình in ra giá trị coconuts

- Kết quả khi thực thi chương trình:

```
dung-19520477@dung19520477-VirtualBox:~/LAB3$ vim bai02.c
dung-19520477@dung19520477-VirtualBox:~/LAB3$ gcc bai02.c -o bai02
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ./bai02
I see 17 coconuts!
dung-19520477@dung19520477-VirtualBox:~/LAB3$
```

Hình 2.b _ Kết quả khi thực hiện chương trình

- Giải thích:
 - Đầu tiên ta khởi tạo một biến pid (kiểu pid_t), biến num_coconuts (kiểu int) và được gán cho giá trị là 17. Gán lệnh fork() cho pid. Lệnh fork() sẽ tạo ra 2 tiến trình là tiến trình cha và tiến trình con cùng chạy song song
 - Khi pid==0, tiến trình con thực thi các lệnh bên trong if, đó là gán giá trị 42 cho biến num_coconuts; tiếp theo là lệnh exit(0) kết thúc tiến trình con và trả lại tài nguyên, biến num_coconuts được trả lại giá trị 17.
 - Khi pid>0, tiến trình cha thực thi, lệnh wait(NULL) làm cha đợi con thực thi kết thúc cha mới chạy, mà khi tiến trình con kết thúc đã trả về 17 cho biến num_coconuts, nên khi lệnh printf() được gọi đã in ra giá trị của biến num_coconuts là 17 và sau đó tiến trình cha kết thúc, nên ta có kết quả như trên hình 2.b.

3. Task name 3

- POSIX Thread (pthread) cung cấp các giao diện lập trình thread trên ngôn ngữ C/C++, cho phép chúng ta tạo ra các ứng dụng chạy song song theo luồng, phù hợp với các hệ thống đa bộ xử lý.

Hàm thay đổi thuộc tính	Mục đích	Giá trị trả về
pthread_attr_init(&tattr) note: biến tattr có type pthread_attr_t	reset giá trị mặc định của tất cả/ một thuộc tính mà được chọn	0

<code>pthread_attr_destroy()</code>	Loại bỏ bộ nhớ cấp phát trong quá trình khởi tạo. Đối tượng thuộc tính trở nên không hợp lệ.	0
<code>pthread_attr_setdetachstate()</code>	Kết hợp/ tách rời (em chưa rõ)	0
<code>pthread_attr_setguardsize()</code> giá trị mặc định: PAGESIZE	Cung cấp khả năng bảo vệ chống tràn con trỏ ngăn xếp (tiểu trình không dùng quá không gian được cấp phát)	Hàm thất bại nếu: + Argument attr is invalid + the argument guardsize is invalid + or the argument guardsize contains an invalid value.
<code>pthread_attr_setscope()</code>	Create (tạo) a bound thread (PTHREAD_SCOPE_SYSTEM) or an unbound thread (PTHREAD_SCOPE_PROCESS).	0
<code>pthread_attr_setschedpolicy()</code>	Lập lịch. Tiêu chuẩn dự kiến POSIX chỉ định các thuộc tính lập lịch SCHED_FIFO(first in first out), SCHED_RR (round-robin) và SCHED_OTHER	0
<code>pthread_attr_setinheritsched()</code>	Đặt chính sách lập lịch kế thừa	0

<code>pthread_attr_setschedparam()</code>	Đặt các thông số lập lịch. Các tham số được xác định trong cấu trúc <code>param</code> , chỉ <code>priority</code> được hỗ trợ. Luồng mới được tạo chạy với <code>priority</code> .	0 The value of <i>param</i> is NULL or <i>tattr</i> is invalid.
<code>pthread_attr_setstacksize()</code>	Xác định kích thước ngăn xếp (tính bằng byte) mà hệ thống sẽ cấp phát. Kích thước không được nhỏ hơn kích thước ngăn xếp tối thiểu do hệ thống xác định.	0 Giá trị được trả về nhỏ hơn giá trị của <code>PTHREAD_STACK_MIN</code> hoặc vượt quá giới hạn do hệ thống áp đặt hoặc <i>tattr</i> không hợp lệ.
<code>pthread_attr_setstackaddr()</code>	Thuộc tính <code>stackaddr</code> xác định cơ sở ngăn xếp của luồng. Nếu được đặt non-null (default NULL) thì hệ thống sẽ khởi tạo ngăn xếp tại địa chỉ đó.	0 Giá trị của <code>base</code> hoặc <i>tattr</i> không chính xác

4. Task name 4

- Chương trình thực hiện các công việc theo yêu cầu:

```
/*#####  
#University of Information Technology  
#IT007 Operating System  
#Pham Mai Dung, 19520477  
#File: bai04.c  
#####*/  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <signal.h>  
  
int loop = 1;  
  
void kill_vimeditor()  
{  
    system("kill -9 `pidof vim`");  
    loop = 0;  
}  
  
void on_signint()  
{  
    printf("\nYou are pressed CTRL+C! Goodbye!\n");  
    loop = 0;  
}  
  
int main()  
{  
    printf("\nWelcome to IT007, I am Mai Dung_19520477!\n");  
    system("gnome-terminal -e 'vim abcd.txt'");  
    loop = 1;  
    signal(SIGINT, kill_vimeditor);  
    while(loop) {}  
    loop = 1;  
    signal(SIGINT, on_signint);  
    while(loop) {}  
    exit(0);  
}
```

27,0-1 Bot

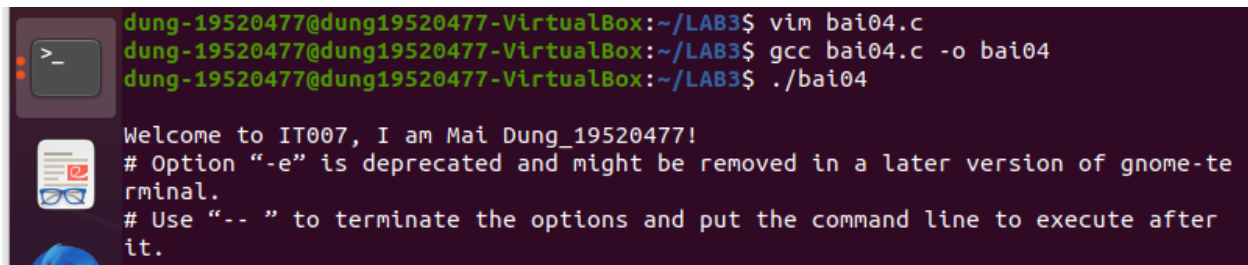
Hình 4.a _ Source code chương trình thực hiện các yêu cầu (file: bai04.c)

- Để giải thích chương trình đã làm những công việc gì, ta dùng lệnh

\$ gcc bai04.c -o bai04

\$./bai04

để biên dịch và thực thi chương trình. Các hình dưới đây là kết quả khi chạy:

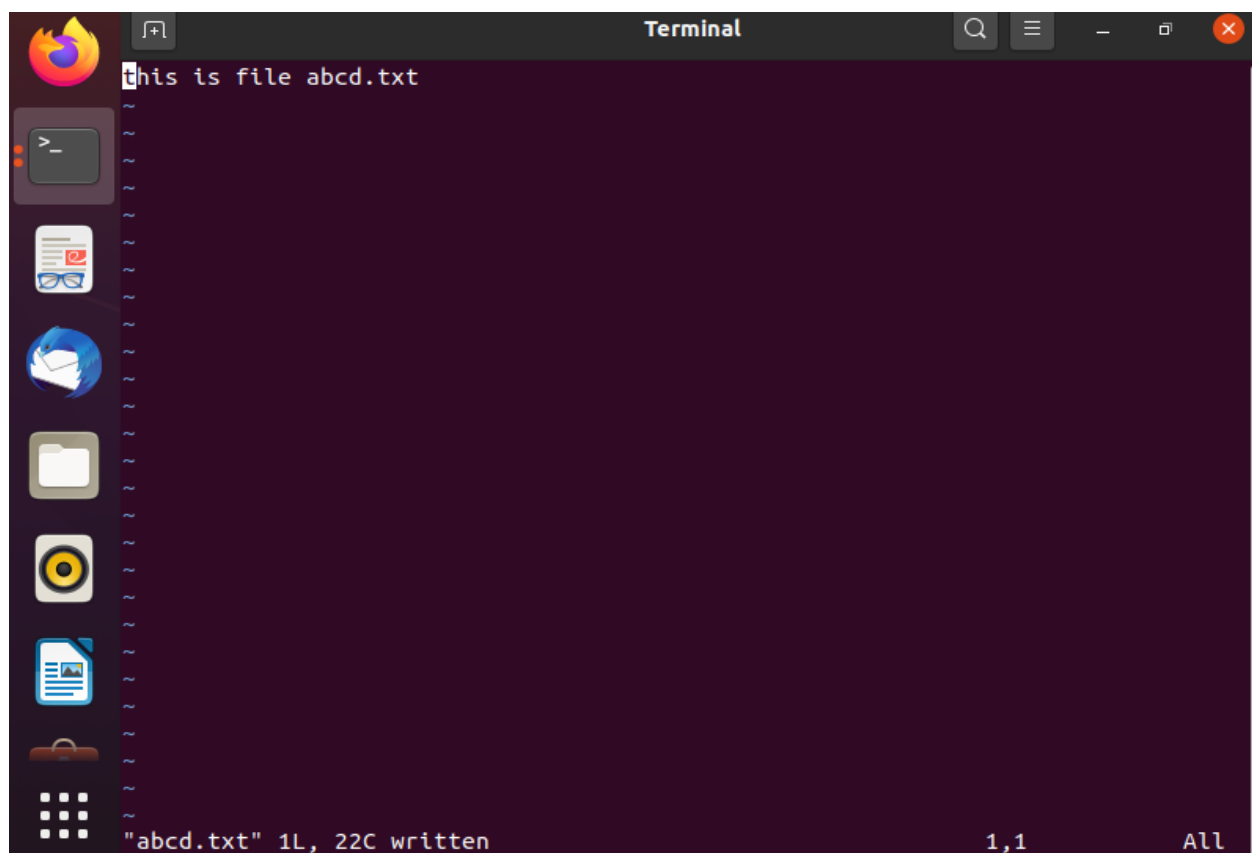


```
dung-19520477@dung19520477-VirtualBox:~/LAB3$ vim bai04.c
dung-19520477@dung19520477-VirtualBox:~/LAB3$ gcc bai04.c -o bai04
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ./bai04

Welcome to IT007, I am Mai Dung_19520477!
# Option "-e" is deprecated and might be removed in a later version of gnome-terminal.
# Use "--" to terminate the options and put the command line to execute after it.
```

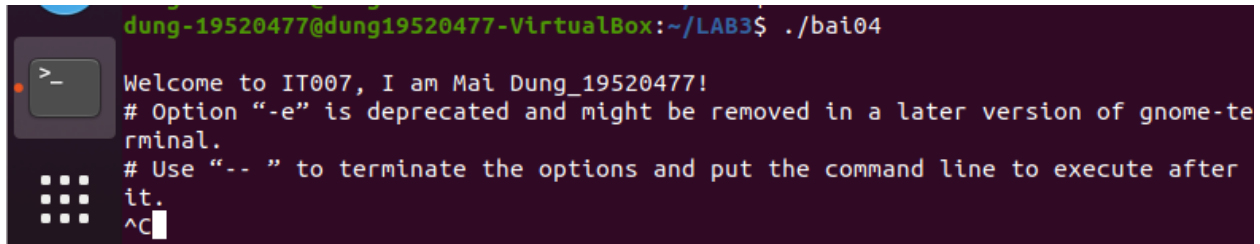
Hình 4.b _ In ra mà hình dòng chữ “Welcome to IT007, I am Mai Dung_19520477!” và mở một cửa sổ terminal mới.

a+b. Như ta thấy trong source code (hình 4.a), hàm main() thực thi lệnh printf() để in dòng chữ như trên. Lệnh tiếp theo là system(“gnome-terminal -e ‘vim abcd.txt’”), lệnh này tạo tiến trình mở một cửa sổ terminal mới, và mở file abcd.txt bằng vim editor bằng lệnh vim abcd.txt.



Hình 4.c _ terminal mới được tạo và mở file abcd.txt bằng vim

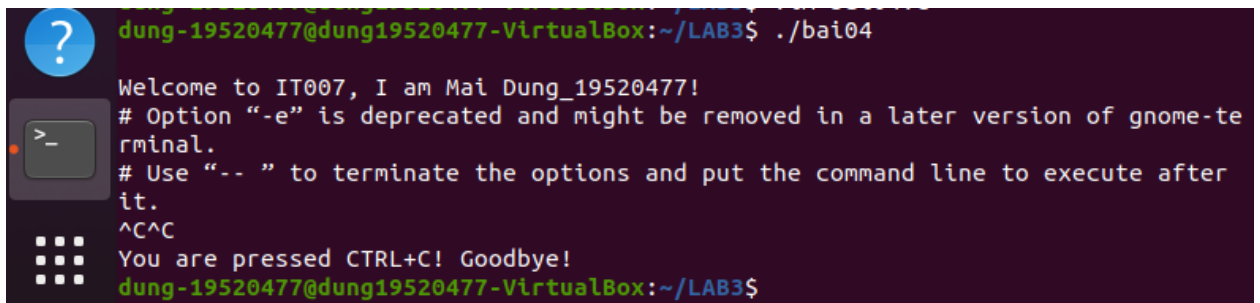
c. Để tắt vim editor (dừng chương trình) khi người dùng nhấn Ctrl+C, ta dùng thư viện signal.h, cài đặt hàm kill_vimeditor() và dùng hàm main() gọi tới nó để gửi tín hiệu kill vim:



```
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ./bai04
Welcome to IT007, I am Mai Dung_19520477!
# Option "-e" is deprecated and might be removed in a later version of gnome-terminal.
# Use "--" to terminate the options and put the command line to execute after it.
^C
```

Hình 4.d _ khi nhấn CTRL+C, ta thấy cửa sổ terminal có chạy vim abcd.txt đã tắt.

d. Để in ra dòng chữ: “You are pressed CTRL+C! Goodbye!” khi nhấn Ctrl+C, cài đặt hàm on_signint(), hàm main() viết lệnh gọi tới nó:



```
dung-19520477@dung19520477-VirtualBox:~/LAB3$ ./bai04
Welcome to IT007, I am Mai Dung_19520477!
# Option "-e" is deprecated and might be removed in a later version of gnome-terminal.
# Use "--" to terminate the options and put the command line to execute after it.
^C^C
You are pressed CTRL+C! Goodbye!
dung-19520477@dung19520477-VirtualBox:~/LAB3$
```

Hình 4.e _ nhấn CTRL+C lần nữa, in ra dòng chữ: “You are pressed CTRL+C! Goodbye!”

...