Name: Ngô Phúc Danh

ID: 21521924

Class: IT007.N11.1

OPERATING SYSTEM
LAB 3'S REPORT

SUMMARY

Self-scrores: 8

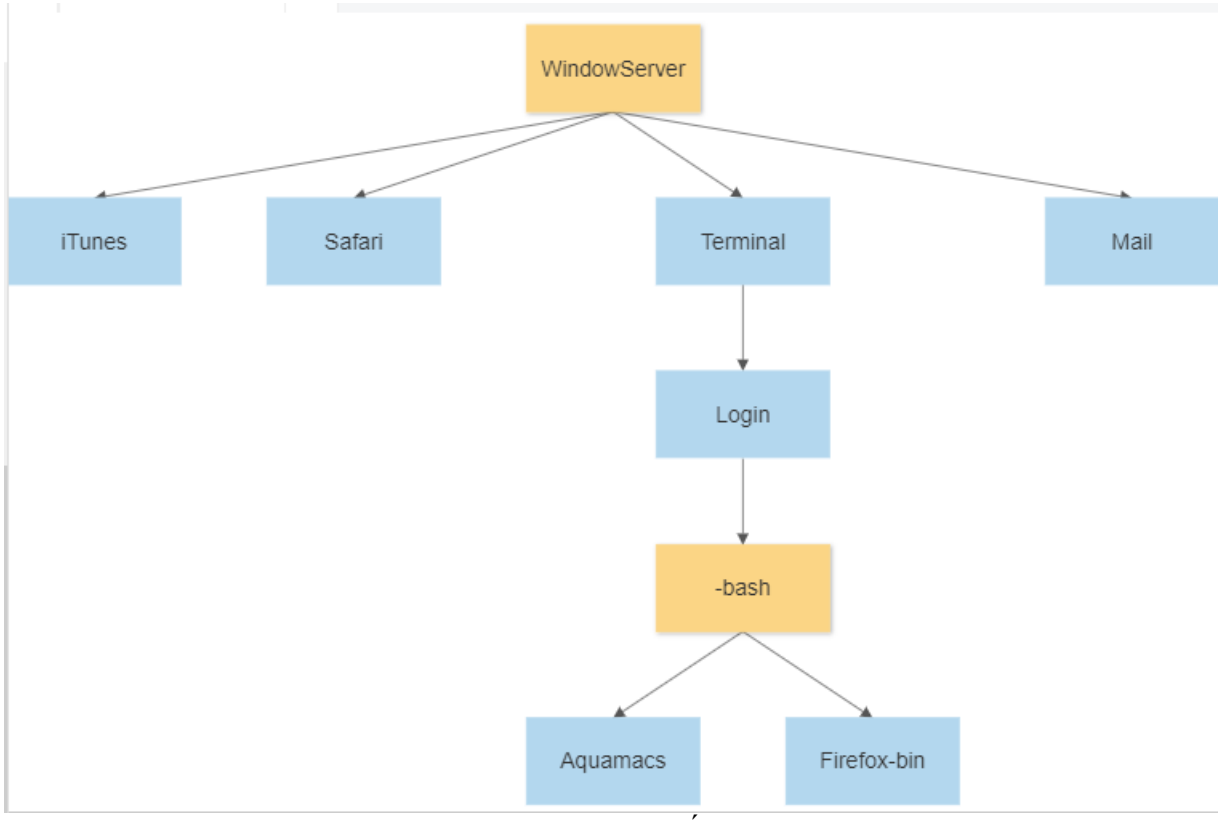
Note: Export file to **PDF and name the file by following format:*
LAB03 – 21521924.pdf

Câu 1: Mối quan hệ cha con giữa các tiến trình.

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

Hình 1.1: Tiến trình.



Hình 1.2: Cây quan hệ tiến trình trên.

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

- Hướng thực hiện: chúng ta sử dụng lệnh có cú pháp là ps -f<>, khi đó dữ liệu sẽ được hiển thị, trong đó ta thấy có PPID thể hiện thông tin PID của tiến trình cha.

```
phucdanh-21521924@phucdanh21521924-virtual-machine:~$ ps -f 5345
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
phucdan+    5345    5327  0 14:58 pts/0    Ss     0:00 bash
phucdanh-21521924@phucdanh21521924-virtual-machine:~$ ps -f 5327
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
phucdan+    5327    3455  0 14:58 ?        Ssl    0:00 /usr/libexec/gnome-termin
phucdanh-21521924@phucdanh21521924-virtual-machine:~$
```

Hình 1.3: Kết quả trả về khi sử dụng câu lệnh ps -f <>.

- Giải thích: 5345 là PID của tiến trình cần tìm tiến trình cha nên sau khi sử dụng lệnh ps -f 5345 ta được kết quả trả về có PPID là 5327. Sau đó ta muốn hiển thị thông tin của tiến trình cha ta dùng lệnh ps -f 5327.

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

- Hướng thực hiện: chúng ta tiến hành sử dụng câu lệnh có cú pháp là pstree -sp <PID của tiến trình cần tìm tiến trình cha>. Sau khi thực hiện lệnh trên thì một cây quan hệ sẽ được trả về màn hình hiển thị.

```
phucdanh-21521924@phucdanh21521924-virtual-machine:/$ pstree -sp 5345
systemd(1)───systemd(3455)───gnome-terminal-(5327)───bash(5345)───pstree(6592)
phucdanh-21521924@phucdanh21521924-virtual-machine:/$ ps -f
```

Hình 1.4: Kết quả trả về khi sử dụng lệnh pstree -sp.

- Giải thích: Khi thực hiện lệnh trên, ta dễ dàng thấy được tiến trình cha của tiến trình có PID 5345 là tiến trình có PID là 5327.

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

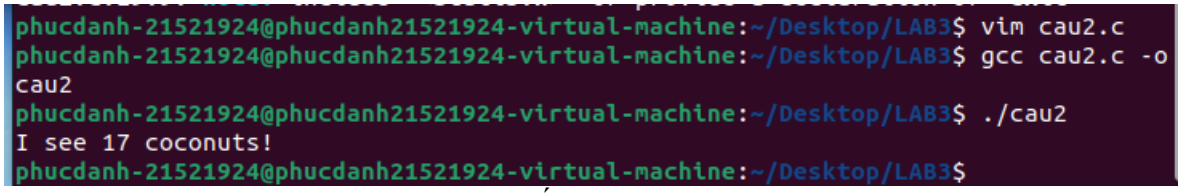
phucdanh-21521924@phucdanh21521924-virtual-machine: ~/...

```c
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
        int 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);
}
```

"cau2.c" 23L, 283B 4,19

Hình 2.1: Mã nguồn chương trình.

- Kết quả chương trình:



Hình 2.1: Kết quả chương trình.

- Giải thích: ở chương trình trên, sau khi thực hiện lệnh fork() chương trình tạo ra một tiến trình con sao chép source code của tiến trình cha. Ta gọi P0 là tiến trình cha còn P1 là tiến trình con. Tại tiến trình cha P(0) chỉ 1 lệnh printf được thực hiện do đó dòng chữ "I see 17 coconuts" được xuất ra màn hình. Còn tiến trình con P(1) không thể xuất ra màn hình dòng printf do có exit(0) làm tiến trình dừng lại. Vì vậy, ta thu được kết quả như trên.

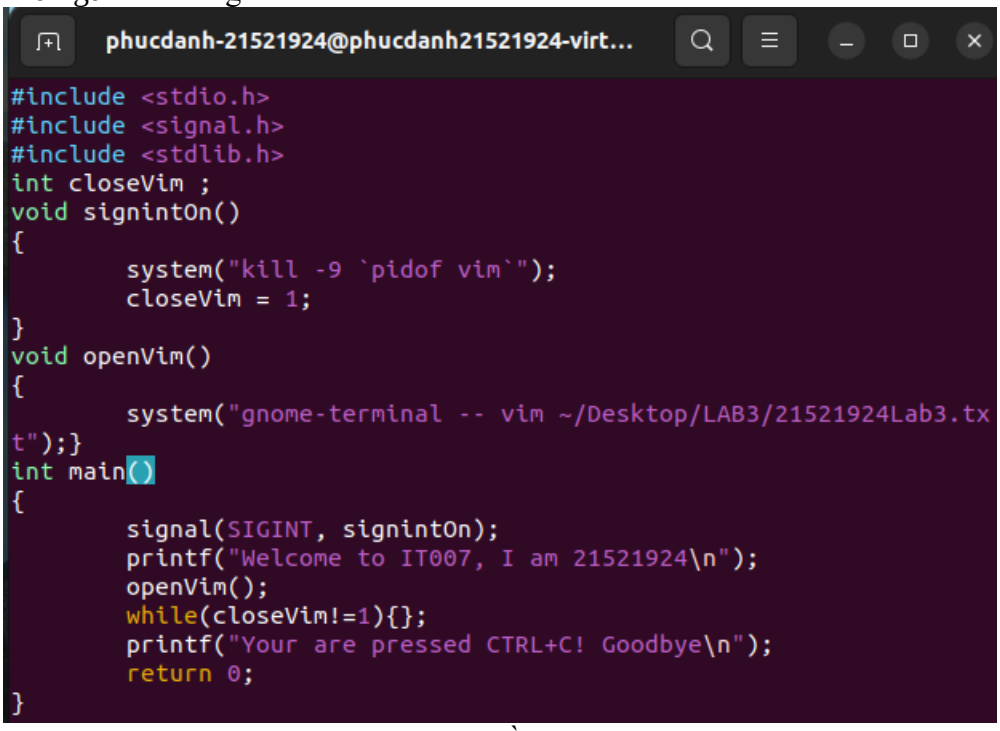
Câu 3: Trong phần thực hành, các ví dụ chỉ sử dụng thuộc tính mặc định của pthread, hãy tìm hiểu POSIX thread và trình bày tất cả các hàm được sử dụng để làm thay đổi thuộc tính của pthread, sau đó viết các chương trình minh họa tác động của các thuộc tính này và chú thích đầy đủ cách sử dụng hàm này trong chương trình. (Gợi ý các hàm liên quan đến thuộc tính của pthread đều bắt đầu bởi: pthread_attr_*).

STT	Tên	Cú pháp	Chức năng
1	pthread_attr_setdetachstate	Int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);	đặt trạng thái tách của tiểu trình được tham chiếu bởi attr thành giá trị được chỉ định trong detachstate
2	pthread_attr_init	int pthread_attr_init(pthread_attr_t *attr);	khởi tạo chuỗi của thuộc tính được trỏ đến bởi attr bằng thuộc tính mặc định của các giá trị
3	pthread_attr_setstacksize	int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);	đặt kích thước stack của các thuộc tính được gọi bởi atrr bằng kích thước được chỉ định trong stacksize
4	pthread_attr_setschedparam	int pthread_attr_setschedparam(pthread_attr_t *restrict attr, const struct sched_param *restrict param);	đặt lịch cho các thuộc tính của tiểu trình được trỏ tới bởi attr thành các thuộc tính được trỏ đến bởi restrict param
5	pthread_attr_setinheritsched	int pthread_attr_setinheritsched(pthread_attr_t *attr, int inheritsched);	đặt thuộc tính lập lịch kế thừa của các tiểu trình được trỏ đến bởi attr giá trị của inheritsched
6	Pthread_attr_setshedpolicy	int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);	đặt lịch cho policy của tiểu trình được trỏ đến trong attr bằng giá trị của tham số policy
7	Pthread_attr_setaffinity_np	int pthread_attr_setaffinity_np (pthread_attr_t * attr , size_t cpusetsize , const cpu_set_t * cpuset);	đặt thuộc tính CPU affinity mask của tiểu trình được trỏ đến bởi attr thành giá trị trong cpuset

8	Pthread_attr_setscope	int pthread_attr_setscope(pthread_attr_t *attr, int scope);	đặt giá trị contention scope của các tiểu trình được trỏ bởi attr giá trị của scope
9	Pthread_attr_setguardsize	int pthread_attr_setguardsize(pthread_attr_t *attrsize_t" guardsize);	đặt kích thước của guard của tiểu trình được trỏ bởi attr bằng giá trị trong guardsize
10	Pthread_attr_setstack	int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr, size_t stacksize)	đặt địa chỉ của các stack của các tiểu trình trỏ đến bởi attr bằng giá trị của trong stackaddr, đồng thời đặt kích thước của các ngăn xếp bằng stacksize

Câu 4: Viết chương trình làm các công việc sau theo thứ tự:
a. In ra dòng chữ: "Welcome to IT007, I am <…>!"
b. Mở tệp abcd.txt bằng vim editor.
c. Tắt vim editor khi người dùng nhấn CRTL+C.
d. Khi người dùng nhấn CTRL+C thì in ra dòng chữ: "You are pressed CTRL+C! Goodbye!".

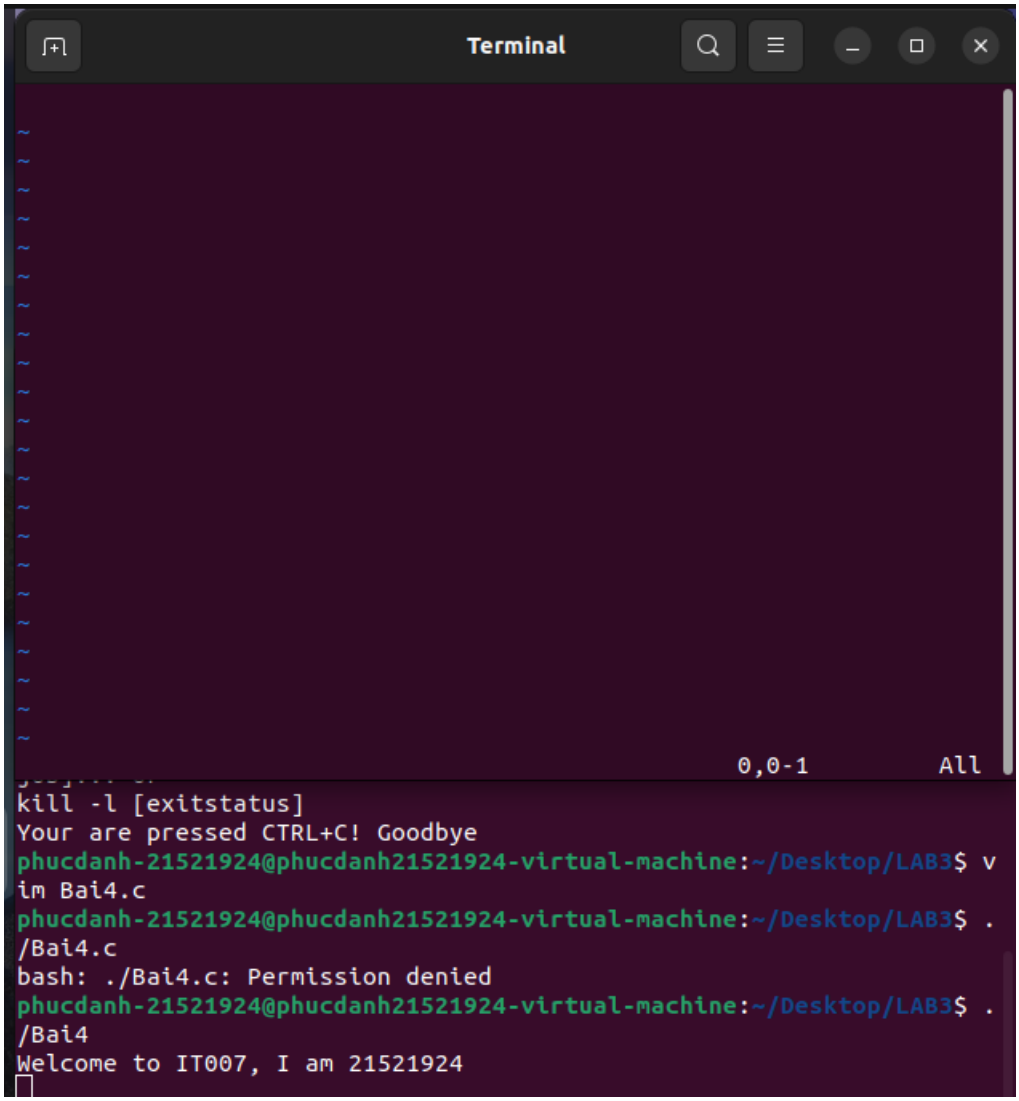
- Mã nguồn chương trình:



Hình 4.1: Mã nguồn chương trình.

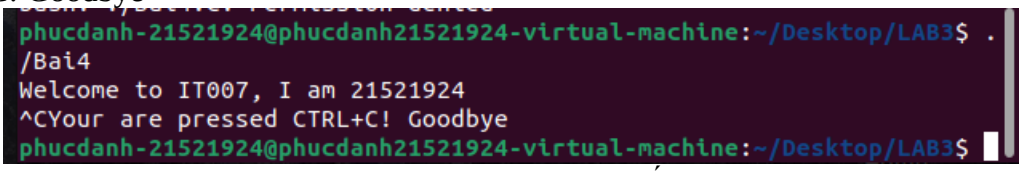
- Hướng thực hiện: ta thực hiện hàm openVim() để mở giao diện Vim trong 1 terminal mới cho việc làm câu c, còn hàm signintOn() dùng để tắt Vim editor khi người dùng nhấn tổ hợp ctrl+c.
- Các bước chương trình thực hiện:

Bước 1. In ra màn hình dòng chữ Welcome to IT007, I am 21521924 sau đó mở file 21521924Lab3.txt trong 1 terminal mới.



Hình 4.2: Chương trình chạy in ra dòng chữ và mở 1 terminal mới.

Bước 2: Khi người dùng nhấn Ctrl+c thì Vim đóng và in ra màn hình dòng chữ You are pressed CTRL+C! Goodbye



Hình 4.3: In ra dòng chữ khi nhấn Ctrl+c.

----------HẾT----------