

Name: Phạm Đức Thế²

ID: 19522253

Class: IT007.M14

OPERATING SYSTEM LAB 3 REPORT

SUMMARY

| Task | | Status | Page |
|-------------|------|------------|------|
| Section 3.5 | Ex 1 | Hoàn thành | 2 |
| | Ex 2 | Hoàn thành | 4 |
| | Ex 3 | Hoàn thành | 5 |
| | Ex 4 | Hoàn thành | 11 |
| | | | |
| | | | |
| | | | |
| ... | | | |
| | | | |

Self-scores: 10/10

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

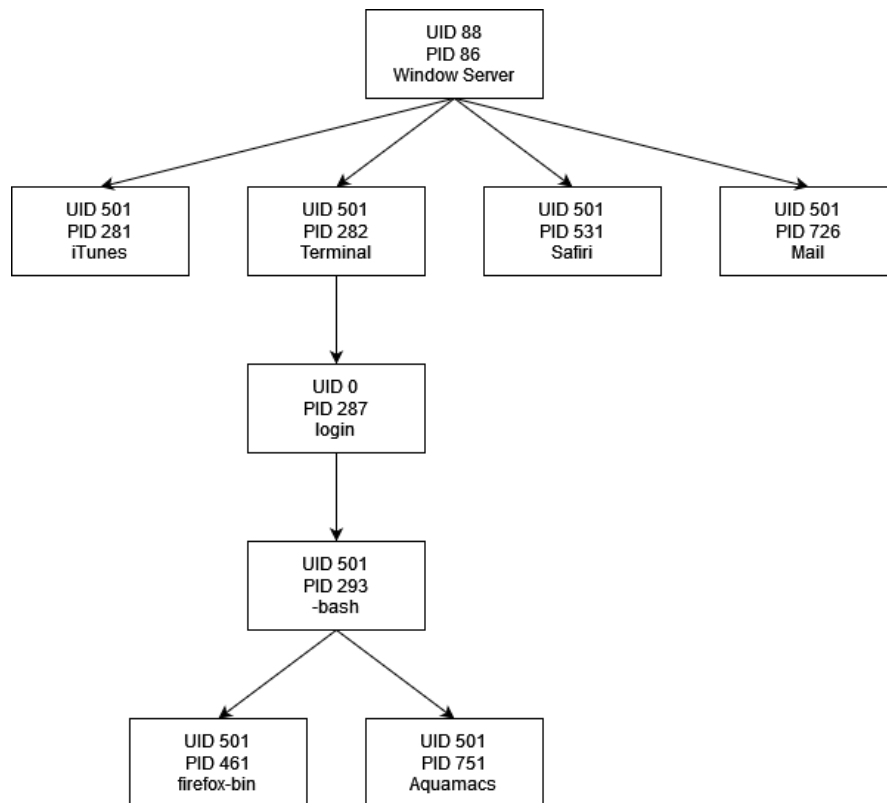
Section 3.5

1. Task name 1: Môi 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 |

📁 Vẽ cây quan hệ parent-child của các tiến trình:



Hình 1: Cây tiến trình

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

- Giống như `top`, `ps` cũng là lệnh giúp hiển thị chi tiết của tiến trình, trong đó có PID là ID của tiến trình và PPID là PID của tiến trình cha của tiến trình đó.

- Để sử dụng lệnh `ps` để tìm tiến trình cha của một tiến trình dựa vào PID của nó ta dùng lệnh: **`ps -f`**

```
the_19522253@the-19522253-VirtualBox:~/LAB03$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
the_195+      1828      1823  0   21:13 pts/0        00:00:00 bash
the_195+      1990      1828  0   21:14 pts/0        00:00:00 ps -f
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 2: Kết quả khi chạy lệnh `ps -f`

- Trong hình, ta có thể thấy rằng tiến trình `bash` có PID là 1828 là tiến trình cha của tiến trình `ps -f` nhờ lệnh `ps` trong qua PID và PPID của chúng.

- Ta có thể sử dụng lệnh `ps -fp [pidlist]` để tìm PPID của tiến trình qua PID của tiến trình đó

```
the_19522253@the-19522253-VirtualBox:~/LAB03$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
the_195+      1828      1823  0   21:13 pts/0        00:00:00 bash
the_195+      1990      1828  0   21:14 pts/0        00:00:00 ps -f
the_19522253@the-19522253-VirtualBox:~/LAB03$ ps -fp 1828
UID          PID     PPID  C  STIME TTY          TIME CMD
the_195+      1828      1823  0   21:13 pts/0        00:00:00 bash
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 3: Kết quả chạy lệnh `ps -fp 1828`

- 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.

- Lệnh `pstree` giúp hiển thị các tiến trình dưới dạng sơ đồ cây. Để sử dụng lệnh này để tìm tiến trình cha của một tiến trình dựa vào PID của nó ta dùng lệnh `pstree -sg [pidlist]`

```
the_19522253@the-19522253-VirtualBox:~/LAB03$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
the_195+      1828      1823  0   21:13 pts/0        00:00:00 bash
the_195+      1990      1828  0   21:14 pts/0        00:00:00 ps -f
the_19522253@the-19522253-VirtualBox:~/LAB03$ ps -fp 1828
UID          PID     PPID  C  STIME TTY          TIME CMD
the_195+      1828      1823  0   21:13 pts/0        00:00:00 bash
the_19522253@the-19522253-VirtualBox:~/LAB03$ pstree -sg 1828
systemd(1)---systemd(1228)---gnome-terminal-(1823)---bash(1828)---pstree(2169)
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

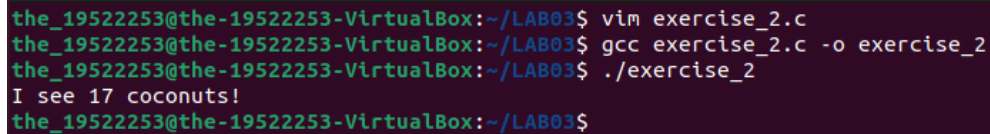
Hình 4: Kết quả chạy khi chạy lệnh `pstree -sg 1828`

- Qua kết quả trả về của lệnh `pstree` ta có thể thấy được tiến trình cha của `bash` (PID 1828) là tiến trình `gnome-terminal-`(PID 1823).

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

```
/*#####  
# University of Information Technology      #  
# IT007 Operating System                  #  
# <Your name>, <your Student ID>         #  
# File: exercise_2.c                     #  
#####*/  
  
#include<stdio.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);  
}
```

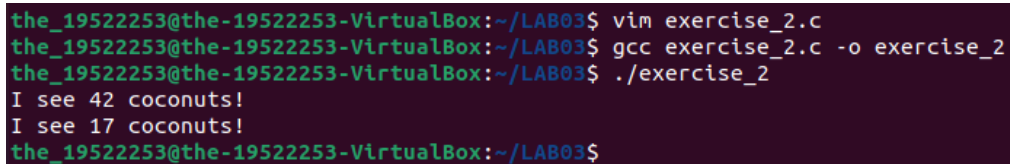
- Chạy lại chương trình:



```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_2.c  
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_2.c -o exercise_2  
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_2  
I see 17 coconuts!  
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 5: Kết quả chạy chương trình

- Vì tiến trình con có dòng lệnh `exit(0)`; khi tiến trình con chạy giá trị của `num_coconut` đã thay đổi thành 42, nhưng vì dòng lệnh trên nên dòng lệnh in kết quả ra màn hình không thực hiện được, nếu xóa dòng `exit(0)`; này đi ta sẽ có kết quả như sau:



```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_2.c  
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_2.c -o exercise_2  
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_2  
I see 42 coconuts!  
I see 17 coconuts!  
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 6: Kết quả chạy chương trình sau khi xóa dòng lệnh `exit(0)`; trong tiến trình con.

3. **Task name 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_*)

| Hàm | Chức năng |
|------------------------------|---|
| Pthread_attr_init | Khởi tạo giá trị mặc định cho đối tượng thuộc tính |
| Pthread_attr_destroy | Xóa bộ nhớ được cấp phát trong quá trình khởi tạo |
| Pthread_attr_getschedparam | Trả về các tham số lịch trình (Scheduling Parameter) được xác định bởi pthread_attr_setschedparam () |
| Pthread_attr_getschedpolicy | Đề xuất scheduling policy của thread |
| Pthread_attr_getdetachstate | Lấy truy xuất trạng thái khởi tạo của thread, có thể thể là riêng lẻ hoặc kết hợp |
| Pthread_attr_getinheritsched | Trả về chính sách lịch trình (scheduling policy) được set bởi pthread_attr_setinheritsched (). |
| Pthread_attr_getscope | Truy xuất phạm vi của thread |
| Pthread_attr_setdetachstate | Sử dụng lại ID và tài nguyên của thread khi nó bị ngắt mà không phải chờ nếu thread có thuộc tính riêng lẻ. |
| Pthread_attr_setguardsize | Set kích thước của khu vực an toàn của thread |
| Pthread_attr_setstackaddr | Set địa chỉ stack của thread |
| Pthread_attr_setstacksize | Set kích thước stack của thread |
| Pthread_attr_getguardsize | Lấy kích thước của khu vực an toàn của thread |
| Pthread_attr_getstackaddr | Trả về địa chỉ stack của thread được set bởi pthread_attr_setstackaddr () |
| Pthread_attr_getstacksize | Trả về kích thước stack của thread được set bởi pthread_attr_setstacksize (). |

Pthread_attr_init

```
int pthread_attr_init(pthread_attr_t *tattr);
#include <pthread.h>

pthread_attr_t tattr;
```

```
int ret;
```

```
/* khởi tạo giá trị mặc định cho thuộc tính */
```

```
ret = pthread_attr_init(&tattr);
```

Pthread_attr_destroy

```
int pthread_attr_destroy(pthread_attr_t *tattr);
```

```
#include <pthread.h>
```

```
pthread_attr_t tattr;
```

```
int ret;
```

```
/* Xóa thuộc tính */
```

```
ret = pthread_attr_destroy(&tattr);
```

Pthread_attr_getschedparam

```
int pthread_attr_getschedparam(pthread_attr_t *tattr,  
    const struct sched_param *param);
```

```
#include <pthread.h>
```

```
pthread_attr_t attr;
```

```
struct sched_param param;
```

```
int ret;
```

```
/* Lấy tham số lịch trình đang tồn tại trong thread */
```

```
ret = pthread_attr_getschedparam (&tattr, &param);
```

Pthread_attr_getschedpolicy

```
int pthread_attr_getschedpolicy(pthread_attr_t *tattr, int *policy);
```

```
#include <pthread.h>

pthread_attr_t tattr;
int policy;
int ret;

/* Lấy chính sách lịch trình của thread */
ret = pthread_attr_getschedpolicy (&tattr, &policy);
```

Pthread_attr_getdetachstate

```
int pthread_attr_getdetachstate(const pthread_attr_t *tattr,
    int *detachstate;
#include <pthread.h>

pthread_attr_t tattr;
int detachstate;
int ret;

/* Lấy trạng thái khởi tạo của thread */
ret = pthread_attr_getdetachstate (&tattr, &detachstate);
```

Pthread_attr_getinheritsched

```
int pthread_attr_getinheritsched(pthread_attr_t *tattr, int *inherit);
#include <pthread.h>

pthread_attr_t tattr;
int inherit;
int ret;
```

```
/* Lấy chính sách lịch trình và độ ưu tiên của thread được khởi tạo */  
ret = pthread_attr_getinheritsched (&tattr, &inherit);
```

Pthread_attr_getscope

```
int pthread_attr_getscope(pthread_attr_t *tattr, int *scope);  
#include <pthread.h>
```

```
pthread_attr_t tattr;  
int scope;  
int ret;
```

```
/* Lấy phạm vi của thread */  
ret = pthread_attr_getscope(&tattr, &scope);
```

Pthread_attr_setdetachstate

```
int pthread_attr_setdetachstate(pthread_attr_t *tattr,int detachstate);  
#include <pthread.h>
```

```
pthread_attr_t tattr;  
int ret;
```

```
/* Set trạng thái riêng lẻ cho thread */  
ret =  
pthread_attr_setdetachstate(&tattr,PTHREAD_CREATE_DETACHED);
```

Pthread_attr_setguardsize

```
#include <pthread.h>
```

```
int pthread_attr_setguardsize(pthread_attr_t *attr, size_t guardsize);
```


Pthread_attr_setstackaddr

```
int pthread_attr_setstackaddr(pthread_attr_t *tattr,void *stackaddr);  
#include <pthread.h>
```

```
pthread_attr_t tattr;  
void *base;  
int ret;
```

```
base = (void *) malloc(PTHREAD_STACK_MIN + 0x4000);
```

```
/* Đặt địa chỉ mới cho thread */  
ret = pthread_attr_setstackaddr(&tattr, base);
```

Pthread_attr_setstacksize

```
int pthread_attr_setstacksize(pthread_attr_t *tattr, int size);  
#include <pthread.h>
```

```
pthread_attr_t tattr;  
int size;  
int ret;
```

```
size = (PTHREAD_STACK_MIN + 0x4000);
```

```
/* Đặt kích thước mới */  
ret = pthread_attr_setstacksize(&tattr, size);
```

Pthread_attr_getguardsize

```
#include <pthread.h>
```

```
int pthread_attr_getguardsize(const pthread_attr_t *attr, size_t
*guardsize);
```

Pthread_attr_getstackaddr

```
int pthread_attr_getstackaddr(pthread_attr_t *tattr,void **stackaddr);
#include <pthread.h>
```

```
pthread_attr_t tattr;
void *base;
int ret;
```

```
/* Lấy một địa chỉ mới */
ret = pthread_attr_getstackaddr (&tattr, *base);
```

Pthread_attr_getstacksize

```
int pthread_attr_getstacksize(pthread_attr_t *tattr, size_t *size);
#include <pthread.h>
```

```
pthread_attr_t tattr;
int size;
int ret;
```

```
/* Lấy kích thước stack */
ret = pthread_attr_getstacksize(&tattr, &size);
```

4. Task name 4: Viết chương trình làm các công việc sau theo thứ tự:

- a. In ra dòng chữ: “Welcome to IT007, I am <your_Student_ID>!”

```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_4.c
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_4.c -o exercise_4
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_4

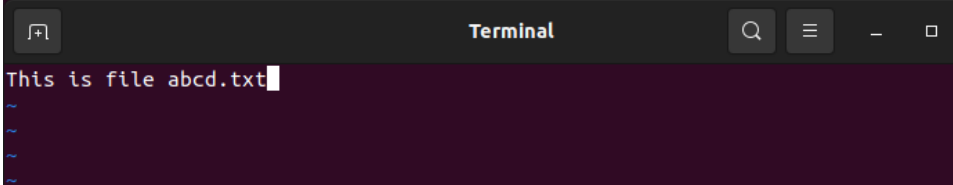
Welcome to IT007, I am 19522253!
```

Hình 7: Kết quả câu a

- b. Mở tệp abcd.txt bằng vim editor

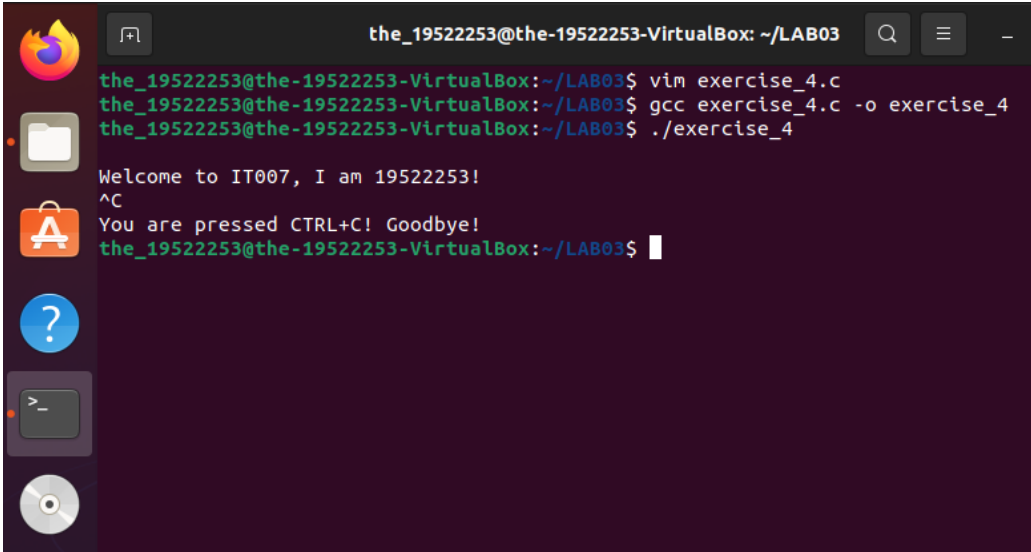
```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_4.c
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_4.c -o exercise_4
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_4

Welcome to IT007, I am 19522253!
```



Hình 8: Kết quả câu b

- c. Tắt vim editor khi người dùng nhấn CTRL+C



```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_4.c
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_4.c -o exercise_4
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_4

Welcome to IT007, I am 19522253!
^C
You are pressed CTRL+C! Goodbye!
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 9: Kết quả câu c

- d. Khi người dùng nhấn CTRL+C thì in ra dòng chữ: “You are pressed CTRL+C! Goodbye!”

```
the_19522253@the-19522253-VirtualBox:~/LAB03$ vim exercise_4.c
the_19522253@the-19522253-VirtualBox:~/LAB03$ gcc exercise_4.c -o exercise_4
the_19522253@the-19522253-VirtualBox:~/LAB03$ ./exercise_4

Welcome to IT007, I am 19522253!
^C
You are pressed CTRL+C! Goodbye!
the_19522253@the-19522253-VirtualBox:~/LAB03$
```

Hình 10: Kết quả câu d

Chương trình:

```
/*#####
# University of Information Technology
# IT007.M14 Operating System
# Pham Duc The, 19522253
# File: exercise_4.c
#####*/

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

#define _XOPEN_SOURCE 600

int loop = 1;

void sig_job(){
    system("kill -9 `pidof vim`");
    printf("\nYou are pressed CTRL+C! Goodbye!\n");
    loop = 0;
}

int main(){
    printf("\nWelcome to IT007, I am 19522253!\n");
    system("gnome-terminal -- vim abcd.txt");
    signal(SIGINT, sig_job);
    while(loop){};
    exit(0);
}
```

Hình 11: Code chương trình