|  |
| --- |
| Name: Phạm Đức Thể  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-scrores: 10/10**

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

# Section 3.5

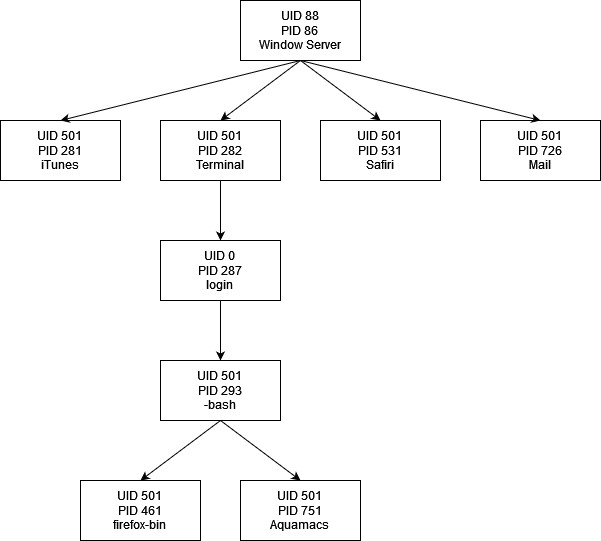
## Task name 1: Mối quan hệ cha-con giữa các tiến trình

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

Table

Description automatically generated

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



Hình 1: Cây tiến trình

1. 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**

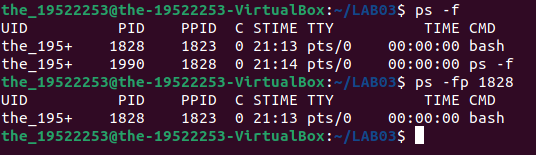
Text

Description automatically generated

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.

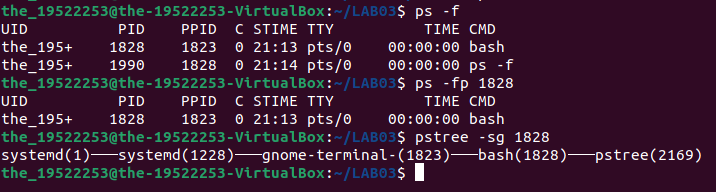
- Ta có thể sử dụng lệnh ps -fp [pidlist] để tìm PPID của tiến trình qua PID của tiến trình đó



Hình 3: Kết quả chạy lệnh ps -fp 1828

1. 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]



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

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

## 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 \*/

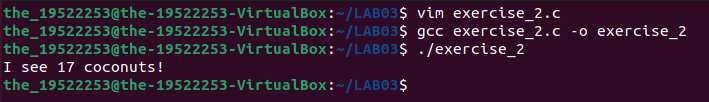
}

printf("I see %d coconuts!\n", num\_coconuts);

exit(0);

}

- Chạy lại chương trình:



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:

Text

Description automatically generated

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

## 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 | Để 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);

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

1. In ra dòng chữ: “Welcome to IT007, I am <your\_Student\_ID>!”

Text

Description automatically generated

Hình 7: Kết quả câu a

1. Mở tệp abcd.txt bằng vim editor

Text

Description automatically generated

Hình 8: Kết quả câu b

1. Tắt vim editor khi người dùng nhấn CRTL+C

Text

Description automatically generated

Hình 9: Kết quả câu c

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

Text

Description automatically generated

Hình 10: Kết quả câu d

Chương trình:

Text

Description automatically generated

Hình 11: Code chương trình