|  |
| --- |
| 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-scrores: 9**

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

# Section 1.5

## Task name 1

1. 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:

parent————child(1)————subchild (1)

| |--subchild (2)

|

|\_ child(2)

* A picture containing text

  Description automatically generatedThực hiện:

1. Cách sử dụng lệnh ps để tìm tiến trình cha của một 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.

HìnhText

Description automatically generated 1.b \_ lệnh ps tìm tiến trình cha dựa vào PID đã biết

1. 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 dã biết>***. Lệnh này sẽ hiển thị một cây tiến trình tương tự như cấu trúc trên phần 1.a mổ 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 dã biết>.***
* A picture containing text, screen, screenshot

  Description automatically generatedVí dụ: trong hình 1.c, ta tìm tiến trình cha của bash có PID là 2221, dùng lệnh *pstree -sp 2221* để hiển thị cây tiến trình, tiến trình cha là tiến trình nằm trước nó: gnome-terminal-(2210).

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

## Task name 2

* Text

  Description automatically generatedChương trình đã cho:

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

* Text

  Description automatically generatedKết quả khi thực thi chương trình:

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.

## 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 mặt định của tất cả/ một thuộc tính mà được chọn | 0 |
| pthread\_attr\_destroy() | Loại bỏ bộ nhớ cấp phát trong quá trình khởi tạo. Đối tượng thuộc tính trở nên không hợp lệ. | 0 |
| pthread\_attr\_setdetachstate() | Kết hợp/ tách rời (em chưa rõ) | 0 |
| pthread\_attr\_setguardsize ()  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. |
| pthread\_attr\_setscope() | Create (tạo) a bound thread (PTHREAD\_SCOPE\_SYSTEM) or an unbound thread (PTHREAD\_SCOPE\_PROCESS). | 0 |
| pthread\_attr\_setschedpolicy() | 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 |
| pthread\_attr\_setinheritsched() | Đặt chính sách lập lịch kế thừa | 0 |
| pthread\_attr\_setschedparam() | Đặt các thông số lập lịch. Các tham số được xác định trong cấu trúc param, chỉ priority được hỗ trợ. Luồng mới được tạo chạy với priority. | 0  The value of param is NULL or tattr is invalid. |
| pthread\_attr\_setstacksize() | 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 PTHREAD\_STACK\_MIN hoặc vượt quá giới hạn do hệ thống áp đặt hoặc tattr không hợp lệ. |
| pthread\_attr\_setstackaddr() | Thuộc tính stackaddr xác định cơ sở ngăn xếp của luồng. Nếu được đặt non-null (default NULL) thì hệ thống sẽ khởi tạo ngắn xếp tại địa chỉ đó. | 0  Giá trị của base hoặc tattr không chính xác |

## Task name 4

* Text

  Description automatically generatedChương trình thực hiện các công việc theo yêu cầu:

HìnhText

Description automatically generated 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***

Text

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

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 screenshot of a computer

Description automatically generated with medium confidencea+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

**Text

Description automatically generatedc.** 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:

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

Text

Description automatically generated**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:

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

…