KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 3.4 – Đồng bộ kết thúc các tiến trình

Cập nhật lần cuối: 22/6/202

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Tạo tiến trình	Ch3: Process Creation	Sử dụng image Ubuntu 14 / 16
Huỷ tiến trình	Ch3: Process Creation	

2021.1.26	Thêm thông tin bài tập 3.

Yêu cầu sinh viên: Hiểu lý thuyết liên quan. Biết biên dịch các đoạn code mẫu và thực thi. Biết cách sử dụng lời gọi fork() tạo ra tiến trình con. Áp dụng lập trình đa nhiệm bằng cách tạo ra các tiến trình con. Áp dụng tạo ra cây tiến trình và quá trình kết thúc đồng bộ.

Đánh giá sinh viên: Hỏi đáp các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các "ví dụ" và bài tập cuối hướng dẫn.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

Programming Problems of Chapter 3.

[2] Greg Gagne , [2019], GitHub OS-BOOK OSC10e, Westminster College, United States

Access https://github.com/greggagne/osc10e in September 2019.

[3] Zombie and Orphan Processes in C, *geeksforgeeks.org*

?	- Tiến trình mồ côi là gì? - Những rủi ro từ tiến trình mồ côi là gì? - Tiến trình xác sống là gì? - Những thiệt hại gây ra bởi tiến trình xác sống là gì?	LAB 1 Ch3 p.151	Slide Ch3.

Yêu cầu 1: tạo ra một tiến trình mồ côi (Orphaned process), biên dịch, thực thi và quan sát hệ thống bằng System Monitor hay lời gọi ps.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_3/task4_1.c***, sau đó biên dịch và thực thi rồi quan sát. Chúng ta có thể chạy một chương trình ở chế độ "background" bằng cách thêm "&" vào sau lời gọi.

```
> ./a.out &
```

Khi đó a.out thực thi bên dưới và chúng ta có thể gõ lệnh hay chạy một tiến trình khác ở bên trên terminal.

Cách khác nữa là: Khi một tiến trình đang thực thi, chúng ta có thể đưa tiến trình này vào "background" thông qua việc gửi tín hiệu SIGTSTP bằng cách nhấn Ctrl + Z, khi đó chúng ta có thể gõ lệnh hoặc chạy một tiến trình khác.

Yêu cầu 2: tạo ra một tiến trình xác sống (Zombie process), biên dịch, thực thi và quan sát hệ thống bằng System Monitor hay lời gọi ps.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_3/task4_1.c***, sau đó biên dịch và thực thi rồi quan sát. Hãy chú ý tiến trình con có ID là bao nhiêu, bộ nhớ được cấp phát ra sao, sự khác thường với các tiến trình khác là gì?

Ngoài cách sử dụng lời gọi wait(), chúng ta có thể bỏ qua tín hiệu SIGCHLD.

Khi một tiến trình con bị chấm dứt, tín hiệu SIGCHLD tương ứng sẽ được gửi đến tiến trình cha, nếu chúng ta thực hiện lời gọi "(SIGCHLD, SIG_IGN)", thì sau đó tín hiệu SIGCHLD sẽ bị hệ thống bỏ qua và chỉ mục của tiến trình con bị xóa khỏi bảng tiến trình . Do đó, không có zombie nào được tạo ra. Tuy nhiên, trong trường hợp này, tiến trình cha không thể biết về tình trạng thoát ra của tiến trình con.

Yêu cầu 3: Lời gọi wait() yêu cầu tiến trình cha chờ đợi đến khi có một tín hiệu từ tiến trình con. Trong trường hợp tiến trình cha chỉ muốn "kiểm tra trạng thái" của tiến trình con, waitpid() cần được sử dụng. Viết chương trình để khi thực thi sẽ tạo ra một tiến trình con cho phép người dùng nhập mã PIN, tiến trình cha chờ đợi trong một khoảng thời gian nhất định và kết thúc chương trình nếu người dùng không nhập gì cả. Ngược lại lấy giá trị trả về từ tiến trình con.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_3/task4_3.c***, sau đó biên dịch và thực thi rồi quan sát.

- Dòng 27 là thao tác nhập, và nếu người dùng không nhập, chương trình sẽ không thể tiếp túc. Chúng ta có thể giải quyết bằng cách giao việc nhập cho một tiến trình con, tiến trình cha sẽ kiểm soát thời gian cho phép nhập. (Thực tế tại các máy ATM, hay các ứng dụng Ví điện tử, Ngân hàng trực tuyến, … thì việc nhập sẽ kết thúc nếu không có thao tác nhập sau một vài phút.)

- Dòng 31 là thời lượng chờ, kết hợp với dòng 39 thì được hiểu thời gian chờ là 5 giây.

- Dòng 33 là kiểm tra trạng thái, với chọn lựa "WNOHANG" thì tiến trình cha sẽ lấy về được trạng thái mà không chờ sự thay đổi trạng thái của tiến trình con.

- Dòng 34 cho thấy hành động của dòng 33 thất bại, khả năng là tiến trình con đã kết thúc (người dùng có nhập mã PIN trong vòng 5 giây).

- Dòng 41 là tiến trình cha kết thúc tiến trình con, pid là ID của tiến trình con, sau khi dòng 40 cho thấy người dùng đã không nhập mã PIN sau 5 giây.

```
27      scanf("%d", & n);
31      int timer = 5;
33        wpid = waitpid(pid, & status, WNOHANG);
34        if (wpid == -1) {
38        timer--;
39        sleep(1);
40        if (timer == 0) {
41          kill(pid, SIGKILL);
```

Đọc thêm: Lời gọi wait() và waitpid() khác nhau như thế nào?

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
```

Tất cả các lời gọi hệ thống này được sử dụng để chờ sự thay đổi trạng thái trong một tiến trình con (của tiến trình đang thực hiện lời gọi) và lấy thông tin về tiến trình con đó. Sự thay đổi trạng thái có thể là: tiến trình con chấm dứt thực thi; tiến trình con bị chặn lại bởi một tín hiệu; hoặc tiến trình con được phục hồi bởi một tín hiệu. Trong trường hợp một tiến trình bị chấm dứt thực thi, việc gọi lệnh wait() cho phép hệ thống giải phóng các tài nguyên liên quan đến tiến trình con; nếu lệnh wait() không được thực hiện, thì tiến trình con bị chấm dứt vẫn ở trạng thái "zombie".

Nếu một tiến trình con đã thay đổi trạng thái, những lời gọi hệ thống này trả về kết quả ngay lập tức. Mặt khác, chúng chặn cho đến khi một tiến trình con thay đổi trạng thái hoặc bộ xử lý tín hiệu làm gián đoạn lời gọi.

Cuộc gọi hệ thống wait () tạm dừng thực thi tiến trình gọi cho đến khi một trong các con của nó kết thúc. Lời gọi wait(&status) tương đương với:

```
wait(&status); //the same with
waitpid(-1, &status, 0);
```

Lời gọi hệ thống waitpid() tạm dừng thực thi tiến trình gọi cho đến khi một tiến trình con - được chỉ định bởi đối số pid - đã thay đổi trạng thái. Mặc định, waitpid() chỉ đợi tiến trình con bị chấm dứt, tuy nhiên có thể dùng đối số tùy chọn để thay đổi, như được mô tả dưới đây.

- Trả về ID của tiến trình con thay đổi trạng thái. Trả về -1 nếu thất bại.

  Đối số pid của hàm:

  - <-1: đợi bất kỳ tiến trình nào có groupID bằng với group của |pid|.
  - -1: đợi bất kỳ tiến trình nào.
  - 0: đợi bất kỳ tiến trình con nào có GroupID bằng với Group ID của tiến trình gọi.
  - >0: Đợi tiến trình có ID bằng với pid.

Bài tập lập trình

1. Việc sử dụng các lệnh sleep(i) là giải pháp rất tệ khi chương trình phải dừng lại và thường chỉ dùng trong cho các quan sát của sinh viên. Hãy loại bỏ các lệnh sleep() trong các yêu cầu đã thực hiện bên trên, và sử dụng đồng bộ kết thúc wait(NULL) để đảm bảo không có tiến trình mồ côi hay xác sống.
2. Loại bỏ tiến trình xác sống trong Yêu cầu 2 bằng cách sử dụng SIGCHLD thay vì lời gọi wait(). Lời gọi này đặt trong mã tiến trình cha.

   ```
   signal(SIGCHLD,SIG_IGN);
   ```

3. Programming Problems 3.19 in book [1]

   Viết chương trình xác định thời gian cần thiết để thực thi một lệnh, lệnh này truyền vào qua phần đối số như minh hoạ dưới đây. Hàm gettimeofday() có thể dùng để tính thời gian. Thân hàm chính nên được thiết kế như sau:

   - Lấy thời gian hiện tại (bắt đầu)
   - fork() để tạo tiến trình con và tiến trình con dùng system() để thực thi lệnh truyền vào.
   - Khi tiến trình con kết thúc, tiến trình cha đợi bằng lời gọi wait().
   - Lấy thời gian hiện tại (kết thúc) và tính ra thời gian đã trôi qua (kết thúc – bắt đầu).

Ví dụ để đo thời gian lệnh ls thực thi

```
>./time ls
 time.c
 time
 Elapsed time: 0.25422
```