**MÔN: HỆ ĐIỀU HÀNH**

**TIẾN TRÌNH**

1. **Giới thiệu**

Tiến trình là một môi trường thực hiện, bao gồm một phân đoạn lệnh và một phân đoạn dữ liệu. Cần phân biệt với khái niệm chương trình chỉ gồm tập hợp lệnh.

Trên hệ điều hành Linux, tiến trình được nhận biết thông qua số hiệu của tiến trình, gọi là *pid*. Cũng như đối với user, nó có thể nằm trong nhóm. Vì thế để phân biệt ta nhận biết qua số hiệu nhóm gọi là *pgrp*. Một số hàm của C cho phép lấy được những thông số này:

*int getpid()* : trả về giá trị int là pid của tiến trình hiện tại

*int getppid()* : trả về giá trị int là pid của tiến trình cha của tiến trình hiện tại

*int getpgrp()* : trả về giá trị int là số hiệu của nhóm tiến trình

*int setpgrp()* : trả về giá trị int là số hiệu nhóm tiến trình mới tạo ra

Ví dụ:

Lệnh: printf("Toi la tien trinh %d thuoc nhom %d",getpid(),getgrp());

Kết quả sẽ là: Toi la tien trinh 235 thuoc nhom 231

1. **Tạo một tiến trình - Lệnh fork**

*int fork()* tạo ra một tiến trình con. Giá trị trả lại là 0 cho tiến trình con và số hiệu pid cho tiến trình cha. Giá trị sẽ là -1 nếu không tạo được tiến trình mới. Theo nguyên tắc cơ bản của hệ thống, tiến trình con và cha sẽ có cùng đoạn mã. Đoạn dữ liệu của tiến trình mới là một bản sao chép chính xác đoạn dữ liệu của tiến trình cha. Tuy nhiên tiến trình con vẫn khác tiến trình cha ở pid, thời gian xử lý, ...

Ví dụ 1:

#include <stdio.h>

int main()

{

int pid;

pid = fork(); // tách tiến trình

printf(“pid = %d”,pid);

switch (pid)

{

case -1: printf( "Khong the tao tien trinh moi" ); break;

case 0: printf( "Day la tien trinh con" );

// mã lệnh dành cho tiến trình con đặt ở đây

break;

default: printf( "Day la tien trinh cha" );

// mã lệnh dành cho tiến trình cha đặt ở đây

break;

}

return 0;

}

Ví dụ 2: Sử dụng hàm fork(), fork\_demo.c

#include <stdio.h>

int main()

{

int pid, n;

char \* message;

printf( "Bat dau.\n" );

pid = fork();

switch (pid)

{

case -1: printf( "Khong the tao tien trinh moi" ); break;

case 0: message = "Day la tien trinh con";

n = 0;

for ( ; n < 3; n++ )

{

printf( "%s %d \n", message );

sleep(1);

}

break;

default: message = "Day la tien trinh cha";

n = 0;

for ( ; n < 4; n++ )

{

printf( "%s %d \n", message );

sleep(1);

}

break;

}

return 0;

}

Biên dịch và thực thi chương trình này, chúng ta sẽ thấy rằng cả 2 tiến trình hoạt động đồng thời và in ra kết quả đan xen nhau. Nếu muốn xem sự liên quan về PID và PPID của cả 2 tiến trình cha và con khi lệnh fork() phát sinh, chúng ta có thể thực hiện chương trình như sau: *./fork\_demo & ps -af*

1. **Dừng một tiến trình**

Lệnh kill của Shell có thể dùng để chấm dứt hoạt động của một tiến trình. Ví dụ như khi muốn dừng tiến trình có pid là 234 ta dùng lệnh: *kill 234*

C cũng có lệnh kill như sau:

*int kill(pid, sig);*

*int pid*: là dấu hiệu nhận biết của một tiến trình.

*int sig*: hằng tín hiệu giao tiếp tiến trình.

1. **Giao tiếp giữa các tiến trình**

Việc giao tiếp giữa các tiến trình được thực hiện thông qua các tín hiệu chuẩn của hệ thống. Tín hiệu là một sự ngắt quãng logic được gửi đến các tiến trình bởi hệ thống để thông báo cho chúng về những sự việc không bình thường trong môi trường hoạt động của chúng (như lỗi bộ nhớ, lỗi vào ra). Nó cũng cho phép các tiến trình liên lạc với nhau.

Dưới đây là một số tín hiệu thường gặp:

* SIGHUP: Tín hiệu này được phát đến các tiến trình vào lúc cuối khi mà nó tự ngắt. Nó cũng được phát đến mọi tiến trình có tiến trình chính tự ngắt.
* SIGINT: Tín hiệu này được phát đến các tiến trình khi ta ra lệnh ngắt.
* SIGQUIT: Tương tự như trên khi ta gõ vào Ctrl + D.
* SIGILL: Lệnh không hợp lệ, tín hiệu được phát ra khi phát hiện 1 lệnh không đúng ở cấp độ vật lý (ví dụ như 1 tiến trình thực hiện một lệnh mà máy tính không có lệnh này).
* SIGTRAP: Tín hiệu được phát ra sau mỗi lệnh trong trường hợp tiến trình có sử dụng lệnh ptrace().
* SIGIOT: Bẫy được phát khi có các vấn đề về vật lý.
* SIGEMT: Bẫy của lệnh phát, được phát ra khi có lỗi vật lý trong khi thực hiện.
* SIGFPE: Được phát ra khi có lỗi về tính toán như một số có dấu phẩy nối có định dạng không hợp lý. Gần như luôn chỉ ra lỗi khi lập trình.
* SIGKILL: Trang bị để kết thúc tiến trình. Không thể bỏ qua hoặc cắt tín hiệu này.
* SIGBUS: Được phát khi gặp lỗi trên bus.
* SYSGEGV: Được phát ra khi gặp lỗi trên phân đoạn sự truy cập dữ liệu bên ngoài phân đoạn dữ liệu được cấp phát cho tiến trình.
* SIGSYS: Đối số không đúng cho hệ thống gọi.
* SIGPIPE: Viết trên một ống dẫn không mở để đọc.
* SIGALRM: Phát ra khi đồng hồ của một tiến trình ngừng lại. Đồng hồ được hoạt động bằng lệnh alrm().
* SIGTERM: Được phát ra khi một tiến trình kết thúc bình thường. Cũng có thể dùng để dừng 1 hệ thống để kết thúc tất cả các tiến trình hoạt động.

1. **Liên lạc giữa hai tiến trình**

Từ một chương trình đơn giản dưới đây sử dụng các lệnh phát và nhận tín hiệu, sau đó giúp liên lạc giữa hai tiến trình. Nội dung của ví dụ là sự liên lạc giữa một tiến trình cha và một tiến trình con thông qua các tín hiệu đã được trình bày phần trước.

#include "stdio.h"

#include "signal.h"

void satthu()

{

printf("Tien trinh (pid = %d) bi loai bo !!!\n",getpid());

kill(getpid(), SIGINT);

}

void chumuu()

{

signal(SIGUSR1, satthu);

printf("Hinh thanh tien trinh moi (pid = %d). Nhung chuan bi loai bo tien trinh nay!!\n",getpid());

}

int main()

{

int pid;

if ((pid = fork())==0) *//con*

chumuu();

else *//cha*

{

sleep(3);

printf("Chap nhan !! Tien trinh (pid = %d) se bi loai bo.\n",pid);

kill(pid, SIGUSR1);

}

return 0;

}

Trong ví dụ trên, tiến trình con có sử dụng hàm *signal(SIGUSR1, satthu)*. Hàm này có tác dụng mỗi khi tiến trình con nhận được tín hiệu SIGUSR1 thì hàm *satthu()* sẽ được thực thi.

Như vậy ở ví dụ trên một tiến trình con đã được tạo ra nhưng nó lại không muốn tiếp tục tồn tại. Do vậy sau khi tạm dừng lại *sleep(3)*, tiến trình cha đã gởi đến cho tiến trình con một tín hiệu là SIGUSR1 bằng lệnh:

*kill(pid, SIGUSR1);*

Ở tiến trình con, tín hiệu SIGUSR1 đã được gán với hàm *satthu*(). Hàm này tạo ra một thông báo báo hiệu tiến trình này sắp chết rồi tự gởi đến chính mình (tiến trình con) tín hiệu SIGINT, tín hiệu ngắt tiến trình. Và tiến trình con đã chết. *kill(getpid(), SIGINT);*