# Tiến trình con, liên lạc giữa tiến trình bằng đường ống

## Tạo tiến trình con với lệnh fork

Một tiến trình có thể tạo ra tiến trình con bằng lời gọi tới hàm fork(). Khi đó tiến trình con sẽ được sao lặp từ tiến trình cha. Chúng ta có thể biết đang ở trong tiến trình nào dựa vào giá trị trả lại của hàm fork()

* =0: đang ở trong tiến trình con
* >0: đang ở trong tiến trình cha và pid tiến trình con là giá trị trả lại
* <0: lỗi, không tạo được tiến trình con

Ví dụ: Biên dịch và chạy chương trình myfork.c dưới đây

// gcc –o myfork myfork.c

#include<stdio.h>

#include<unistd.h>

int main() {

int pid = fork();

if (pid < 0) { print("Cannot create the child process"); exit(-1);}

if (pid==0) print("hello from the child");

else print("hello from the parent. Child’s pid=%d", pid);

}

**Bài số 1**:Tính cả tiến trình cha, có bao nhiêu tiến trình được tạo ra dưới đây? Điền vào google form câu trả lời.

// fork1.c

int main() {

fork();

fork();

fork();

return 0;

}

**Bài số 2**: Nội dung được in ra màn hình sẽ gồm bao nhiêu chữ big, medium, small? Điền vào google form câu trả lời.

// fork2.c

void main() {

int pid;

pid = fork();

printf(pid);

if (pid==0) { // it's the child process

fork(); printf("medium\n");

fork();

fork(); printf("small\n");

}

else printf("big\n");

}

**Bài số 3**: Cho đoạn mã chương trình dưới đây. Dòng A sẽ in ra kết quả gì? Điền vào google form câu trả lời.

// fork3.c

int value = 5;

int main() {

pid\_t pid;

pid = fork();

if (pid == 0){ /\* child process \*/

value += 15;

return 0;

}

else if (pid > 0){ /\* parent process \*/

wait(NULL); /\* wait for the child process to terminate \*/

printf("PARENT: value = %d\n",value); // dòng A

return 0;

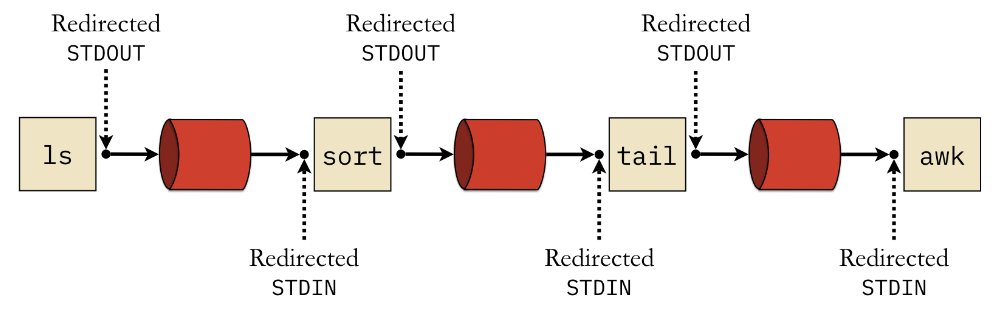
}

}

## Pipes

Đường ống (pipe) là cơ chế hàng đợi cho phép nối tệp đầu ra của một tiến trình với tệp đầu vào của một tiến trình khác. Ví dụ

$ ls -l | sort -n -k 5 | tail -n 1 | awk '{print $NF}'



Lệnh ls –l liệt kê chi tiết danh sách các tệp trong thư mục hiện tại. Danh sách này được cấp làm đầu vào cho lệnh sort –n –k 5 để sắp xếp theo cột thứ 5 là kích thước tệp từ nhỏ đến lớn. Tiếp theo lệnh tail –n 1 cắt lấy dòng cuối, ứng với tệp có kích thước lớn nhất. Lệnh awk in ra cột cuối cùng, là tên của tệp có kích thước lớn nhất (NF – number of fields).

Lưu ý rằng đường ống là 1 chiều và chỉ dùng kết nối giữa các tiến trình có quan hệ cha/con. Trong ví dụ trên các lệnh được thực thi như là các tiến trình con của hệ vỏ dòng lệnh.

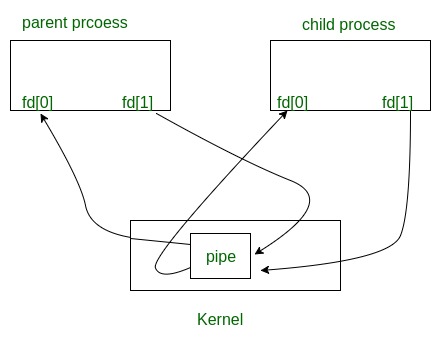
**Hàm pipe()**

Tạo một đường ống với 2 tham số thẻ mô tả tệp (file descriptor)

**int pipe(int fd[2]); // return <0 in case of failure**

Trong đó fd[0] là thẻ tệp cho phía đọc, fd[1] là thẻ tệp cho phía ghi. Sau đó thực hiện lệnh fork() để tạo tiến trình con. Nếu tiến trình cha gửi dữ liệu tới tiến trình con qua đường ống thì

* Tiến trình cha cần đóng lại tệp fd[0], dùng fd[1] để ghi dữ liệu ra
* Tiến trình con cần đóng lại tệp fd[1], dùng fd[0] để đọc dữ liệu vào



Trường hợp cần trao đổi 2 chiều giữa tiến trình cha và con thì phải tạo 2 đường ống riêng biệt.

Ví dụ tiến trình cha gửi 1 thông điệp tới tiến trình con

**// Yêu cầu: nhập đoạn mã dưới đây, đặt tên là anonpipe1.c và chạy thử**

**int main(){**

**char \*msg = “Hello child\n”, buf[255];**

**int fd[2], pid;**

**if (pipe(fd) < 0) exit(-1);**

**if ((pid = fork()) > 0) { // parent**

**close(fd[0]);**

**write(fd[1], msg, strlen(msg));**

**wait(NULL); // wait for the child to exit**

**}**

**else { //child**

**close(fd[1]); memset(buf, 0, 255);**

**read(fd[0], buf, 254);**

**printf("%s\n", buf);**

**}**

**return 0;**

**}**

**Bài số 4**: Sửa chương trình trên, sử dụng 2 đường ống để tiến trình cha gửi 1 xâu ký tự, rồi tiến trình con chuyển xâu đó sang chữ hoa và gửi trả lại tiến trình cha. Tiến trình cha in ra kết quả xâu nhận lại được. Hàm toupper(ch) trong thư viện ctype.h chuyển ký tự ch sang ký tự chữ hoa. Cho trước khung chương trình trong tệp anonpipe2.c. Hãy hoàn tất và nộp qua google form.