Bài thực hành số 3

Lập trình pipe

Ghi chú:

- Sinh viên nộp bài tại trang web: `www.cse.hcmut.edu.vn/portal`
- File nộp bài đặt tên là: *`lab3.tar.bz2`* (sử dụng lệnh *make pack*)
- Hạn chót nộp bài: *`11:59pm 05/08/2010`*
- SV có thể chỉnh sửa source code mẫu nếu thấy cần thiết
- Mọi gian lận sẽ nhận điểm KHÔNG nếu bị phát hiện

1 Giới thiệu

1.1 Mục tiêu

Viết một chương trình C trên Linux để chạy các lệnh trên Linux sử dụng kĩ thuật lập trình pipe.

1.2 Kiến thức cần biết

- Lập trình C trên Linux
- Lập trình multiprocess
- Lập trình pipe
- Makefile

1.3 Mô tả chương trình

Đầu vào (Input) của chương trình:

- Các lệnh cần chạy

Đầu ra (Output) của chương trình:

- Kết quả chạy được của các lệnh đó.
- Lỗi khi chạy chương trình.

Chương trình khi chạy sinh ra 2 process: process cha và process con. Process cha có nhiệm vụ đọc các lệnh từ file input, sau đó gửi các lệnh này sang cho process con bằng cách sử dụng pipe. Process con sau khi nhận được lệnh sẽ chạy lệnh đó và xuất kết quả vào file output, xuất lỗi vào file error. Trước khi chạy một lệnh nào đó, process con ghi thông tin số thứ tự của dòng lệnh vào file output và file error. Process con sử dụng kỹ thuật chuyển hướng xuất nhập để xuất kết quả vào file output và file error.

1.4 Ý tưởng hiện thực chương trình

- Tạo 2 pipe để thực hiện việc truyền dữ liệu giữa process con và process cha. Một pipe phục vụ việc truyền dòng lệnh process cha đọc được sang process con. Pipe còn lại để process con thông báo cho process cha biết nó đã thực thi xong dòng lệnh hiện tại và yêu cầu process cha truyền dòng lệnh tiếp theo.

- Khi hai process được tạo, process cha đọc các lệnh trong file input, và truyền lệnh này sang process con. Process con nhận được lệnh này và thực thi nó.
- Trước khi thực thi lệnh, process con chuyển hướng xuất của việc thực thi lệnh sang file output và chuyển hướng lỗi của việc thực thi lệnh sang file errror.
- Trước khi xuất kết quả và lỗi vào file output và file error, process con ghi thông tin dòng lệnh vào hai file này.
- Khi thực thi xong một lệnh, process con gửi thông báo sang process cha để yêu cầu process cha gửi sang lệnh tiếp theo.

1.5 Một số kỹ thuật lập trình

1.5.1 Xử lý thông số nhập vào từ chương trình

Một chương trình tốt thường cho phép người dùng thiết lập một vài thông số khi chạy chương trình, chẳng hạn khi thực hiện lệnh:

```
$ ls -R
```

người dùng đã truyền vào thông số `-R` để liệt kê các file và thư mục không chỉ trong thư mục hiện hành mà còn cả những thư mục con của thư mục hiện hành nếu có.

Sau đây là đoạn chương trình mẫu, sử dụng hàm `getopt()` để xử lý thông số -R ở trên:

```
int opt;
extern char *optarg;

while ((opt = getopt(argc, argv, "R")) != EOF) {
    switch (opt) {
        case 'R':
            // Option -R occurs
            // Proccess that option here
            break;
        default:
            // Other options
            break;
    }
}
```

1.5.2 Xử lý file cấu hình

File cấu hình thường ở dạng text và có cấu trúc (đơn giản). Ta thường sử dụng các hàm sau để xử lý những dạng file này:

- fopen: mở một file
- fgets: đọc một dòng trong file
- fclose: đóng file đã mở
- open: mở một file
- write: ghi một chuỗi các ký tự vào file
- close: đóng file đã mở

1.5.3 Chuyển hướng xuất nhập file

Khi một process được tạo trong hệ thống, mặc định process này được hệ thống cung cấp sẵn 3 file: stdin (0), stdout (1) và stderr (2) tương ứng với thiết bị nhập chuẩn (bản phím), thiết bị xuất chuẩn (monitor), và thiết bị

báo lỗi chuẩn (monitor). Do đó khi ta ghi bất kỳ thông tin nào vào file số 1 (stdout), điều này có nghĩa là thông tin đó sẽ được ghi vào thiết bị xuất chuẩn, tức là thông tin đó sẽ được in ra màn hình.

Để chuyển hướng xuất nhập file input, output và error, ta có thể sử dụng hàm `dup` hoặc `dup2` để trỏ các file descriptor tương ứng vào file mình cần chuyển hướng. Giả sử ta đang trỏ file descriptor số 1 vào file output.txt, khi đó bất kỳ thông tin nào ghi vào file số 1 cũng có nghĩa là ta đang ghi thông tin đó vào file output.txt. Vì vậy khi thực thi một lệnh, mặc định kết quả của nó sẽ được ghi vào file số 1, điều này có nghĩa là thông tin output của lệnh này đang ghi vào file output.txt.

2 Yêu cầu

Chương trình sau khi biên dịch có tên là `pipe`, hỗ trợ các thông số sau:

- `-h`: Hiển thị thông tin hướng dẫn sử dụng chương trình
- `-i` *filename*: Chọn file input. Đây là file chứa thông tin các lệnh cần chạy. Tên file input mặc định là *`input.txt`*
- `-o` *filename*: Chọn file output. Đây là file chứa kết quả của các lệnh. Tên file output mặc định là *`output.txt`*
- *`-e filename:`* Chọn file error. Đây là file chứa thông báo lỗi của các lệnh. Tên file error mặc định là *`error.txt`*

Cú pháp chạy chương trình `pipe`:

```
pipe [-h] [-i filename] [-o filename] [-e filename]
```

Một số lưu ý khi xử lý option:

- Nếu dòng lệnh nhập vào không đúng với cú pháp ở trên thì báo lỗi, thoát chương trình ngay lập tức.
- Độ ưu tiên của option -h là cao nhất, sau đó là các option để chạy chương trình. Điều này có nghĩa là nếu dòng lệnh có option -h và các option chạy chương trình thì ta chỉ quan tâm đến option -h, và xem như không có các option còn lại.

3 Định dạng file input, output, error

3.1 File input

Mỗi dòng trong file input là một lệnh cần chạy. Ví dụ:

```
cd /etc   // Lệnh 0
cd /etca  // Lệnh 1
pwd       // Lệnh 2
```

Với file input mẫu ở trên, chương trình xem như file input có 3 lệnh cần được thực thi. Lệnh `cd /etc`, lệnh `cd /etca` và lệnh `pwd`.

3.2 File output

- Kết quả của các lệnh trong file input sẽ được ghi vào file output.
- Nội dung của chúng phải được ghi theo đúng thứ tự của các lệnh trong file input.
- Trước mỗi kết quả có dòng chú thích ghi đây là kết quả của dòng lệnh nào trong file input (số thứ tự bắt đầu là 0)

- Giữa 2 kết quả của 2 dòng lệnh cách nhau bằng một dòng trống (không có khoảng trắng, dấu tab, ... ở dòng này)
- Nếu một lệnh không xuất quả, vẫn phải ghi dòng chú thích và cách với các kết quả khác bởi một dòng trống.
- Nếu lệnh phía trước làm thay đổi cấu trúc cây thư mục (mkdir, rm, rmdir, touch, ...) thì kết quả của nó sẽ ảnh hưởng đến lệnh phía sau, nếu không (ls, pwd, cd, ...) thì kết quả của nó không ảnh hưởng đến dòng lệnh phía sau.

```
#command 0                  // Dòng chú thích thông tin dòng lệnh
                            // Dòng trống
#command 1                  // Lệnh 1 không tạo output

#command 2
/home/ntnguyen/lab3/lab3    // Kết quả của lệnh 2
```

3.3 File error

- Lỗi của một lệnh khi chạy sẽ được ghi vào file error.
- Nội dung của chúng phải được ghi theo đúng thứ tự của các lệnh trong file input.
- Trước mỗi thông tin lỗi có dòng chú thích ghi đây là lỗi của dòng lệnh nào trong file input (số thứ tự bắt đầu là 0)
- Giữa 2 thông tin lỗi của 2 dòng lệnh cách nhau bằng một dòng trống (không có khoảng trắng, dấu tab, ... ở dòng này)
- Nếu thực thi lệnh không có lỗi, vẫn phải ghi dòng chú thích và cách với các thông tin lỗi khác bởi một dòng trống.

```
#command 0                  // Thông tin dòng lệnh
                            // Dòng trống
#command 1
cd: 1: can't cd to /etca    // Lỗi của lệnh 1

#command 2                  // Lệnh 2 không có lỗi
```