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

Hướng dẫn Lab 5.1 – Đường ống liên lạc

Nếu như các tiểu trình có thể liên lạc nội bộ bằng các biến số toàn cục, thì các tiến trình cha con lại rất khó khăn trong liên lạc do hoàn toàn không thể dùng phương pháp khai báo biến số toàn cục. Hệ điều hành cung cấp các cơ chế liên lạc bao gồm Vùng nhớ chia sẻ và Gửi thông điệp được giới thiệu trong LAB 5 này. Trước hết, chúng ta hãy tìm hiểu về Đường ống liên lạc.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Chia sẻ giữa các tiểu trình	Ch3: Process Ch3.3.7.4 Pipes	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18 Sử dụng Windows 10
Đường ống bình thường		
Liên lạc trong WIN32		

Yêu cầu sinh viên: Hiểu lý thuyết về liên lạc giữa các tiến trình. Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng cho các bài toán liên lạc giữa các tiến trình.

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 Yêu cầu trong buổi thực hành và Bài tập cuối hướng dẫn trong thời gian cho phép của giảng viên.

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.

Chia sẻ thông tin giữa các tiến trình.

Yêu cầu 1: Có thể dùng biến số toàn cục để chia sẻ thông tin giữa các tiến trình cha con không?

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_5/task1_1.c***, biên dịch và thực thi.

- Dòng 10 khai báo biến value toàn cục.

- Dòng 16 là sự cập nhật biến value thực hiện bởi tiến trình con.

```
10      int value = 5;
16          value += 15;
21          printf("PARENT: value = %d", value); /* LINE A */
```

- Giá trị value được in ra tại dòng 21 là bao nhiêu? Giải thích.

Yêu cầu 2: Có thể dùng biến số toàn cục để chia sẻ thông tin giữa các tiến trình và tiểu trình không?

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_5/task1_2.c***, biên dịch và thực thi.

- Dòng 10 khai báo biến value toàn cục.

- Dòng 21 là sự cập nhật biến value thực hiện bởi tiểu trình runner.

```
10      int value = 5;
21          value += 15;
16          printf("PARENT: value = %d", value); /* LINE A */
```

- Giá trị value được in ra tại dòng 16 là bao nhiêu? Giải thích.

- SV nhận xét gì về biến số chia sẻ **value** trong 2 yêu cầu nêu trên?

Đường ống bình thường (Ordinary pipe)

- Được sử dụng để truyền dữ liệu giữa các tiến trình theo cơ chế FIFO.

- Thường được sử dụng cục bộ giữa các tiến trình có quan hệ cha con.

Yêu cầu 3: Hoàn thiện chương trình mà khi thực thi tiến trình cha sẽ gửi một chuỗi kí tự, tiến trình con đón nhận rồi in ra màn hình.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_5/task1_3.c***, biên dịch và thực thi. Đây cũng là đoạn mã trong Hình 21, 22 sách [1].

- Dòng 10 là thư viện chưa thao tác trên Đường ống.

- Dòng 12 định nghĩa kích thước tối đa của chuỗi kí tự.

- Dòng 13 và 14 định nghĩa tên 2 đầu đường ống cho tiện sử dụng (thay vì mặc định là 0 và 1).

- Dòng 16 và 17 khai báo chuỗi gửi và chuỗi nhận.

- Dòng 18 và 21 là lời gọi tạo đường ống, nếu lời gọi thất bại, giá trị -1 được trả về.

- Dòng 35 đến 39 là thao tác GHI của tiến trình cha, lưu ý rằng đường ống cần được đóng đầu đọc bên kia lại (dòng 35), nếu không dữ liệu ghi sẽ "đi ra mất". Dòng 37 là ghi với 3 đối số: đầu ghi, chuỗi ghi, kích thước chuỗi ghi bao gồm kí tự kết thúc chuỗi. Cuối cùng đầu ghi cần đóng lại tại dòng 39.

- Dòng 43 đến 48 là thao tác ĐỌC của tiến trình con, tương tự, đường ống cần được đóng đầu ghi bên kia lại (dòng 43), nếu không dữ liệu đọc sẽ không tìm thấy kí hiệu "kết thúc chuỗi" và quá trình đọc không thể kết thúc. Dòng 45 là đọc với 3 đối số: đầu đọc, biến số chứa chuỗi đọc về, số kí tự đọc tối đa. Cuối cùng đầu đọc cũng cần đóng lại tại dòng 48.

```
10      #include <unistd.h>
12      #define BUFFER_SIZE 25
13      #define READ_END 0
14      #define WRITE_END 1

16        char write_msg[BUFFER_SIZE] = "Greetings";
17        char read_msg[BUFFER_SIZE];
18        int fd[2];
21        if (pipe(fd) == -1) {

            /* close the unused end of the pipe */
35          close(fd[READ_END]);
            /* write to the pipe */
37          write(fd[WRITE_END], write_msg, strlen(write_msg) +
        1);
            /* close the write end of the pipe */
39          close(fd[WRITE_END]);

            /* close the unused end of the pipe */
43          close(fd[WRITE_END]);
            /* read from the pipe */
45          read(fd[READ_END], read_msg, BUFFER_SIZE);
46          printf("read %s", read_msg);
            /* close the read end of the pipe */
48          close(fd[READ_END]);
```

Bài tập lập trình.

1. Tiến trình cha nhận vào các đối số thông qua lời gọi thực thi. Và lần lượt ghi vào đường ống. Tiến trình con nhận dữ liệu rồi in ra màn hình.

[3] Nói về gửi nhiều chuỗi kí tự cho các tiến trình con:

https://stackoverflow.com/questions/61838722/how-to-send-multiple-strings-from-parent-to-child-processes-trough-pipe

2. Tiến trình cha chuyển đối số đầu tiên (argv [1]) là một số nguyên lớn hơn 3 cho tiến trình con thông qua đường ống. Tiến trình con nhận, tính giá trị n! = 1 * 2 * … * n và ghi nó vào đường ống. Tiến trình cha nhận và xuất dữ liệu ra màn hình. Sử dụng đường ống bình thường (Ordinary Pipe).

```
>./baitap2A.out 4
4! = 24
```

Gợi ý: cần giải quyết 2 vấn đề chính:

- Đường ống chỉ chuyển kí tự, chuỗi kí tự nên tiến trình con cần chuyển nó sang số nguyên bởi hàm atoi().

- Có 2 quá trình gửi nhận cha gửi n >> con, rồi con gửi n! >> cha. Cần phải thiết lập 2 đường ống cho kênh liên lạc hai chiều này.

[4] Nói về giải pháp liên lạc 2 chiều: https://stackoverflow.com/questions/31888663/sending-characters-from-parent-to-child-process-and-returning-char-count-to-pare

3. Tiến trình cha đọc hai số nguyên và một thao tác +, -, *, / và chuyển tất cả cho tiến trình con. Tiến trình con tính toán kết quả và trả về cho tiến trình cha để in ra màn hình.

```
>./baitap3A.out 4 6 +
4 + 6 = 10
```