HỆ ĐIỀU HÀNH

TUẦN 10

GIAO TIẾP TIẾN TRÌNH – ĐƯỜNG ỐNG (PIPE) – NAMED PIPE

Hướng dẫn làm bài:

Khởi tạo pipe:

Thư viện:

```
#include <unistd.h>
```

Hàm ghi / đọc

```
ssize_t write(int fd, const void *buf, size_t count)
ssize_t read(int fd, const void *buf, size_t count)
```

Kết quả trả về:

- -1 thất bại
- Thành công: số byte ghi được hoặc đọc được.

Named Pipe

Mang ý nghĩa toàn cục

Tương tự như unnamed pipe

Được ghi nhận trên file system

Có thể sử dụng với các tiến trình không có quan hệ cha con

Có thể tạo từ dấu nhắc lệnh

Thư viện:

```
#include <sys/types.h>
#include<sys/stat.h>
```

Hàm khởi tạo

```
int mknod(const char *path, mode_t mode, dev_t dev)
```

- Đối số:

- path: đường dẫn đến pipe; kết hợp s_IFIFO với quyền khác.
- mode: quyền truy cập trên file
- dev: mặc định là 0

Kết quả trả về:

- Thất bại: -1;
- Thành công: 0

Hàm sử dụng

```
int mkfifo(const char *pathname, mode_t mode)
```

Đối số:

- path: đường dẫn đến pipe
- mode: quyền truy cập trên file

Kết quả trả về:

- Thất bại: -1
- Thành công: 0

Tiến trình cha ghi dữ liệu vào pipe
Tiến trình vào đọc, hiển thị dữ liệu, ghi dữ liệu phản hồi.
Tiến trình cha đọc dữ liệu phản hồi và hiển thị ra màn hình.
Ví dụ 2.1: Cho phép người dùng nhập vào 1 chuỗi, gửi chuỗi này qua tiến trình thứ 2. Tiến trình thứ 2 cho phép nhập lại 1 chuỗi khác, gửi ngược lại tiến trình đầu tiên.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <sys/wait.h>

#define FIFO1 "ff.1"
#define FIFO2 "ff.2"

#define PM 0666
extern int errno;

#define PIPE_BUF 4096

int main(int argc, char* argv[])
{
    char s1[PIPE_BUF], s2[PIPE_BUF];
    int childpid, readfd, writefd;
    //kiểm tra 2 file FIFO1 và FIFO2 có khởi tạo thành công?
    if((mknod(FIFO1, S_IFIFO | PM, 0)<0)&&(errno!=EEXIST))
    {
        printf("Fail to create FIFO 1. Aborted.\n");
        return -1;
    }
    if((mknod(FIFO2, S_IFIFO | PM, 0)<0)&&(errno!=EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2. Aborted.\n");
        return -1;
    }
    childpid=fork();
    if(childpid==0)  //child
    {
        //kiểm tra có thể mở 2 file FIFO1, FIFO2
        if((readfd=open(FIFO1, 0))<0)
            perror("Child cannot open readFIFO.\n");
        if((writefd=open(FIFO2, 1))<0)
            perror("Child cannot open writeFIFO.\n");
```

```
        //xử lý chương trình.
        read(readfd, s2, PIPE_BUF);
        printf("Child read from parent: %s\n", s2);
        printf("Enter response: ");
        gets(s1);
        write(writefd, s1, strlen(s1));
        close(readfd);
        close(writefd);
        return 1;
    }
    else
    if(childpid>0)   //parent
    {
        //Kiểm tra có mở thành công 2 file FIFO1, FIFO2.
        if((writefd=open(FIFO1, 1))<0)
            perror("Parent cannot open writeFIFO.\n");
        if((readfd=open(FIFO2, 0))<0)
            perror("Child cannot open readFIFO.\n");
        // xử lý chương trình.
        printf("Enter data to FIFO1: ");
        gets(s1);
        write(writefd, s1, strlen(s1));
        read(readfd, s2, PIPE_BUF);
        printf("Parent read from child: %s\n", s2);
        while(wait((int*) 0)!=childpid);
            close(readfd);
            close(writefd);
        if(unlink(FIFO1)<0)
            perror("Cannot remove FIFO1.\n");
        if(unlink(FIFO2)<0)
            perror("Cannot remove FIFO2.\n");
        return 1;
    }
    else // Khong tao duoc tien trinh con
    {
        printf("Fork failed\n");
        return -1;
    }
}
```

Tương tự, ta cũng có thể làm tương tự bài trên, nhưng dùng số nguyên thay vì dùng chuỗi.

Ví dụ 2.2:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <sys/wait.h>

#define FIFO1 "ff.1"
#define FIFO2 "ff.2"
```

```c
#define PM 0666
extern int errno;

#define PIPE_BUF 4096

int main(int argc, char* argv[])
{
    int num1, num2;
    int childpid, readfd, writefd;
    //kiểm tra 2 file FIFO1 và FIFO2 có khởi tạo thành công?
    if((mknod(FIFO1, S_IFIFO | PM, 0)<0)&&(errno!=EEXIST))
    {
        printf("Fail to create FIFO 1. Aborted.\n");
        return -1;
    }
    if((mknod(FIFO2, S_IFIFO | PM, 0)<0)&&(errno!=EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2. Aborted.\n");
        return -1;
    }
    childpid=fork();
    if(childpid==0)  //child
    {
        //kiểm tra có thể mở 2 file FIFO1, FIFO2
        if((readfd=open(FIFO1, 0))<0)
            perror("Child cannot open readFIFO.\n");
        if((writefd=open(FIFO2, 1))<0)
            perror("Child cannot open writeFIFO.\n");
        //xử lý chương trình.
        read(readfd, &num2, sizeof(num2));
        printf("Child read from parent: %d\n", num2);
        printf("Enter response: ");
        scanf("%d",&num1);
        write(writefd, &num1, sizeof(num1));
        close(readfd);
        close(writefd);
        return 1;
    }
    else
    if(childpid>0)   //parent
    {
        //Kiểm tra có mở thành công 2 file FIFO1, FIFO2.
        if((writefd=open(FIFO1, 1))<0)
            perror("Parent cannot open writeFIFO.\n");
        if((readfd=open(FIFO2, 0))<0)
            perror("Child cannot open readFIFO.\n");
```

```
        // xử lý chương trình.
        printf("Enter data to FIFO1: ");
        scanf("%d",&num1);
        write(writefd, &num1, sizeof(num1));
        read(readfd, &num2, sizeof(num2));
        printf("Parent read from child: %d\n", num2);
        while(wait((int*) 0)!=childpid);
            close(readfd);
            close(writefd);
        if(unlink(FIFO1)<0)
            perror("Cannot remove FIFO1.\n");
        if(unlink(FIFO2)<0)
            perror("Cannot remove FIFO2.\n");
        return 1;
    }
    else // Khong tao duoc tien trinh con
    {
        printf("Fork failed\n");
        return -1;
    }
}
```

Bài tập thực hành

Bài 1:

Tiến trình cha chuyển đổi 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 vô danh *(Unnamed Pipe).*

```
>./baitap1A.out 4
4! = 24
```

Giải lại vấn đề với đường ống có tên *(Named Pipe).*

Bài 2:

Tiến trình cha đọc hai số nguyên và một thao tác +, -, x, / và chuyển tất cả cho tiến trình con. Quá trình con tính toán kết quả và trả về cho tiến trình cha. Quá trình cha ghi kết quả vào một tệp.

```
>./baitap2A.out 4 6 +
4 + 6 = 10
```

Giải lại vấn đề với đường ống có tên *(Named Pipe).*