HỆ ĐIỀU HÀNH

TUẦN 10

GIAO TIẾP TIẾN TRÌNH – ĐƯỜNG ỐNG (PIPE) - UNNAMED PIPE

Làm việc với tập tin

```
FILE *ptr;
ptr = fopen("fileopen","mode");
```

For example,

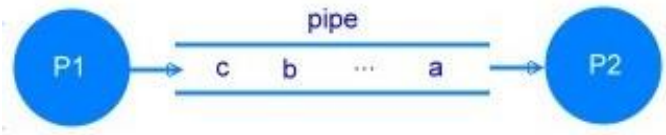
```
fopen("E:\\cprogram\\newprogram.txt","w");

fopen("E:\\cprogram\\oldprogram.txt","r");
```

```
fclose(ptr);
```

Hướng dẫn làm bài:

Giới thiệu: Một pipe là một kênh liên lạc trực tiếp giữa hai tiến trình : dữ liệu xuất của tiến trình này được chuyển đến làm dữ liệu nhập cho tiến trình kia dưới dạng một dòng các byte. Khi một pipe được thiết lập giữa hai tiến trình, một trong chúng sẽ ghi dữ liệu vào pipe và tiến trình kia sẽ đọc dữ liệu từ pipe. Thứ tự dữ liệu truyền qua pipe được bảo toàn theo nguyên tắc FIFO. Một pipe có kích thước giới hạn (thường là 4096 ký tự).



Hình Liên lạc qua pipe

Một tiến trình chỉ có thể sử dụng một pipe do nó tạo ra hay kế thừa từ tiến trình cha. Hệ điều hành cung cấp các lời gọi hệ thống read/write cho các tiến trình thực hiện thao tác đọc/ghi dữ liệu trong pipe. Hệ điều hành cũng chịu trách nhiệm đồng bộ hóa việc truy xuất pipe trong các tình huống:

Tiến trình đọc pipe sẽ bị khóa nếu pipe trống, nó sẽ phải đợi đến khi pipe có dữ liệu để truy xuất. Tiến trình ghi pipe sẽ bị khóa nếu pipe đầy, nó sẽ phải đợi đến khi pipe có chỗ trống để chứa dữ liệu. Khi thực hiện pipe sẽ có 2 loại pipe: unname pipe (pipe không tên) và name pipe (pipe có tên).

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.

Unnamed pipe:

Thường được sử dụng cục bộ

Dành cho các tiến trình có quan hệ cha con

Hàm tạo unnamed pipe:

```
int pipe(int filedes[2]);
```

Kết quả

- Thành công: 0; hai file mô tả tương ứng được trả về trong file filedes[0] và filedes[1].
- Với hệ thống cũ, filedes[0] được sử dụng để đọc, filedes[1] được sử dụng để ghi
- Trong các hệ thống mới sử dụng fullduplex, nếu filedes[0] được dùng để đọc thì filedes[1] được dùng để ghi và ngược lại.
- Thất bại trả về -1.

Ví dụ 1.1:

Tiến trình con đọc dữ liệu từ đối số truyền, ghi vào pipe. Tiến trình cha đọc từ pipe và xuất ra màn hình.

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
int main(int argc, char* argv[])
{
    char result[100]; int fp[2];
    int pid;

    if(argc<2)
    {
        printf("Doi so thieu.\n");
        return -1;
    }

    if(pipe(fp)==0)
    {
        pid = fork();
        if(pid<0)
        {
            printf("Fork failed\n");
            return -1;
        }
        else
            if(pid==0)
            {
                printf("Data from child: %s\n", argv[1]);
                close(fp[0]);
                write(fp[1], argv[1], strlen(argv[1]));
            }
```

kiem tra

ang trong thân ca tin trình con

doc gia tri tu dong` lenh

khoa nguon` doc tu duong ong

ghi gia tri tu dong` lenh

do dai` cua gia tri

```
        else
        {
            close(fp[1]);
            read(fp[0], result, strlen(argv[1]));
            printf("Read from child: %s\n", result);
        }
    } // of row 14
    else
    {
        printf("Pipe failed\n");
        return -2;
    }
}
```

doc tu cha

truoc khi doc khoa duong` ong tu thang` con

doc bao nhieu byte

Trong ví dụ trên, ta truyền đối số dưới dạng ký tự. Nếu muốn truyền đối số dưới dạng số nguyên, ta có thể dùng biến bình thường, tuy nhiên, ta phải dùng hàm `sizeof(var)` để xác định độ lớn của biến.

Ví dụ 1.2:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    int fp[2];
    int pid;
    if(argc<2)
    {
        printf("Doi so thieu.\n");
        return -1;
    }
    if(pipe(fp)==0)
    {
        pid = fork();
        if(pid<0)
        {
            printf("Fork failed\n"); return -1;
        }
        else
            if(pid==0)
            {
                int n=atoi(argv[1]);
                printf("Data from child: %d\n", n);
                close(fp[0]);
                write(fp[1], n, sizeof(n));
            }
            else
            {
                close(fp[1]);
                int tam;
                read(fp[0], tam, sizeof(tam));
```

```
            }
    } // of row 14
    else
    {
        printf("Pipe failed\n");
        return -2;
    }
}
```

Tuy nhiên, trong trường hợp 2 tiến trình đều có nhu cầu gửi dữ liệu cho nhau. Ta phải sử dụng 2 đường ống. Một đường ống chuyên gửi dữ liệu từ tiến trình 1 qua tiến trình 2, và ngược lại.

Ví dụ 1.3: Tiến trình cha cho người dùng nhập vào 1 số, gửi số này xuống tiến trình con. Tiến trình con nhận số từ tiến trình cha, tiến trình con cho người dùng nhập vào 1 số, gửi sang tiến trình cha. Tiến trình cha xuất số này ra.

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

int main(int argc, char* argv[])
    {
        int fp1[2],fp2[2];
        int pid;
    if(argc<2)
    {
        printf("Doi so thieu.\n");
        return -1;
    }
    if((pipe(fp1)==0)&&(pipe(fp2)==0))
    {
        pid = fork();
        if(pid<0)
        {
            printf("Fork failed\n");
            return -1;
        }
        else
            if(pid==0) // Trong than cua tien trinh con
            {
                close(fp1[1]);
                int tam;
                read(fp1[0], &tam, sizeof(tam));
                printf("Read from parents: %d\n", tam);
                printf("Reply from child: ");
                scanf("%d",tam);
                close(fp2[0]);
                write(fp2[1], &tam, sizeof(tam));
            }
```

```
            else // Trong than cua tien trinh cha
            {
                int n;
                scanf("%d",n);
                printf("Data from parent: %d\n", n);
                close(fp1[0]);
                write(fp1[1], &n, sizeof(n));
                close(fp2[1]);
                read(fp2[0], &n, sizeof(n));
                printf("Data from child %d\n",n);
            }
    } // of row 14
    else
        {
            printf("Pipe failed\n");
            return -2;
        }
}
```

Bài tập thực hành

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 vô danh *(Unnamed Pipe).*

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