

# 实 验 报 告

学 号	2103003 1009	姓 名	惠欣宇	专业班级	计算机4班
课程名称	操作系统			学期	2023年秋季学期
任课教师	窦金凤	完成日期	2023.12.15	上机课时间	周五 3-6
实 验 名 称	Linux 进程间的通信 — pipe 管道				

## 一、实验目的及要求

学习如何利用管道机制、消息缓冲队列、共享存储区机制进行进程间的通信，并加深对上述通信机制的理解。

## 二、实验内容

1. 了解系统调用 pipe() 的功能和实现过程。
2. 编写一 C 语言程序，使其用管道来实现父子进程间通信。子进程向父进程发送字符串 “is sending a message to parent!”；父进程则从管道中读出子进程发来的消息，并将其显示到屏幕上，然后终止。
3. 运行该程序，观察、记录并简单分析其运行结果。

## 三、实验源码

子进程向父进程通过管道发送信息：

```

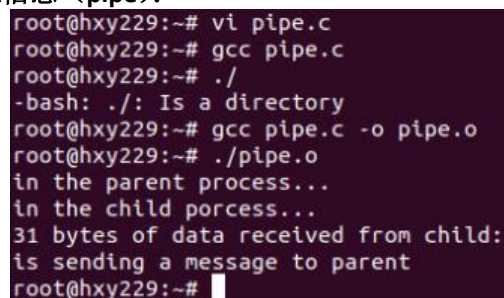
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/types.h>
5  #include <sys/wait.h>
6  #include <string.h>
7  int main() {
8      //filedes用于存放管道两端的文件描述符，filedes[0]为读出端，filedes[1]为写入端
9      int filedes[2];
10     pipe(filedes); //利用filedes创建一个管道
11     if (fork() == 0) { //子进程创建成功
12         printf("in the child process...\n");
13         //使用str存放子进程需要向管道中写入的信息
14         char str[101] = "is sending a message to parent\n";
15         close(filedes[0]); //子进程负责写入，因此关闭读出端
16         write(filedes[1], str, strlen(str)); //子进程执行对管道的写入操作
17     } else {
18         printf("in the parent process...\n");
19         close(filedes[1]); //父进程负责读出，因此关闭写入端
20         char buf[31]; //使用buf存放父进程需要从管道中读出的信息
21         read(filedes[0], buf, sizeof(buf)); //父进程执行对管道的读出操作
22         //打印从管道中读出的信息量的信息
23         printf("%ld bytes of data received from child:\n", sizeof(buf));
24         printf("%s", buf); //打印从管道中读出的信息
25     }
26     return 0;
27 }
```

父进程通过管道向子进程发送信息：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6 #include <string.h>
7 int main() {
8     //filedes用于存放管道两端的文件描述符，filedes[0]为读出端，filedes[1]为写入端
9     int filedes[2];
10    pipe(filedes); //利用filedes创建一个管道
11    int id = 0;
12    if ((id = fork()) > 0) { //父进程中
13        printf("in the father porcess...\n");
14        //使用str存放父进程需要向管道中写入的信息
15        char str[101] = "is sending a message to son\n";
16        close(filedes[0]); //父进程负责写入，因此关闭读出端
17        write(filedes[1], str, strlen(str)); //父进程执行对管道的写入操作
18    } else if (id == 0){
19        printf("in the son process...\n");
20        close(filedes[1]); //子进程负责读出，因此关闭写入端
21        char buf[31]; //使用buf存放子进程需要从管道中读出的信息
22        read(filedes[0], buf, sizeof(buf)); //子进程执行对管道的读出操作
23        //打印从管道中读出的信息量的信息
24        printf("%ld bytes of data received from father:\n", sizeof(buf));
25        printf("%s", buf); //打印从管道中读出的信息
26    }
27    return 0;
28 }
```

#### 四、实验结果

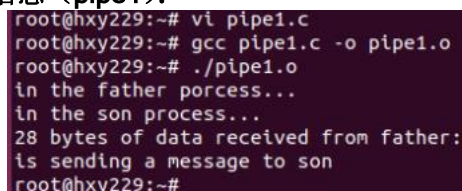
子进程向父进程通过管道发送信息 (pipe):



```
root@hxy229:~# vi pipe.c
root@hxy229:~# gcc pipe.c
root@hxy229:~# ./
-bash: ./: Is a directory
root@hxy229:~# gcc pipe.c -o pipe.o
root@hxy229:~# ./pipe.o
in the parent process...
in the child porcess...
31 bytes of data received from child:
is sending a message to parent
root@hxy229:~#
```

图 1 pipe 程序执行结果 (子 -> 父)

父进程通过管道向子进程发送信息 (pipe1):



```
root@hxy229:~# vi pipe1.c
root@hxy229:~# gcc pipe1.c -o pipe1.o
root@hxy229:~# ./pipe1.o
in the father porcess...
in the son process...
28 bytes of data received from father:
is sending a message to son
root@hxy229:~#
```

图 2 pipe1 程序执行结果 (父 -> 子)

## 五、结果分析

`pipe` 函数会建立管道,并将文件描述词由参数 `filedes` 数组返回。

`filedes[0]`为管道里的读取端,所以 `pipe` 用 `read` 调用它

`filedes[1]`则为管道的写入端,所以 `pipe` 用 `write` 调用它。

返回值: 若成功则返回零,否则返回-1,错误原因存于 `ERRNO` 中。

错误代码:

`EMFILE` 进程已用完文件描述词最大量;

`ENFILE` 系统已无文件描述词可用;

`EFAULT` 参数 `filedes` 数组地址不合法。

下面以子进程向父进程通过管道发送信息为例来解读一下管道通信的过程:

在代码中,我们首先定义了一个长度为 2 的数组 `filedes`,用于存放管道两端的文件描述符,`filedes[0]`为读出端,`filedes[1]`为写入端。然后再利用 `pipe` 函数将 `filedes` 构造为一个管道,用于子进程与父进程之间的通信使用。紧接着就由父进程调用 `fork` 创建了一个子进程,经由实验一的分析可以得知,子进程应当进入 `if(fork() == 0)`部分的代码进行执行,父进程应当进入 `else` 部分的代码进行执行。

然后子进程首先输出了一个自己的打印信息,告诉我们此时正在执行子进程,然后将子进程需要向父进程发送的信息写入 `str` 变量,然后关闭 `filedes` 管道的读出端,再调用 `write` 函数将 `str` 中的信息(即子进程需要向父进程发送的信息)写入 `filedes` 管道,保存起来。

然后在父进程中首先输出一条信息告诉我们此时正在执行父进程。在父进程中,需要对子进程写入管道的信息进行读出,因此我们需要将管道的写入端关闭。然后调用 `read` 函数从管道中将信息读出到 `buf` 变量中保存起来,然后打印出管道中信息的长度(即字符串的长度),然后再将读出的信息打印出来。

再与源代码中的 `str` 的信息进行比对,发现相同,因此我们可以知晓子进程通过 `filedes` 管道进行信息通信的过程是正确的,我在图 2 中画出了本次通信的过程,如下:

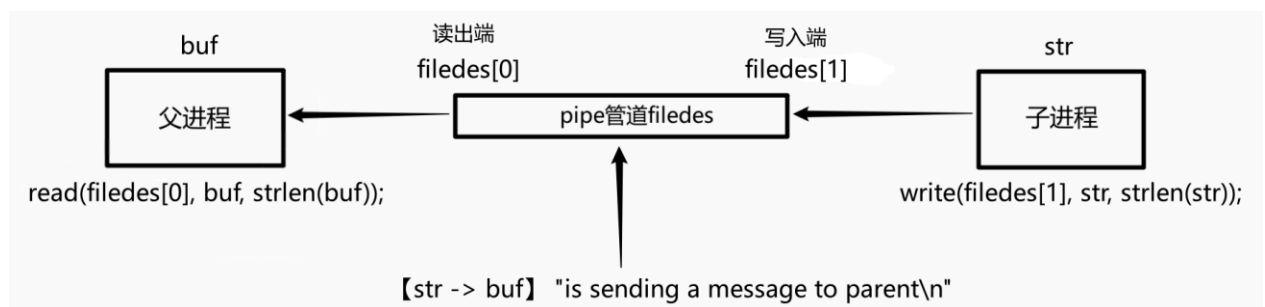


图 2 pipe 程序通信的示意图 (子 -> 父)

依据上述对子进程向父进程通过管道发送信息的分析过程，我们可以同样借此方法来分析父进程通过管道向子进程发送信息的过程，这里我就不再赘述，给出如下示意图（如图 3）：

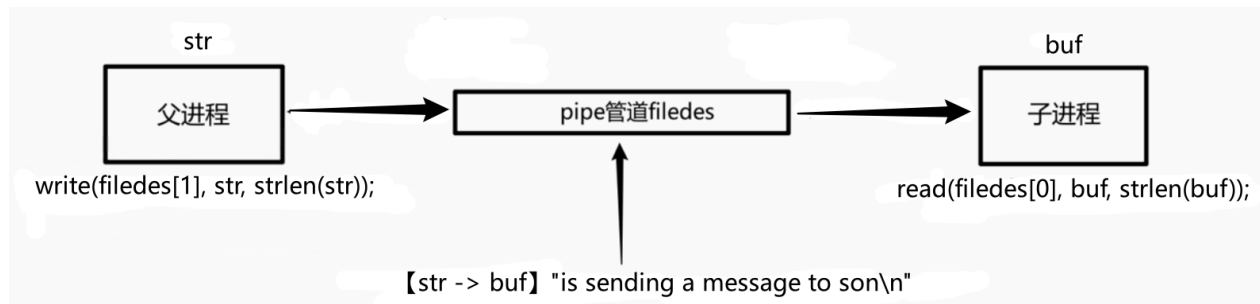


图 3 pipe1 程序通信的示意图（父 -> 子）

后续查阅资料发现管道通信具有下面的几个特点：

- ①只能用于具有共同祖先的进程（具有亲缘关系的进程）之间进行通信；通常，一个管道由一个进程创建，然后该进程调用 `fork()`，此后父子进程之间就可以应用该管道；
- ②一般而言，进程退出，管道释放，所以管道的生命周期跟随进程；
- ③一般而言，内核会对管道操作进行同步与互斥；
- ④管道是半双工的，数据只能向一个方向流动；需要双方通信时，需要建立起两个管道。