实验五 进程间通信

（二）进程的管道通信实验

【实验目的】

1、了解什么是管道
2、熟悉 UNIX/LINUX 支持的管道通信方式

【实验内容】

1、编制一段程序，实现进程的管道通信。使用 pipe()建立一条管道线。两个子进程 p1 和 p2 分别向管道各写一句话：

Child 1 is sending message!
Child 2 is sending message!

而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。

<参考程序>

```
# include<unistd.h>
# include<signal.h>
# include<stdio.h>
# include<stdlib.h>
int pid1,pid2;
main()
{
  int fd[2];
  char OutPipe[100],InPipe[100];
  pipe(fd);
while((pid1=fork())== -1);
if(pid1==0)
{
  lockf(fd[1],1,0);
  sprintf(OutPipe, “child 1 process is sending message!”);
  write(fd[1],OutPipe,50);
  sleep(5);
  lockf(fd[1],0,0);
  exit(0);
}
  else
  {
while((pid2=fork())== -1);
if(pid2==0)
```

```
{
   lockf(fd[1],1,0);
   sprintf(OutPipe, “ child 2 process is sending message!”);
   write(fd[1],OutPipe,50);
  sleep(5);
  lockf(fd[1],0,0);
  exit(0);
}
else
{
  wait(0);
  read(fd[0],InPipe,50);
  printf(“%s\n”,InPipe);
  wait(0);
  read(fd[0],InPipe,50);
  printf(“%s\n”,InPipe);
  exit(0);
}
}
}
```

实验要求：运行程序并分析结果。

2．在父进程中用 pipe()建立一条管道线，往管道里写一句话，两个子进程接收这句话。

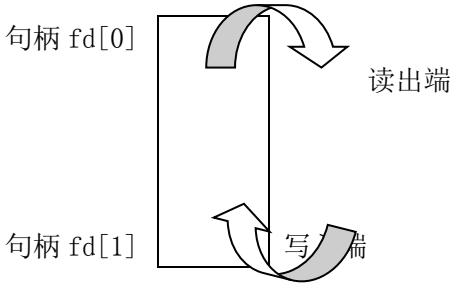
【实验报告】

1、列出调试通过程序的清单，分析运行结果。
2、给出必要的程序设计思路和方法（或列出流程图）。
3、总结上机调试过程中所遇到的问题和解决方法及感想。

【实验相关资料】

一、什么是管道

UNIX 系统在 OS 的发展上，最重要的贡献之一便是该系统首创了管道（pipe）。这也是 UNIX 系统的一大特色。
所谓管道，是指能够连接一个写进程和一个读进程的、并允许它们以生产者—消费者方式进行通信的一个共享文件，又称为 pipe 文件。由写进程从管道的写入端（句柄 1）将数据写入管道，而读进程则从管道的读出端（句柄 0）读出数据。



二、管道的类型：

1、有名管道

一个可以在文件系统中长期存在的、具有路径名的文件。用系统调用 mknod( )建立。它克服无名管道使用上的局限性，可让更多的进程也能利用管道进行通信。因而其它进程可以知道它的存在，并能利用路径名来访问该文件。对有名管道的访问方式与访问其他文件一样，需先用 open( )打开。

2、无名管道

一个临时文件。利用 pipe( )建立起来的无名文件（无路径名）。只用该系统调用所返回的文件描述符来标识该文件，故只有调用 pipe( )的进程及其子孙进程才能识别此文件描述符，才能利用该文件（管道）进行通信。当这些进程不再使用此管道时，核心收回其索引结点。二种管道的读写方式是相同的，本文只讲无名管道。

3、pipe 文件的建立

分配磁盘和内存索引结点、为读进程分配文件表项、为写进程分配文件表项、分配用户文件描述符

4、读/写进程互斥

内核为地址设置一个读指针和一个写指针，按先进先出顺序读、写。为使读、写进程互斥地访问 pipe 文件，需使各进程互斥地访问 pipe 文件索引结点中的直接地址项。因此，每次进程在访问 pipe 文件前，都需检查该索引文件是否已被上锁。若是，进程便睡眠等待，否则，将其上锁，进行读/写。操作结束后解锁，并唤醒因该索引结点上锁而睡眠的进程。

三、所涉及的系统调用

1、pipe( )

建立一无名管道。
系统调用格式

```
pipe(filedes)
```

参数定义

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

其中，filedes[1]是写入端，filedes[0]是读出端。
该函数使用头文件如下：

```
#include <unistd.h>
#inlcude <signal.h>
#include <stdio.h>
```

2、read( )

系统调用格式

```
read(fd,buf,nbyte)
```

功能：从 fd 所指示的文件中读出 nbyte 个字节的数据，并将它们送至由指针 buf 所指示的缓冲区中。如该文件被加锁，等待，直到锁打开为止。
参数定义

```
int  read(fd,buf,nbyte);
int  fd;
char *buf;
unsigned  nbyte;
```

3、write( )

系统调用格式

```
read(fd,buf,nbyte)
```

功能：把 nbyte 个字节的数据，从 buf 所指向的缓冲区写到由 fd 所指向的文件中。如文件加锁，暂停写入，直至开锁。
参数定义同 read( )。