**五、管道通信**

**5.1 实验目的**

理解进程之间通信的基本原理

掌握利用管道进行进程之间通信的方法

**5.2 预备知识**

在 Linux 系统中，进程与进程之间需要互相交换数据，即进程通信。进程通信可以发生

在同一主机的进程之间，也可以发生在不同主机与主机的进程之间。

Linux 下进程间通信的几种主要手段有：

1.无名管道（Pipe）及有名管道（named pipe）：管道可用于具有亲缘关系进程间的通信，

有名管道克服了无名管道没有名字的限制，因此，除具有管道所具有的功能外，它还允许无

亲缘关系进程间的通信；

2.信号（Signal）：信号是比较复杂的通信方式，用于通知接受进程有某种事件发生，除

了用于进程间通信外，进程还可以发送信号给进程本身；Linux 除了支持 Unix 早期信号语

义函数 sigal 外，还支持语义符合 Posix.1 标准的信号函数 sigaction（实际上，该函数是基于BSD 的，BSD 为了实现可靠信号机制，又能够统一对外接口，用 sigaction 函数重新实现了signal 函数）；

3.报文（Message）队列（消息队列）：消息队列是消息的链接表，包括 Posix 消息队列

systemV 消息队列。有足够权限的进程可以向队列中添加消息，被赋予读权限的进程则可以

读走队列中的消息。消息队列克服了信号承载信息量少，管道只能承载无格式字节流以及缓

冲区大小受限等缺点。

4.共享内存：使得多个进程可以访问同一块内存空间，是最快的可用 IPC 形式。是针对

其他通信机制运行效率较低而设计的。往往与其它通信机制，如信号量结合使用，来达到进

程间的同步及互斥。

5.信号量（semaphore）：主要作为进程间以及同一进程不同线程之间的同步手段。

6.套接口（Socket）：更为一般的进程间通信机制，可用于不同机器之间的进程间通信。

起初是由 Unix 系统的 BSD 分支开发出来的，但现在一般可以移植到其它类 Unix 系统上：

Linux 和 System V 的变种都支持套接字。

下面具体介绍管道方式：

管道允许进程之间按照先进先出的方式传送数据。管道的实现是采用一个连接写进程和

读进程的文件系统中的共享文件作为数据存储，所以该管道又称为管道（pipe）文件。写进

程将数据写入管道文件，读进程从管道文件中读出数据。管道是一种临界资源，所以对管道

的应用要考虑进程的同步。

**5.3 实验内容**

1、创建无名管道，然后向该管道写入字符，再将写入的字符读出

2、利用 fork 创建子进程，然后父子进程之间通过管道进行通信

**5.4 实验指导**

1、创建无名管道的系统调用格式为：

int pipe(filedes);

int filedes[2];

其中 filedes 是文件描述符，由 filedes[0]和 filedes[1]组成。filedes[1]是写入管道的描述符，filedes[0]是从管道读的描述符。

2、进程写管道

进程向管道写数据时必须注意管道的大小限制，管道中已有的数据与要写入之和不能大于管道的容量。

写管道的系统调用格式为：

write(filedes[1], buf, size);

int filedes[1], size;

char buf[];

其中 filedes[1]为写入管道的描述符；buf 是要写入管道的数据；size 是写入的数据长度。

3、进程读管道

进程从管道中读数据时，如果管道中已经有数据，进程按照先进先出的原则通过读指针读取数据。

读管道的系统调用格式为：

read(filedes[0], buf, size);

int filedes[0], size;

char buf[];

其中 filedes[0]为从管道读的描述符；从管道中读出的数据放入 buf 中；size 是读出数据的长度。

5.**5 参考源代码**

1、无名管道读写字符串

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<fcntl.h>

char string[]="Welcome to unnamed pipe";

int main(){

int filedes[2];

char inbuf[256];

if(pipe(filedes)<0){ //创建管道

printf("Create unnamed pipe failed\n");

return 1;

}

write(filedes[1],string,23); //写入管道

memset(inbuf,0,sizeof(inbuf)); // 把缓冲区内容都置为 0

read(filedes[0],inbuf,23); //读取管道内容到缓冲区

printf("%s\n",inbuf); //打印缓冲区信息

return 0;

}

2、父子进程通过管道进行通信

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#include<fcntl.h>

char string[]="welcome to unnamed pipe";

int main(){

int filedes[2],i,pid;

char inbuf[256];

if(pipe(filedes)<0){

printf("Create unnamed pipe failed\n");

return 1;

}

pid=fork(); //创建子进程

if(pid<0){

printf("Create child process failed\n");

return 2;

}

if(pid>0){ //父进程，写入管道

write(filedes[1],string,23);

printf("Parent process sended\n");

wait(0);

}

if(pid==0){ //子进程，读出管道内容，并打印

memset(inbuf,0,sizeof(inbuf));

read(filedes[0],inbuf,23);

printf("Child process,received\n");

printf("%s\n",inbuf);

}

return 0;

}

0,sizeof(inbuf));