**实验5 进程通信（1）无名管道和共享内存通信**

实验指导

1. 无名管道通信

**1.无名管道通信**

管道分为无名管道（PIPE）和有名管道（FIFO）两种，无名管道用于具有亲缘关系进程间的通信，即可用于父进程和子进程间的通信。

无名管道通信是通过创建内存中的共享文件，连接一个写进程和一个读进程，由写进程从无名管道的写入端将数据写入无名管道，而读进程则从无名管道的读出端读出数据。



**2.无名管道特点**

无名管道具有以下特点：

* 无名管道只能用于具有亲缘关系的进程之间进行通信。
* 无名管道是半双工的通信模式，具有固定的读端和写端。
* 无名管道是一种特殊的文件，存在于内存中，可通过read()、write()对其进行操作。

1. 无名管道系统函数
2. pipe函数

pipe()用于创建一个无名管道。

头文件：

#include<unistd.h>

系统调用格式：

int pipe(int fd[2])

参数定义：

int fd[2];

参数说明：fd[2]是有2个整数的数组，存放打开文件描述符，其中， fd[0]指向无名管道的读端，fd[1]指向无名管道的写端。

返回值：调用成功返回0，否则返回-1。

1. read函数

从文件读出数据

头文件：

#include <unistd.h>

系统调用格式：

ssize\_t read(int fd, void \*buf, size\_t count);

参数说明：fd为文件描述符，如果从无名管道读出数据，参数fd为无名管道读端描述字fd[0]；buf为存放读出数据的缓冲区指针，count为要读取的字节数。

返回值：调用成功返回实际读取到的字节数。如果返回0，表示已到达文件尾部或无数据可读。如果为-1，则表示出错。

例：

n=read(fd[0],buf,count); /\*从无名管道读出count字节数据，并送至指针buf缓冲区。\*/

1. write函数

向文件写入数据

头文件：

#include <unistd.h>

系统调用格式：

ssize\_t write(int fd, const void \*buf, size\_t count);

参数说明：fd为文件描述符，如果向无名管道写入数据，参数fd为无名管道写入端描述字fd[1]；buf为存放写入数据的缓冲区指针，count为要写入的字节数。

返回值：调用成功会返回实际写入的字节数。如果出错返回-1，错误代码存入errno中。

例：

n=write(fd[1],buf,count); /\*把buf所指向的缓冲区中count 个字节的数据写入无名管道\*/

1. 共享内存通信

共享内存为进程提供了直接通过内存进行通信的有效手段，是进程通信中最快捷的方法。共享内存区是被多个进程共享的一部分物理内存，多个进程把该共享内存区域映射到自己的虚拟地址空间，可以直接访问该共享内存区域，从而实现多个进程通过共享内存区进行通信。

System V IPC机制下创建和使用共享内存进行通信需要有四个系统调用来完成，分别是：shmget()，shmat()，shmdt()以及shmctl()，需要如下三个头文件：

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

1. shmget()函数

创建一个新的共享内存区或打开一个已经存在的共享内存区

系统调用格式：

int shmget(key\_t key,int size,int shmflag);

参数说明：

key表示共享内存区的键值；

size是创建的新共享内存区的字节数；

shmflag是用户设置的标志（表示对该共享内存区的特殊要求），如IPC\_CREAT。

shmflag由控制命令或操作允许权组成：

控制命令为：IPC\_CREAT和IPC\_EXCL

* IPC\_CREAT表示若系统中尚无指定关键字的共享内存区，则由核心建立一个共享内存区；若系统中已有共享内存段，便忽略IPC\_CREAT。
* IPC\_EXCL与IPC\_CREAT一起使用，确保创建一个新的共享内存区。如果指定关键字的共享内存区已经存在，则给出错提示。

例：

shmid=shmget(key,size,(IPC\_CREAT|0400)); /\*创建关键字为key，长度为size的共享内存段\*/

2. shmat()函数

将共享内存区域映射到进程的虚拟地址空间。

系统调用格式：

void \*shmat(int shmid,const void \*shmaddr,int shmflag);

参数说明：

shmid 为要操作的共享内存标识符，该值一般由 shmget 函数返回。   
shmaddr 指定共享内存的映射地址。此值为 0，则由系统来选择映射的地址。  
shmflag用来指定共享内存段的访问权限和映射条件

返回值：调用成功返回共享内存区所附接到的进程虚地址，失败则返回-1。

3. shmdt()函数

把一个共享内存区从指定进程的虚地址空间断开。

系统调用格式：

int shmdt(void \*addr);

参数说明：addr是要断开连接的虚地址，即系统调用shmat()所返回的虚地址。

返回值：调用成功返回0值，调用不成功返回-1。

4. shmctl()函数

共享内存区的控制，对其状态信息进行读取和修改。

系统调用格式：

int shmctl(int shmid, int cmd, struct shmid\_ds \*buf);

shmid：共享内存标识符，由 shmget 函数返回。

cmd：要执行的操作。包括 IPC\_RMID、IPC\_SET、IPC\_STAT 和IPC\_INFO

buf ：struct shmid\_ds 结构的临时共享内存变量信息

参数说明：shmid是shmget()返回的共享内存区的引用标识符；buf是用户缓冲区地址；cmd是操作命令，shmctl()根据控制命令cmd对共享内存区进行控制。cmd的取值含义如下：

* IPC\_STAT：查询共享内存区的状态信息，返回包含在指定的shmid相关数据结构中的状态信息，并且把它放置在用户存储区中的buf指针所指的数据结构中。
* IPC\_SET：设置或改变共享内存区的属性。
* IPC\_RMID：用于删除共享内存区。在共享内存区与所有进程分离时，实际进行删除。

返回值：调用成功返回0值，否则返回-1。