**实验三 进程间通信**

# 姓 名 陈正江 学 号 19001531 成绩

实验时间 2023.11.02 指导教师(签名)

**（诚信声明：本实验报告内容，均由本人亲自上机完成。 签名：\_\_\_\_\_\_\_\_\_\_\_\_\_）**

一．实验目的

1.掌握利用管道机制实现进程间的通信的方法

2.掌握利用消息缓冲队列机制实现进程间的通信的方法

3.掌握利用共享存储区机制实现进程间的通信的方法

4.加深对上述三种通信机制的理解。

二．实验工具与设备

已安装 Linux 操作系统的计算机。

三．实验内容

1. **掌握实现进程间通信的系统调用的功能和方法**

进程通信，是指进程之间交换信息，进程之间的同步、互斥也是一种信息交换，也是一种通信。但是，这里所说的“通信”是指进程之间交换较多的信息这样一种情况，特别是在由数据相关和有合作关系的进程之间，这种信息交换是十分必要和数量较大的。进程间通信是协调解决多个进程之间的约束关系，实现进程共同进展的关键技术，是多道系统中控制进程并发执行必不可少的机制。

（1）进程的通信方式

**直接通信：**是指信息直接传递给接收方，如管道。在发送时，指定接收方的地址或标识，也可以指定多个接收方或广播式地址，send(Receiver, message)。在接收时，允许接收来自任意发送方的消息，并在读出消息的同时获取发送方的地址，receive(Sender,message)。

**间接通信：**借助于收发双方进程之外的共享数据结构作为通信中转，如消息队列。这种数据结构称为缓冲区或信箱。通常收方和发方的数目可以是任意的。

（2）进程间通信的类型

**共享存储器系统：**基于共享数据结构的通信方式，只能传递状态和整数值（控制信息），包括进程互斥和同步所采用的信号量机制，速度快，但传送信息量小，编程复杂，属于低级通信；基于共享存储区的通信方式，能够传送任意数量的数据，属于高级通信。

**消息传递系统：**在消息传递系统中，进程间的数据交换以消息为单位，用户直接利用系统提供的一组通信命令（原语）来实现通信。

**管道通信：**管道是一条在进程间以字节流方式传送的通信通道。它由 OS 核心的缓冲区（通常几十 KB）来实现，是单向的；在实质上，是一个有 OS 维护的特殊共享文件，常用于命令行所指定的输入输出重定向和管道命令。在使用管道前要建立相应的管道，然后才可使用。

1. **编写一段程序，实现进程间的管道通信。 其中，父进程通过管道向子进程发送一个字符串（子进程的进程号），子进程将它显示出来。**

**（1）程序设计如下：**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<errno.h>

#include<string.h>

int main()

{

int fd[2], p;

char pipein[100], pipeout[100];

pipe(fd);

while ((p = fork()) == -1);

if (p > 0)

{

sprintf(pipein ,"child\_id = %d", p);

write(fd[1], pipein, sizeof(pipein));

exit(0);

}

else

{

read(fd[0], pipeout, sizeof(pipeout));

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

exit(0);

}

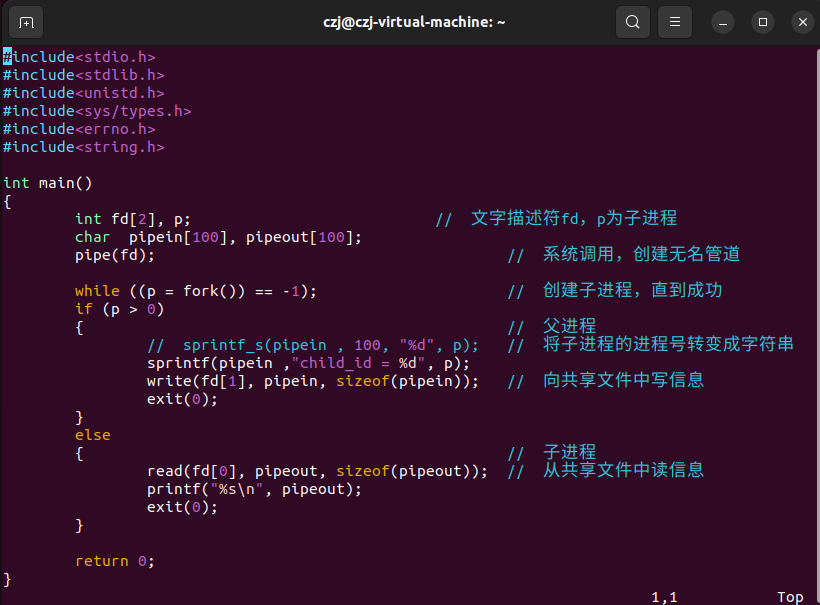
return 0;

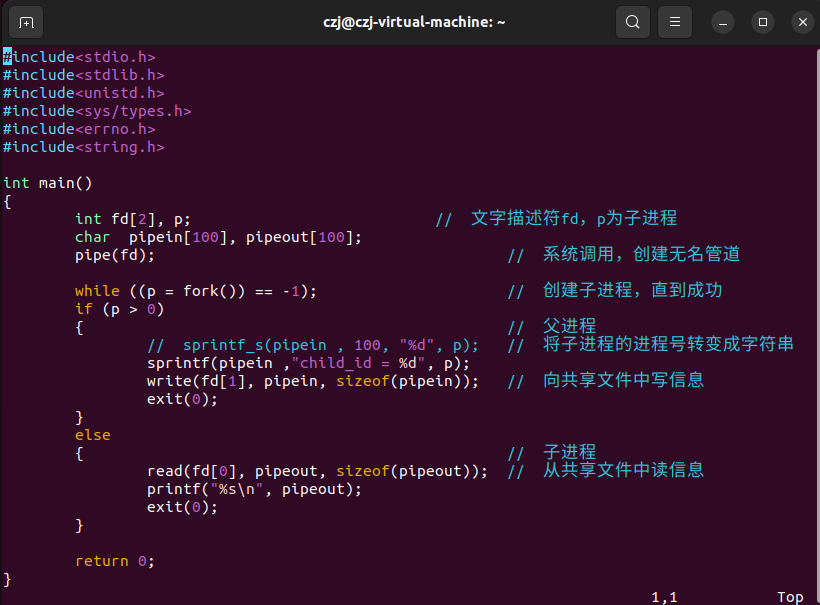
}

**（2）上机操作**

·打开终端[czj@czj-virtual-machine~]$，创建c文件，然后进入vim编辑器，编辑该文件；



·按下键盘“i”键，编写 C 语言代码；



·编辑完成后按 esc 键，输入:wq 保存并退出；

·编译链接test1102\_1.c文件，然后运行该文件，运行截图如下；



可见，子进程已经打印出了父进程发送过来的消息，子进程的进程号为12920。

1. **编写一段程序，使用消息缓冲队列来实现 client 进程和 server 进程之间的通信。**
2. **程序设计如下：**

include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<string.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#define MSGKEY 75

struct message

{

int type;

char text[1000];

}msg;

int msgqid, i;

void Client()

{

int i;

msgqid = msgget(MSGKEY,0777);

for (i = 10; i >= 1; i--)

{

msg.type = i;

msgsnd(msgqid, &msg, sizeof(msg), 0);

printf("Client has sent message to Server! %d\n", i);

}

exit(0);

}

void Server()

{

msgqid = msgget(MSGKEY, IPC\_CREAT|0777);

// 由关键字获得消息队列,因为读操作的数字代号是4，写是2，执行是1,7=4+2+1，四位数字分别代表特殊权限位，拥有者位，同组用内户位，其余用户

do{

msgrcv(msgqid, &msg, sizeof(msg), 0, 0);

printf("Server has received message from Client! %d\n", msg.type);

}while (msg.type !=1);

msgctl(msgqid, IPC\_RMID, 0);

exit(0);

}

int main()

{

while((i = fork()) == -1);

if (!i)

Server();

while((i = fork()) == -1);

if (!i)

Client();

wait(0);

wait(0);

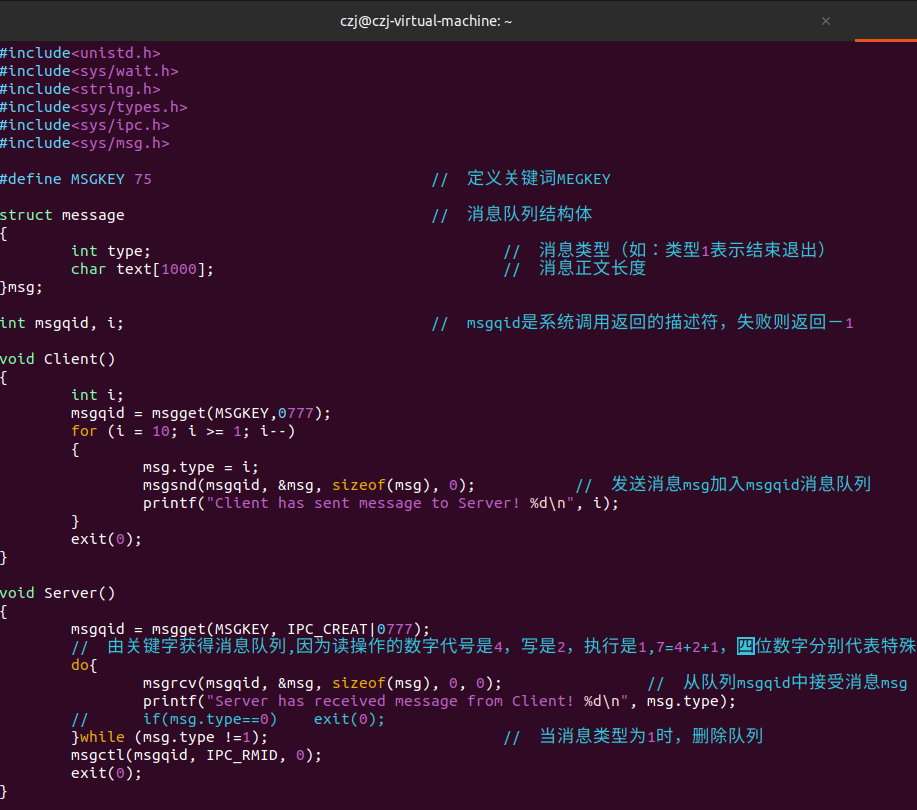
return 0;

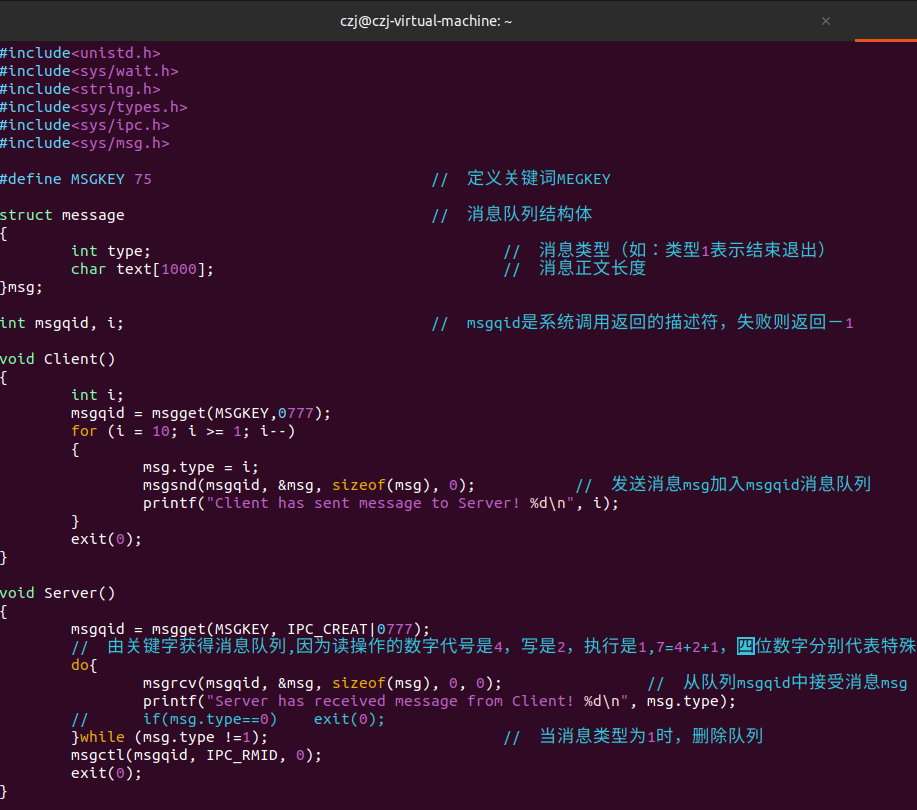
}

**（2）上机操作**

·打开终端[czj@czj-virtual-machine ~]$，创建c文件，然后进入vim编辑器，编辑该文件；



·按下键盘“i”键，编写 C 语言代码；



·编辑完成后按 esc 键，输入:wq 保存并退出；

·编译链接test1102\_2.c文件，然后运行该文件，运行截图如下；



可知Client已成功发送10条消息，并且Server接收到10条消息，当收到的消息类型为1的时候，取消该消息队列，并退出Server。

1. **编写一个与上述功能相同的程序，使其用共享存储区来实现两个进程之间的通信。**

**（1）程序设计如下：**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<string.h>

#include<sys/msg.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#define SHMKEY 75

struct sys\_data{

int type;};

int shmid, i;

void Client()

{

void\* shm = (void\*)0;

int shmid;

struct sys\_data \*da;

shmid = shmget( SHMKEY, sizeof(struct sys\_data), 0666);

if(shmid == -1)

{

printf( " shmget error\n" );

exit( -1);

}

else

printf("Client shmid = %d\n ", shmid);

shm = shmat(shmid, 0, 0);

if(shm == -1)

{

printf( " shmat error \n");

exit( -1);

}

da = (struct sys\_data\*)shm;

printf( "Client shmat start! \n");

while(da->type != 1)

{

printf( "Client type = %d \n", da->type);

while(da->type != -1); // 当共享存储区的第一个字节为-1 时

// 则表示Server空闲，可发送请求

da->type = rand()%10;

printf( "Client has sent massage to Server! %d\n" , da->type);

sleep(1);

}

exit(0);

}

void Server()

{

void\* shm = (void\*)0;

int shmid;

struct sys\_data \*da;

float ftemp = 0.0, fhumi = 0.0;

shmid = shmget( SHMKEY, sizeof(struct sys\_data), 0666|IPC\_CREAT);

if(shmid == -1)

{

printf( " shmget error\n" );

exit( -1);

}

else

printf("Server shmid = %d\n ", shmid);

shm = shmat(shmid, 0, 0);

if(shm == -1)

{

printf( " shmat error \n");

exit( -1);

}

da = (struct sys\_data\*)shm;

da->type = -1;

while(da->type != 1)

{

sleep(2);

printf(" Server has received massage from Client! %d\n ", da->type);

if(da->type == 1)

break;

da->type = -1;

}

printf( "Server received and exit successfully ! " );

shmctl(shmid, IPC\_RMID, 0);

exit(0);

}

int main()

{

while((i = fork()) == -1);

if (!i)

Server();

while((i = fork()) == -1);

if (!i)

Client();

wait(0);

wait(0);

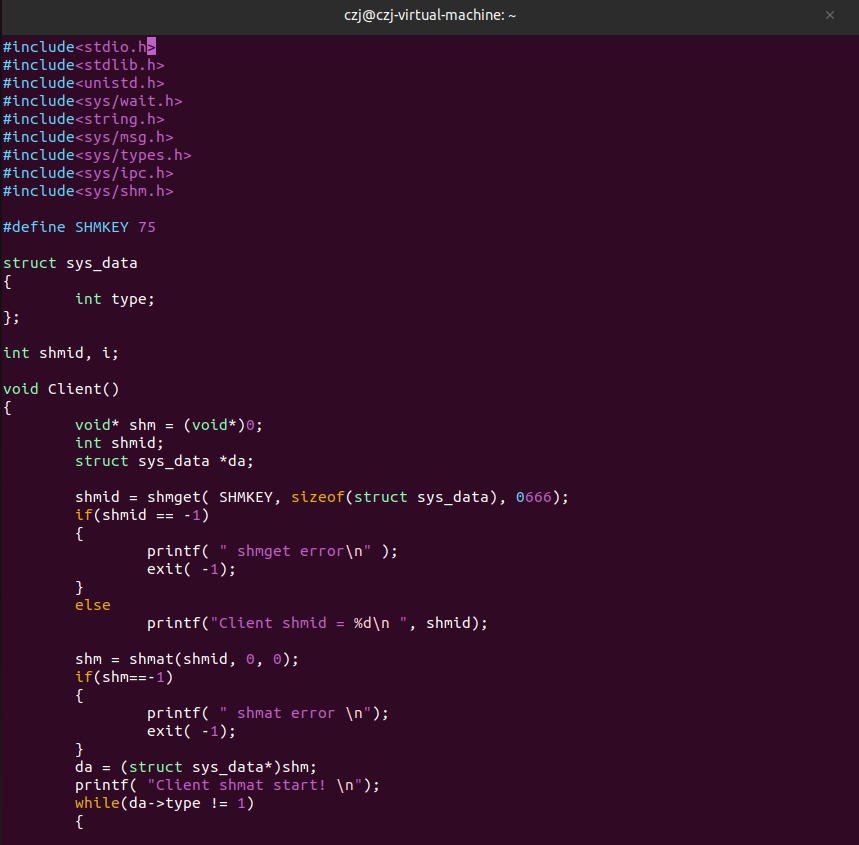
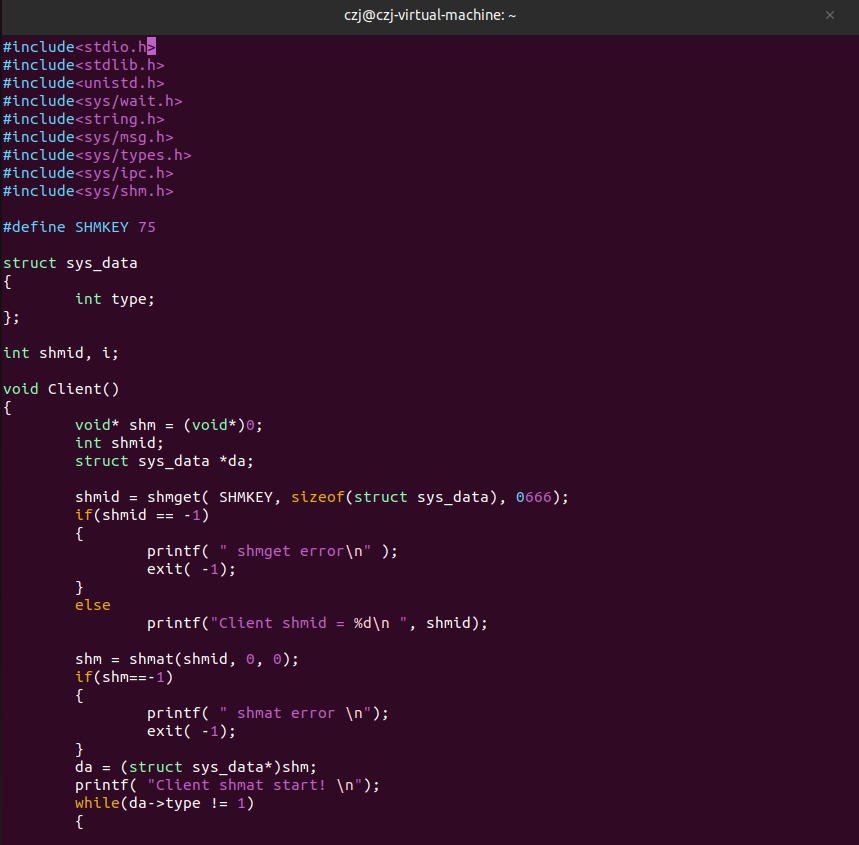
return 0;

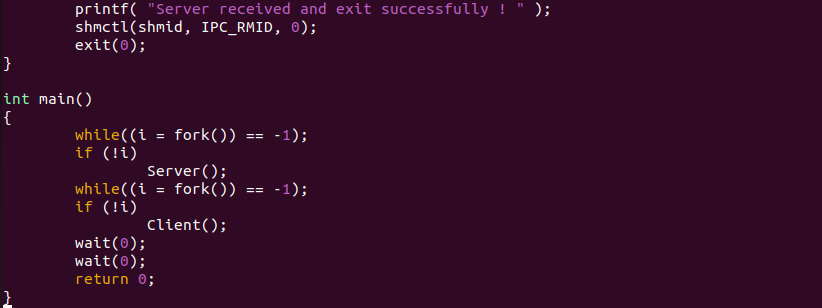
}

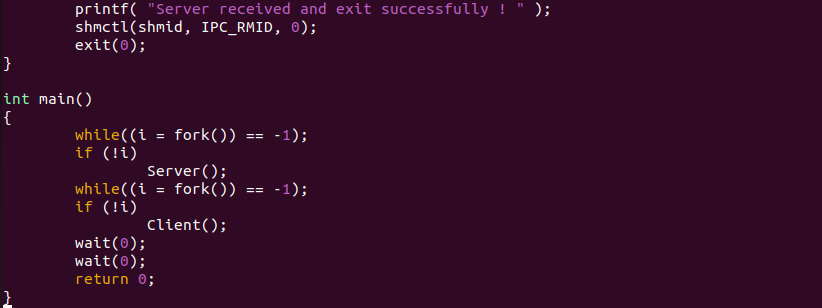
**（2）上机操作**

·打开终端[czj@czj-virtual-machine ~]$，创建c文件，然后进入vim编辑器，编辑该文件；



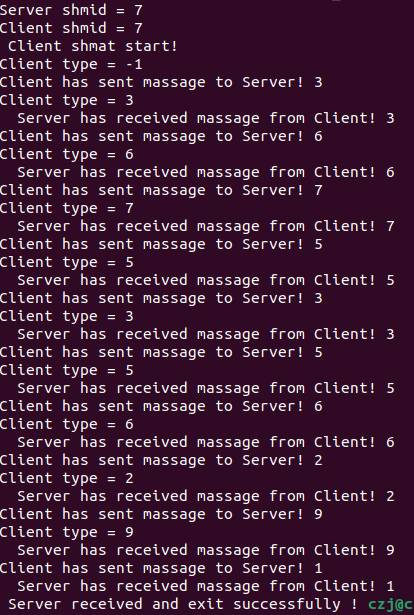
·按下键盘“i”键，编写 C 语言代码

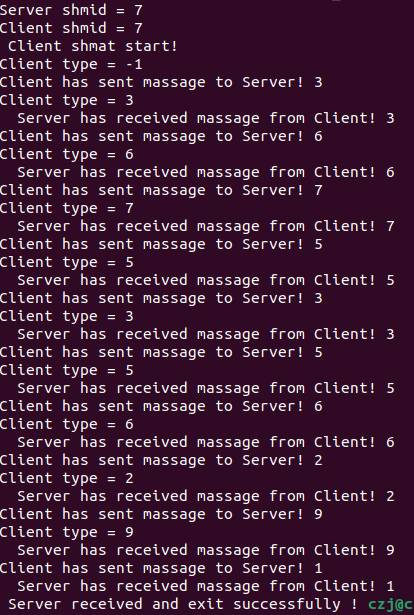




·编辑完成后按 esc 键，输入:wq 保存并退出；

·编译链接test1102\_3.c文件，然后运行该文件，运行截图如下；





可见，运行结果和预期结果一致，“Client has sent message to Server!”和“Server has received message from Client!”在屏幕上交替出现。

Client随机发送类型0~9的消息，且Server已成功接收，当消息类型为1的时候，取消该存储区，并退出Server。

但在运行过程中发现，消息显示有一定的延迟现象：每当Client发送一次消息后，Server要经过一小段时间才有回应；同样，之后Client又需等待一小段时间才发送下一个消息。出现这种延迟现象，是程序设计的问题，当Client发送消息后，并没有任何措施通知Server，此时Client并没有放弃系统控制权，仍然占用CPU的时间片；只有当系统进行调度时，切换到了Server进程才能进行应答。

# 四．思考题

1. **上述哪些通信机制提供了发送进程和接收进程之间的同步功能？这些同步是如何进行的？**

（1）上述通信机制中，管道通信机制提供了发送进程和接收进程之间的同步功能。

（2）当写进程把一定数量的数据写入管道，便去睡眠等待，直到读进程取走数据后，再把它唤醒。当读进程读一空的管道时，也应睡眠等待，直到写进程将数据写入管道后，才将之唤醒，从而实现进程的同步。

1. **上述通信机制各有什么特点，它们分别适合于何种场合？**

**（1）管道通信机制**

管道通信机制首创于UNIX系统，由于它能有效地传输大量数据，因而又被引入到许多其他操作系统中。为了协调双方通信，管道通信机制有以下三个特点：

·互斥：即当一个进程正在对管道执行读/写操作时，另一个进程必须等待；

·同步：当输入进程把一定数量的数据写入管道以后，便去睡眠等待，直到输出进程取走数据以后再将它唤醒；当读进程读一空管道时，也应睡眠等待，直到写进程将数据写入管道后再将其唤醒；

·确定对方是否存在：只有确认了对方已存在时才能进行通信。

综上，管道通信简单方便，但局限于单向通信，双方通信时需要建立起两个管道，所以，适用场合是：数据的单向流动，对于无名管道，只能用于父子进程或者兄弟进程之间（具有亲缘关系的进程之间）；有名管道允许无亲缘关系进程间的通信。

**（2）消息缓冲队列通信**

在该机制中，进程不必借助任何共享存储区或数据结构，而是以格式化的消息为单位，将通信的数据封装在消息中，并利用操作系统提供的一组通信命令，在进程间进行消息传递，完成数据交换。

消息缓冲通信机制下，消息的发送不需要接收方准备好，随时都可以发送，因此消息的传送和控制并不保证完全同步，当一个程序不再处于激活状态的时候，它有可能继续睡眠，造成上述实验内容3中发送消息多次后才接收消息的现象。

消息传递通信的特点是：隐藏了通信实现细节，使通信过程对用户透明化，降低了通信程序设计的复杂性和错误率，成为当前应用最为广泛的一类进程间通信的机制。如：在计算机网络中，消息又称为报文；在微内核操作系统中，微内核与服务器之间的通信无一例外都是采用了消息传递机制；由于该机制能很好地支持多处理机系统、分布式系统和计算机网络，因此也成为这些领域最主要的通信工具。

消息队列的特点是：消息队列是由消息类型和正文组成的链表，存放在内核中并由标识符标识；消息队列克服了传递信息少（如信号机制）、只能承载无格式字节流（如管道）以及缓冲区大小受限等缺点；这种机制下消息传递自身带有同步的控制，当未等到消息的时候，进程进入睡眠状态，不再消耗CPU资源。

**（3）共享存储区机制**

共享存储区通信的特点是：没有中间环节，直接将共享的内存页面通过附接，映射到相互通信的进程各自的虚拟地址空间中，从而使多个进程可以直接访问同一个物理内存页面，如同访问自己的私有空间一样，但实质上不是私有的而是共享，因此，这种通信方式是在同一计算机系统中诸进程间通信最快捷的方法。但是它的局限性也在此，即共享内存的诸进程必须共处同一个计算机系统，有物理内存可以共享才行。

综上，共享存储区利用存储区直接交换信息，无须复制，快捷、可通信信息量大，适用于存储区能被诸进程共同访问且高效率通信的场合。但是，共享存储区是通过将共享的存储区直接附加到进程的虚拟地址空间中来实现的，因此，无法实现这些进程之间的读写操作的同步问题，必须由各进程利用其他同步工具解决。另外，由于存储区存在于计算机系统中．所以只能由处于同一个计算机系统中的进程共享，不方便网络通信。