**实验三 进程间通信**

# 姓 名 学 号 成绩

实验时间 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。

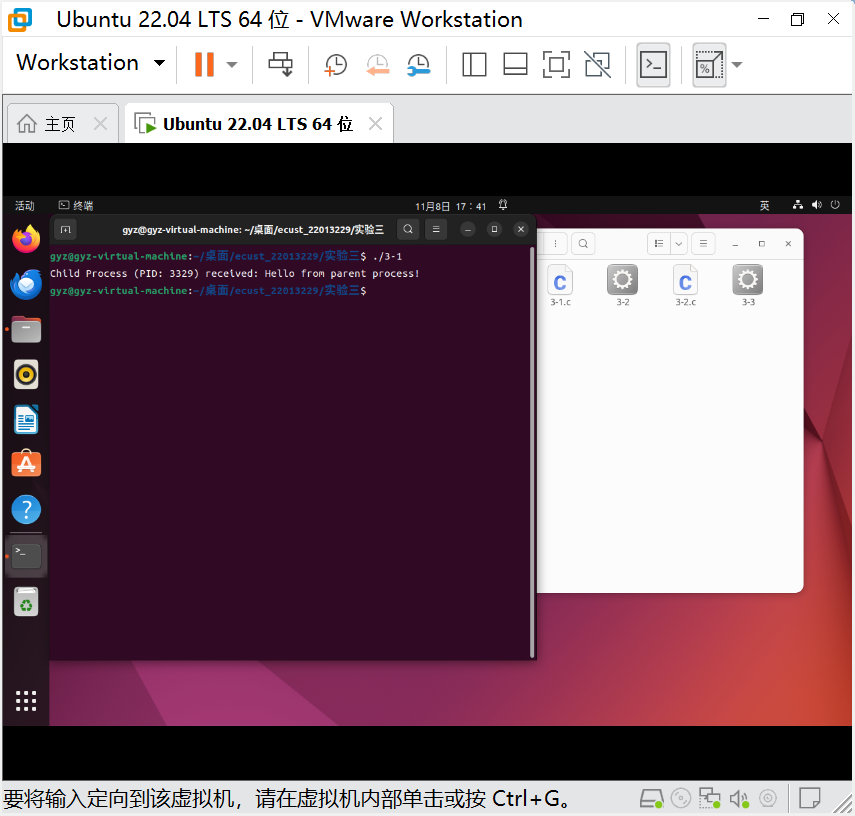
二．实验工具与设备

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**
2. **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。**

****

****

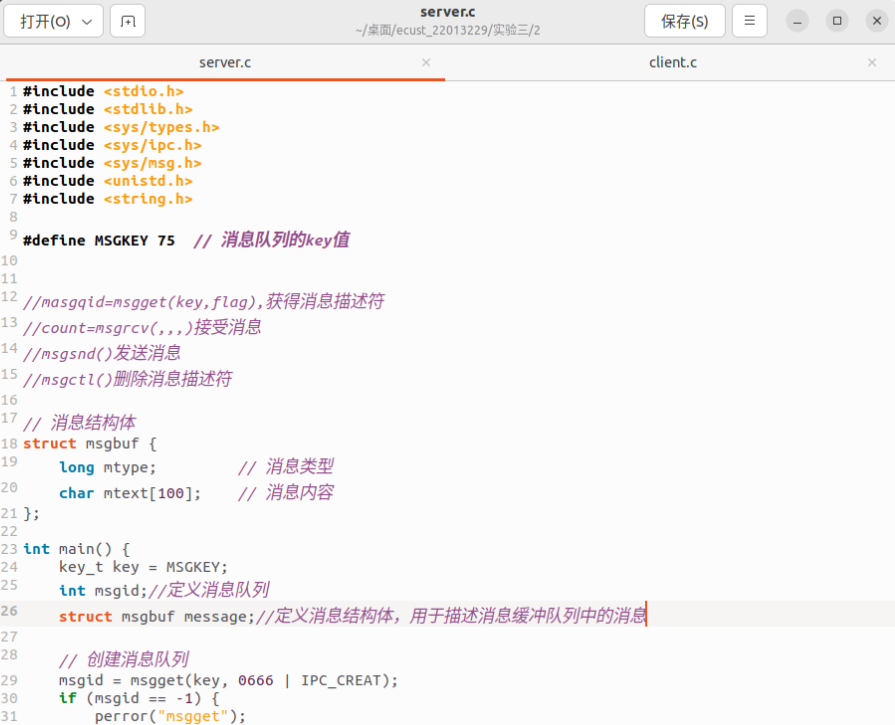
1. **编写一段程序，使用消息缓冲队列来实现 client 进程和 server 进程之间的通信。**

server 进程先建立一个关键字为 MSGKEY(如 75)的消息队列，等待其他进程发来的消息。server进程每接收到一个消息，便显示字符串“Server has received message from Client!”。当遇到类型为 1的消息，则作为结束信号，取消该队列，并退出 server。

client 进程则使用关键字为 MSGKEY 的消息队列，先后发送类型从 10 到 1 的消息，然后退出。client 进程每发送一条消息，便显示字符串“Client has sent message to Server!”。最后一条消息，是 server进程需要结束的信号。

**Server进程和client进程都向消息描述符为msgid中的message（自定义）中写读消息。**

**Server程序：**

****

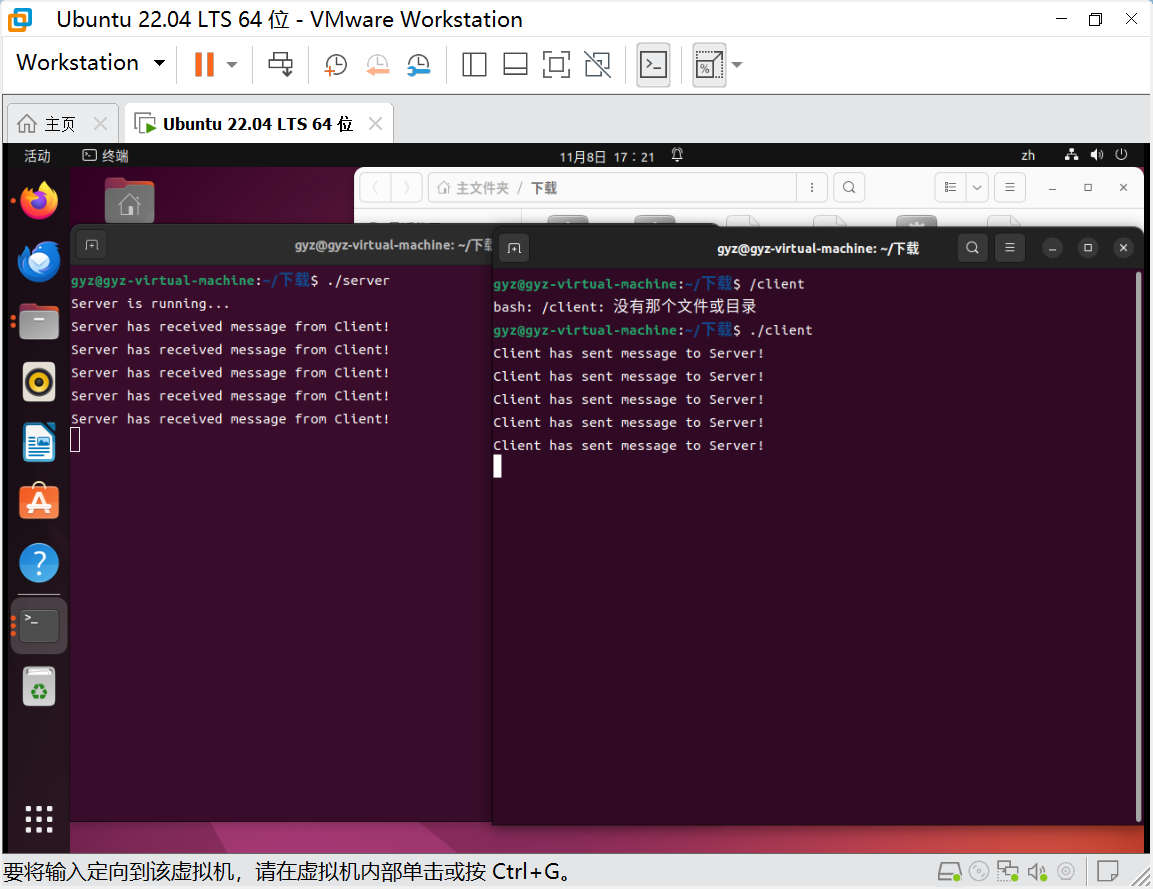
****

**Client程序：**

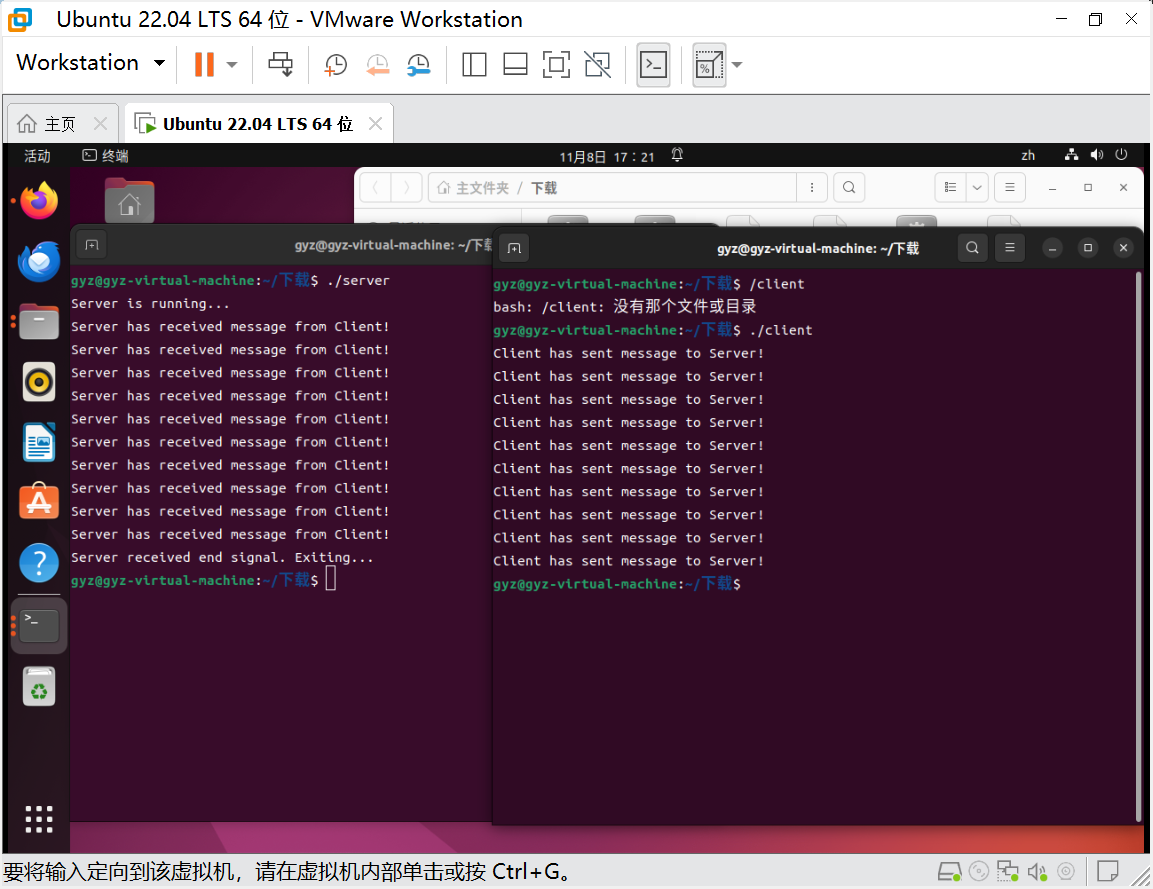
****

****

**运行中，还未结束：**

****

**运行结束：**

****

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。**

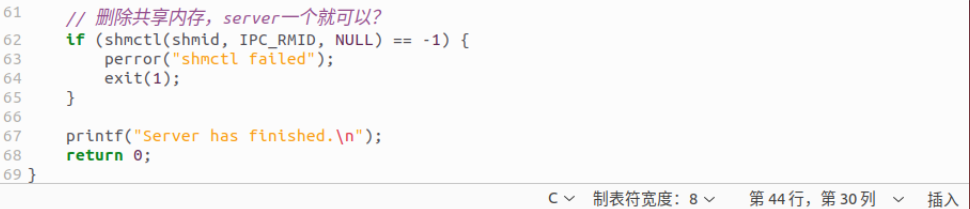
server 进程先建立一个关键字为 SHMKEY(如 75)的共享区，并将第一个字节置为-1，作为数据空的标志。等待其他进程发来的消息，当该字节的值发生变换时，表示收到了信息，进行处理，然后再次把它的值置为-1。server 进程每接收到一个消息，便显示字符串“Server has received message from Client!”。当遇到的值为 0，则视为结束信号，取消该存储区，并退出 server。

client 进程则建立一个关键字为 SHMKEY(如 75)的共享区，当共享取得第一个字节为-1 时server端空闲，可发送请求。client 随机填入 9 到 0。期间等待 server 端的再次空闲。进行完这些操作后，client 退出。client 进程每发送一条消息，便显示字符串“Client has sent message to Server!”

**Server程序：**

****

****

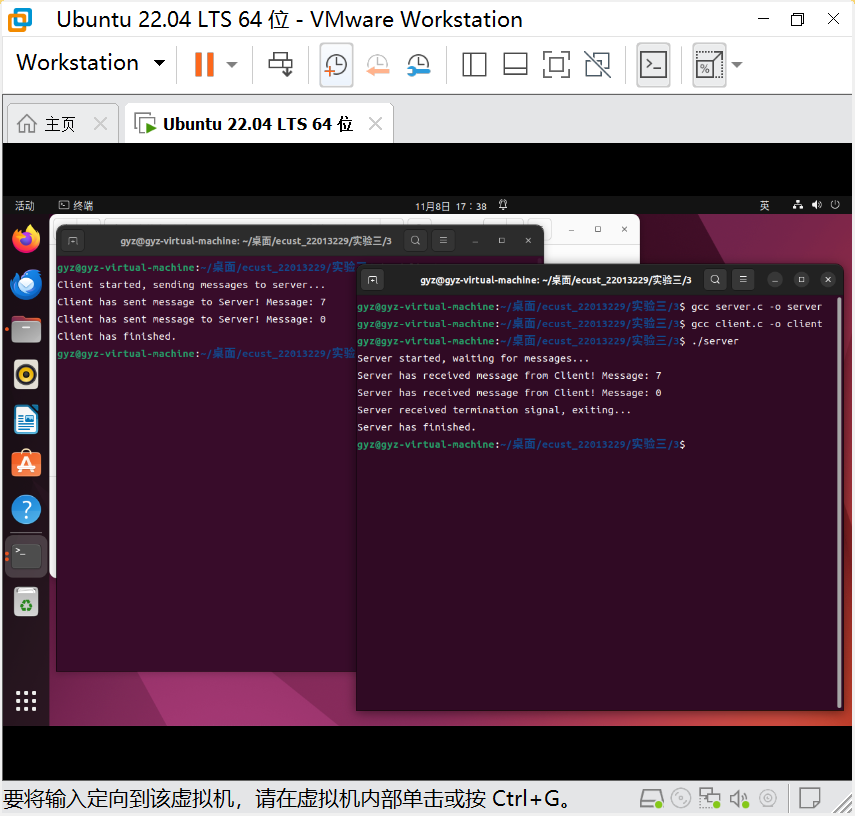
****

**Client程序：**

****

****

**运行结束：**

****

# 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）管道通信、消息缓冲队列通信提供了发送进程和接收进程之间的同步功能；而共享内存通信本身不提供同步机制，其通常需要借助其他机制来实现，例如信号量、互斥锁等。

（2）实现：

**1）管道通信：**

父进程和子进程在管道通信中存在一定的同步：

①阻塞行为：read操作是阻塞的，意味着子进程会在调用read时阻塞，直到有数据可读为止。同样，write操作会阻塞父进程，直到管道缓冲区有足够的空间来写入数据。如果管道的缓冲区满，父进程会被阻塞；如果管道为空，子进程会被阻塞。

②父子进程之间的同步：在父进程写入数据之后，子进程会读取数据。由于管道在本质上是一种缓冲区机制，因此，如果父进程提前关闭写端而子进程还没有读取到数据，会导致读取失败。因此，父子进程需要协调好管道的读写端，通常父进程在写完数据后关闭写端，子进程在读完数据后关闭读端。

**2）消息缓冲队列通信：**

消息队列提供了一种通过消息类型进行同步的机制。

①消息接收的阻塞行为：msgrcv（消息接收）操作是阻塞的，直到有消息到达队列才会返回。因此，接收进程会在调用msgrcv时阻塞，直到接收到符合条件的消息。

②结束信号同步：在消息队列的实现中，client会发送一个类型为 1 的“结束信号”消息，server在接收到该信号时退出。通过这一机制，客户端和服务器之间可以通过消息类型进行同步控制。

**3）共享存储区通信：**

共享内存本身不提供同步机制。同步通常需要借助其他机制来实现。

①同步控制：在上述代码中，client和server通过检查共享内存中的数据来同步。client在写入数据时检查\*shm\_ptr是否为空闲状态（值为-1）。一旦client写入数据，它会将共享内存标记为已处理（-1），server则读取并处理数据。

②显式同步：client和server通过对共享内存中的标记值进行检查和设置来协调执行顺序。虽然这种方式简单有效，但它依赖于进程自己控制同步，缺乏内建的同步机制，因此更容易出错。若没有合适的同步机制，可能导致竞争条件或数据一致性问题。

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：**

1）特点：

①管道是最简单的进程间通信方式，通常用于父进程与子进程之间，或者具有某种顺序的进程之间的通信。

②单向通信：传统的管道是单向通信的，即数据只能从管道的一端流向另一端。为了实现双向通信，需要创建两个管道。

③匿名管道与命名管道：匿名管道通常用于有亲缘关系的进程之间（如父子进程）。不具备文件名，进程间可以通过文件描述符来访问；命名管道可以在没有父子关系的进程间通信。与文件系统中的普通文件类似，命名管道有一个路径名，可以由不同的进程打开进行通信。

④数据传输简单：管道中的数据以字节流的方式进行传输。

⑤阻塞式：如果读取的进程没有数据可读，或者写入的进程没有空间写入，操作会被阻塞，直到有条件满足。

2）适用场合：

①父子进程之间的简单通信。

②较小的、局部的进程间通信。

**（2）消息缓冲队列通信：**

1）特点：

①消息队列是一个由多个消息组成的队列，进程可以将消息放入队列或者从队列中取出消息。

②异步通信：发送方和接收方不需要同时存在，发送方将消息发送到消息队列，接收方可以在之后的某个时刻读取消息。

③消息的结构化：消息通常包含数据和类型等信息，可以传输结构化数据。

④优先级支持：消息队列支持不同优先级的消息，发送时可以指定消息的优先级，接收时可以根据优先级选择性地读取。

⑤队列的持久性：消息队列中的消息可以被存储直到被接收，即使接收方未及时读取，消息也不会丢失。

⑥进程之间通信的灵活性：支持多个进程同时访问同一个消息队列，允许不同进程之间进行消息传递。

2）适用场合：

①多个进程之间的通信。

②需要按消息类型区分并组织数据的场合。

③需要处理顺序并且希望通过优先级控制消息的情况。

**（3）共享存储区通信：**

1）特点：

①共享内存是一种高效的进程间通信方式，允许多个进程访问同一块内存区域。

②高效性：进程直接访问共享的内存区域，速度非常快，通常是所有IPC方式中最快的。

③无数据拷贝：进程直接访问共享内存中的数据，而不需要在进程之间复制数据，这减少了系统开销。

④需同步机制：由于多个进程可能同时访问共享内存，因此需要使用同步机制（如互斥锁、信号量）来保证数据的一致性和防止竞态条件。

⑤适用大数据传输：共享内存适合需要传输大量数据的场景，因为它不需要对数据进行复制，适合大规模的数据传输。

2）适用场合：

①大规模数据交换的场合。

②需要低延迟、高效访问的场合。

③多个进程需要同时读取和写入共享资源时。