桂林电子科技大学

**计算机操作系统**  实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | **实验三 进程间通信** | | | | | | | |  | 辅导员意见：  成绩 辅导员  签 名 |
| 院 系 | 计算机与信息安全学院 | | | 专业 | | 计算机 | | |
| 学 号 | 2000500927 | | | 姓名 | | 吴河山 | | |
| 实验日期 | 2022 | 年 | 12 | | 月 | | 4 | 日 |
|  |  | | | | | | | |

## 一、实验目的

1、了解什么是信号

2、熟悉LINUX系统中进程之间软中断通信的基本原理

3、了解什么是管道

4、熟悉UNIX/LINUX支持的管道通信方式

5、了解什么是消息

6、熟悉消息传送的机理

7、了解和熟悉共享存储机制

## 二、实验内容

**（1）信号机制**

1、编写程序：用fork( )创建两个子进程，再用系统调用signal( )让父进程捕捉键盘上来的中断信号（即按^c键）；捕捉到中断信号后，父进程用系统调用kill( )向两个子进程发出信号，子进程捕捉到信号后分别输出下列信息后终止：

Child process1 is killed by parent!

Child process2 is killed by parent!

父进程等待两个子进程终止后，输出如下的信息后终止：

Parent process is killed!

2、分析利用软中断通信实现进程同步的机理

**（2）进程管道通信**

1、编写程序实现进程的管道通信。用系统调用pipe( )建立一管道，二个子进程P1和P2分别向管道各写一句话：

Child 1 is sending a message!

Child 2 is sending a message!

2、父进程从管道中读出二个来自子进程的信息并显示（要求先接收P1，后P2）。

**（3）消息发送与接收**

1、消息的创建、发送和接收。使用系统调用msgget( ),msgsnd( ),msgrev( ),及msgctl( )编制一长度为１k的消息发送和接收的程序。

## 三、实验环境

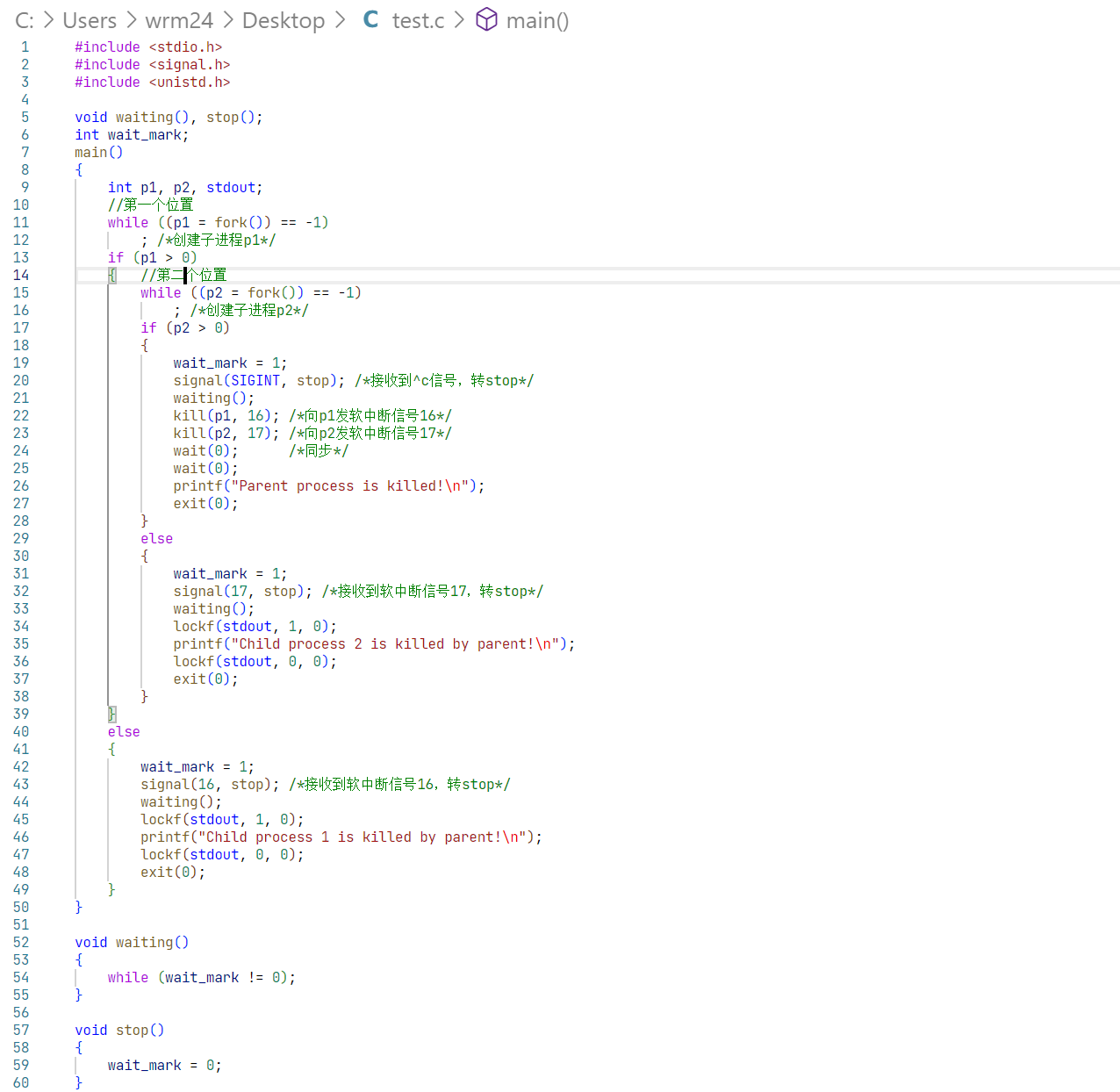
系统环境：Linux 5.15.0-56-generic #62-Ubuntu SMP Tue Nov 22 19:54:14 UTC 2022 x86\_64 x86\_64 x86\_64 GNU/Linux

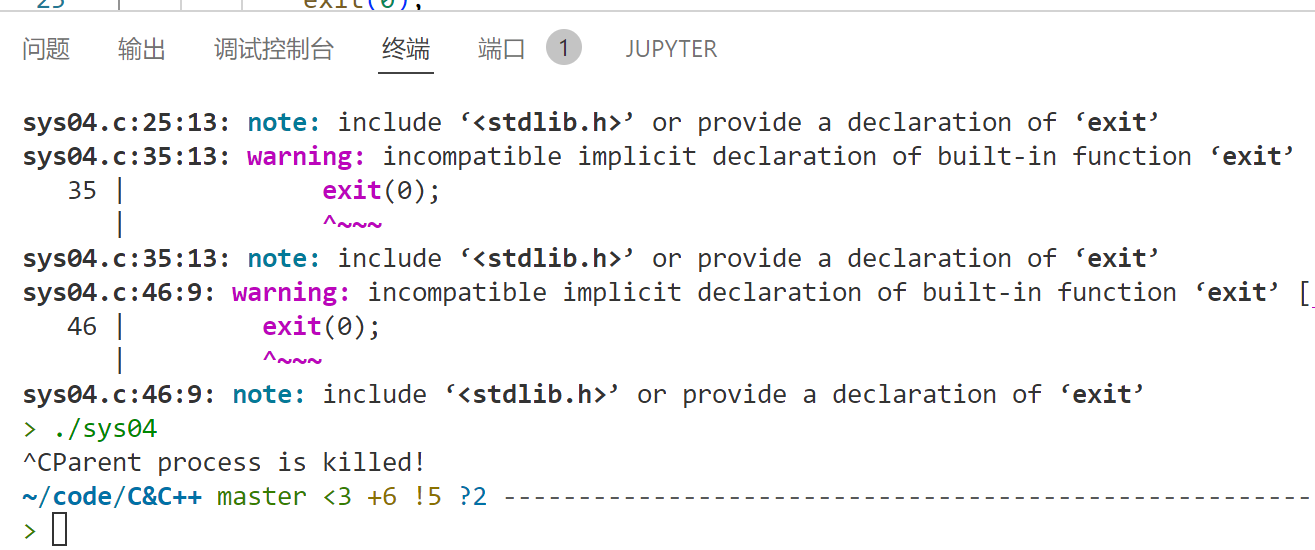
GCC版本：gcc version 11.3.0 (Ubuntu 11.3.0-1ubuntu1~22.04)

编辑器：vscode

## 四、实验步骤

**1.信号机制**





运行程序之后，按Ctrl+C发送信号给前台进程组中的所有进程，终止正在运行的程序并打印Parent process is killed!。此时，两个子进程均没有输出。function=0时，缺省值，进程在收到sig信号后应终止自己；所以在放到1时，P1终止，不在执行，同理放在P2时，两个子进程均终止，所以不输出。

**signal(SIGINT,stop)如果放在第一个位置：**

两个子进程与父进程都会输出

**signal(SIGINT,stop)如果放在第二个位置：**

此时，子进程P1不能输出，因为signal(SIGINT,stop)放在①号位置，所以预置函数对子进程p1不起作用，它的作用范围是父进程和p2，按下ctrl+c后父进程和p2会显示打印输出，而p1还是用操作系统对ctrl+c的中断的处理方法，直接返回操作系统。

**思考：**

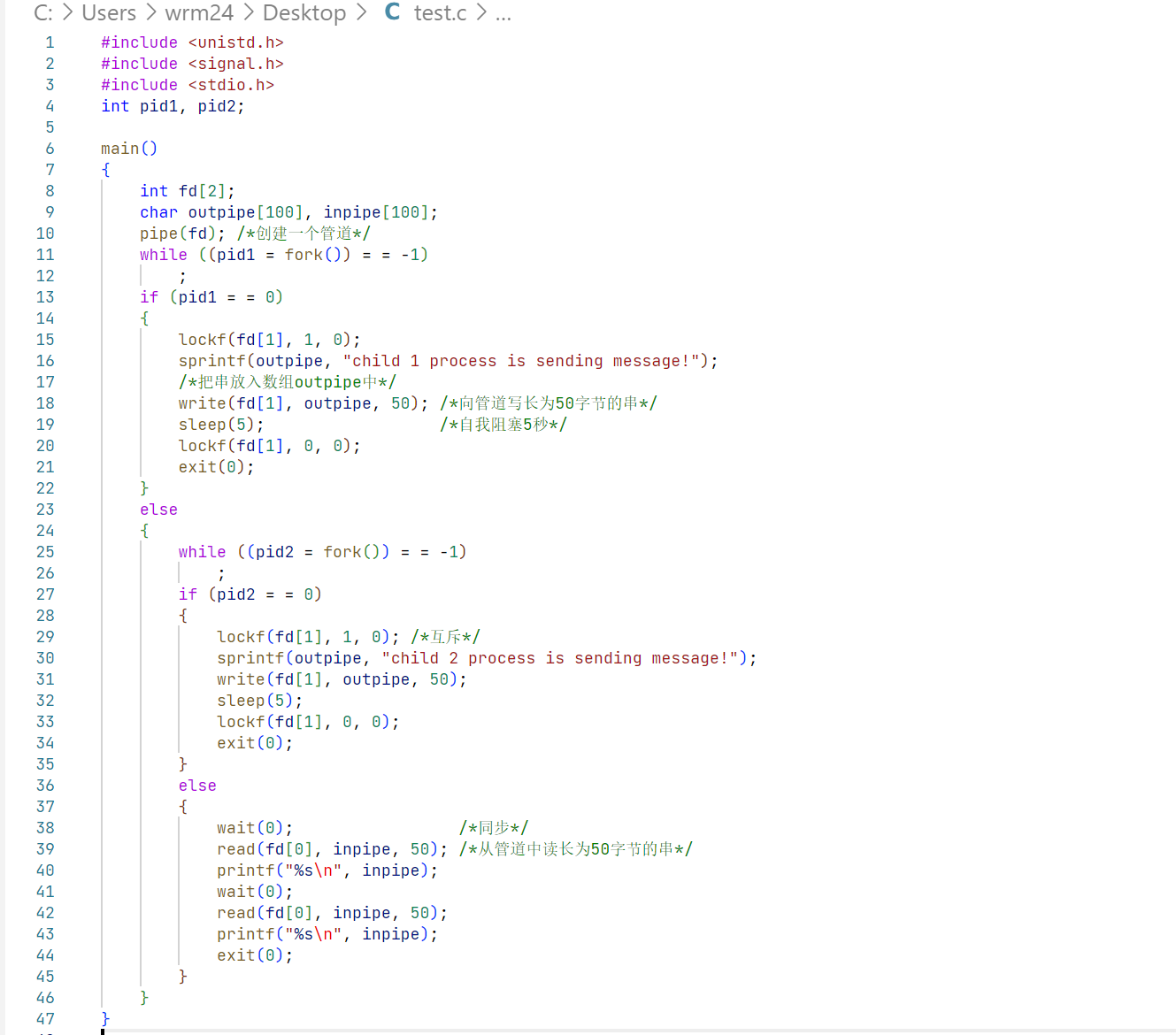
1. 该程序段前面部分用了两个wait(0)，它们起什么作用？

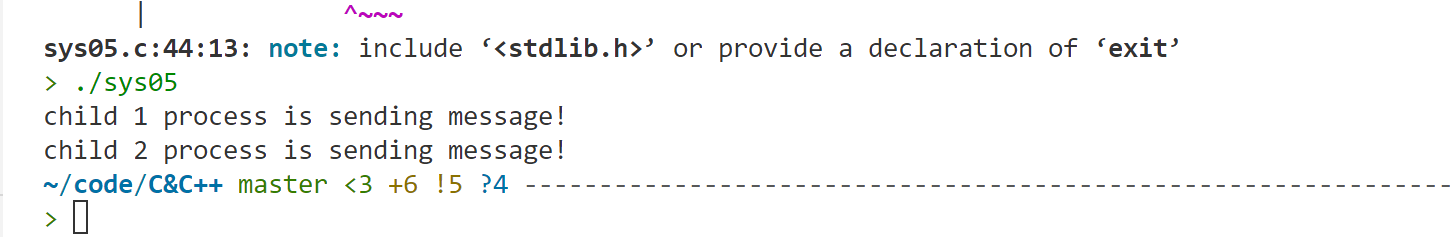
同步与等待子进程结束。

1. 该程序段中每个进程退出时都用了语句exit(0)，为什么？

会暂停目前正在执行的进程，直到有信号来到。

**2.进程管道通信**





**运行结果：**

延迟5秒后显示

child 1 process is sending message!

再延迟5秒

child 2 process is sending message!

**思考：**

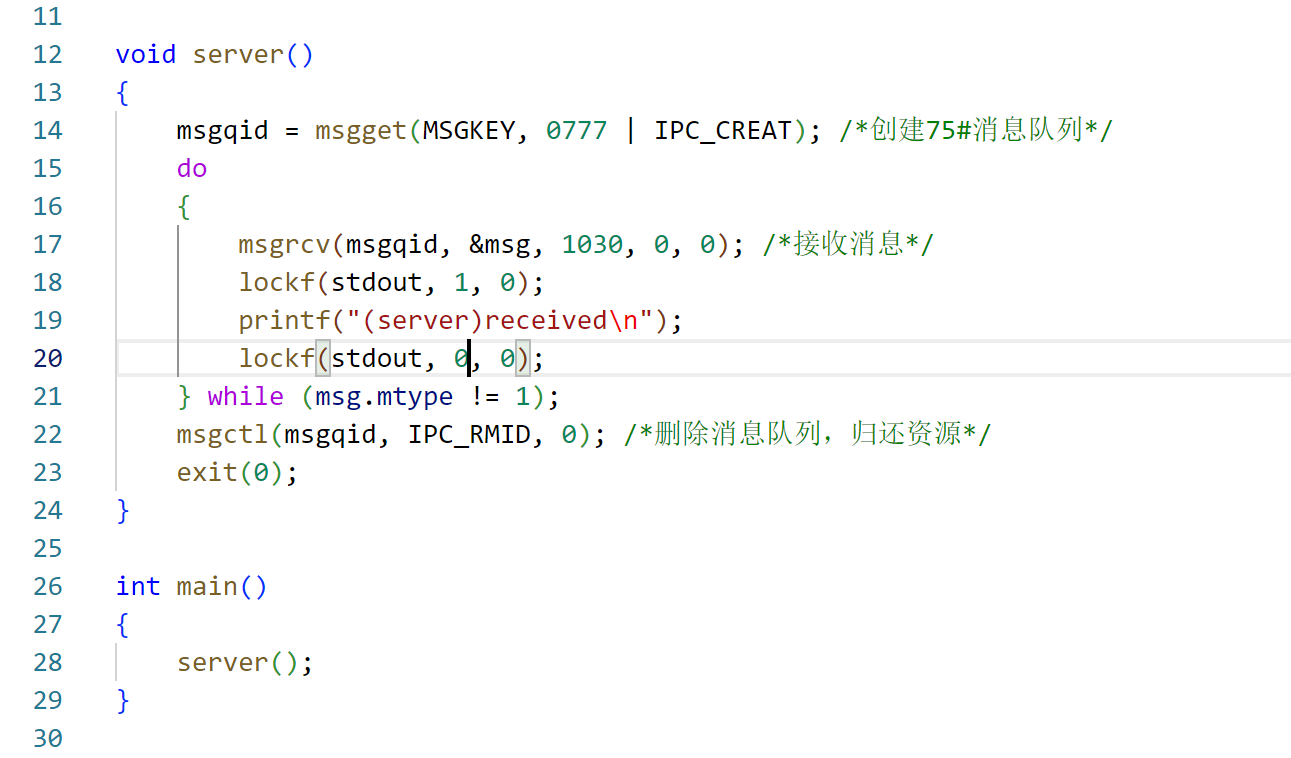
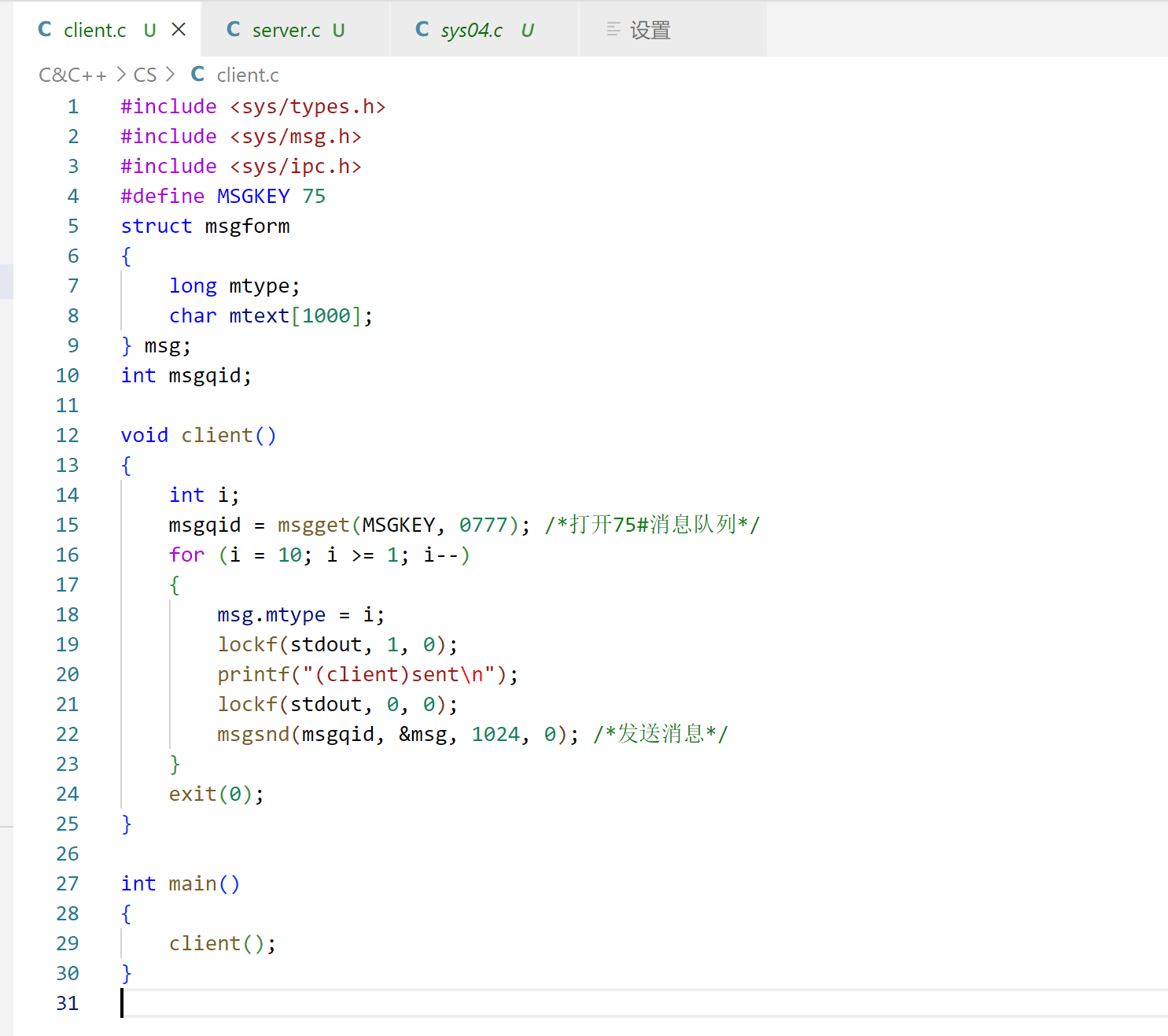
1. 程序中的sleep(5)起什么作用？

保证所有数据都传进管道中。

2、子进程1和2为什么也能**对管**道进行操作？

实验中所用到的无名管道实际上是一个没有路径的临时文件，进程通过该文件的文件描述符来识别它，而子进程会继承父进程的环境和上下文中的大部分内容，包括文件描述符，从而子进程也能对父进程中创建的管道进行操作**。**

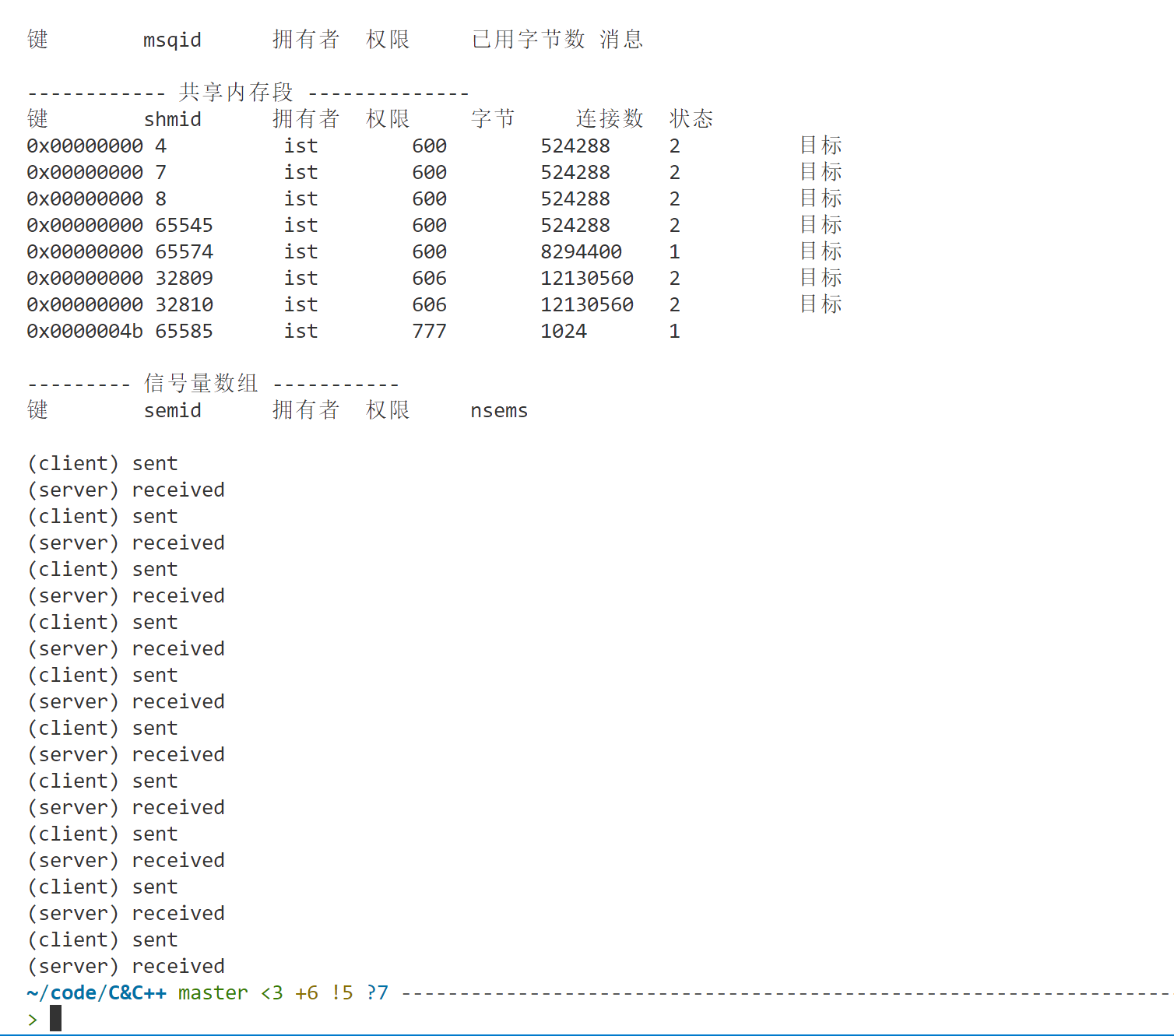
**3.消息发送与接收**



代码现象：每当client发送一个消息后，server接收该消息，client再发送下一条。也就是说“(client)sent”和 “(server)received”的字样应该在屏幕上交替出现。实际的结果大多是，先由client发送了两条消息，然后server接收一条消息。此后client 、server交替发送和接收消息。最后server一次接收两条消息。client 和server 分别发送和接收了10条消息，与预期设想一致。

**3.共享存储区通信**





思考题：

1、比较两种消息通信机制中数据传输的时间

由于两种机制实现的机理和用处都不一样，难以直接进行时间上的比较。如果比较其性能，应更加全面的分析。

（1）消息队列的建立比共享区的设立消耗的资源少。前者只是一个软件上设定的问题，后者需要对硬件的操作，实现内存的映像，当然控制起来比前者复杂。如果每次都重新进行队列或共享的建立，共享区的设立没有什么优势。

（2）当消息队列和共享区建立好后，共享区的数据传输，受到了系统硬件的支持，不耗费多余的资源；而消息传递，由软件进行控制和实现，需要消耗一定的cpu的资源。从这个意义上讲，共享区更适合频繁和大量的数据传输。

（3）消息的传递，自身就带有同步的控制。当等到消息的时候，进程进入睡眠状态，不再消耗cpu资源。而共享队列如果不借助其他机制进行同步，接收数据的一方必须进行不断的查询，白白浪费了大量的cpu资源。可见，消息方式的使用更加灵活。

**六、问题记录和实验总结**

本次实验学习了与进程通信相关的内容，了解了三种基本通信方法：消息传递、管道通信与共享存储，进程通信主要有：数据传输、资源共享、通知事件、进程控制等目的。根据实验操作验证了在GNU/Liunx下进程通信的代码过程，更加深刻理解了操作系统有关进程间的合作与通信。