实验题目：进程管理及进程通信

姓名：王波 学号：19122557 实验日期：2021年9月28日

**进程管理及进程通信**

**实验环境：**

Linux环境：虚拟机：Ubuntu 20.04.3

**实验目的:**

1.利用Linux提供的系统调用设计程序，加深对进程概念的理解。

2.体会系统进程调度的方法和效果。

3.了解进程之间的通信方式以及各种通信方式的使用。

**实验准备：**

复习操作系统课程中有关进程、进程控制的概念以及进程通信等内容（包括软中断

通信、管道、消息队列、共享内存通信及信号量概念）。

熟悉本《实验指导》第五部分有关进程控制、进程通信的系统调用。它会引导你学

会怎样掌握进程控制。

阅读例程中的程序段。

**实验方法：**

用vi 编写c 程序（假定程序文件名为prog1.c）

编译程序

$ gcc -o prog1.o prog1.c

或 $ cc-o prog1.o prog1.c

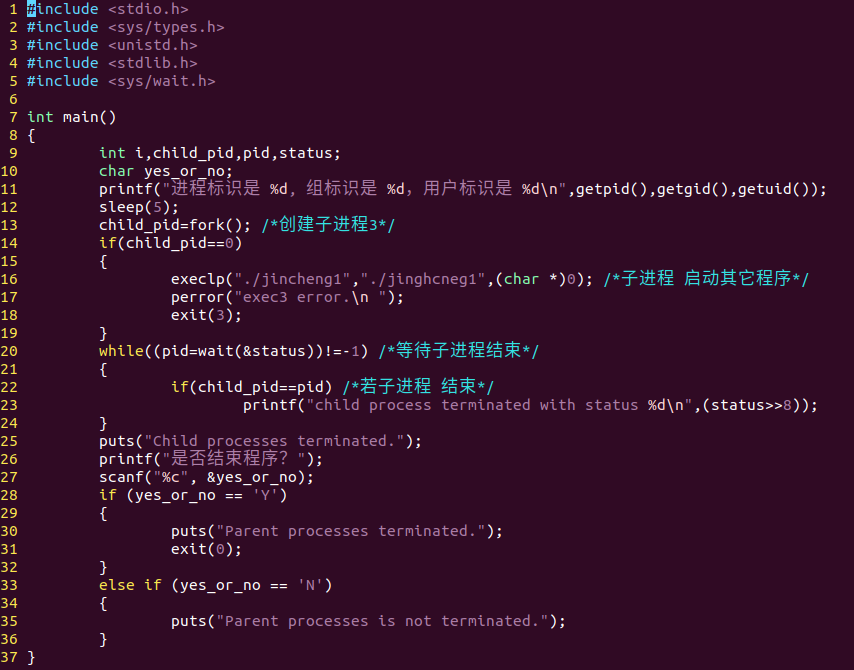
运行 $./prog1.o

**实验内容及步骤：**

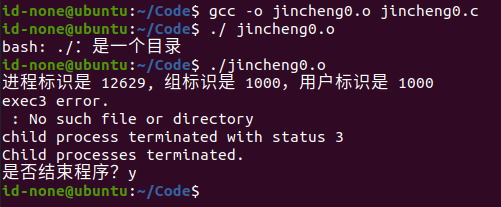
用vi 编写使用系统调用的C 语言程序。

编写程序。显示进程的有关标识（进程标识、组标识、用户标识等）。经过5秒钟后，执行另一个程序，最后按用户指示（如：Y/N）结束操作。

代码如下：

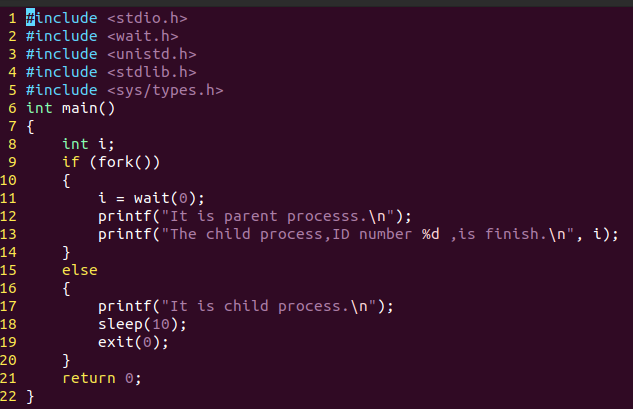


编译结果如下图所示：

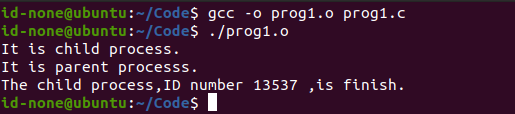


1. 参考例程1，编写程序。实现父进程创建一个子进程。体会子进程与父进程分别获得不同返回值，进而执行不同的程序段的方法。

代码如下：



编译结果如下图所示：



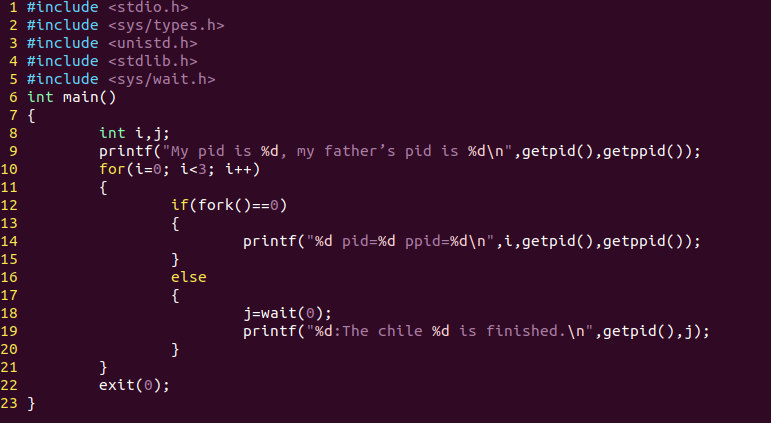
思考：子进程是如何产生的？ 又是如何结束的？子进程被创建后它的运行环境是怎

样建立的？

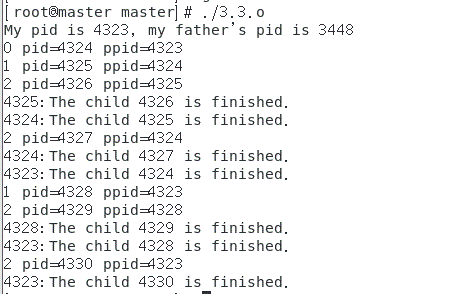
答：子进程是由父进程用fork()函数创建形成的，通过exit（）函数自我结束，子进程被创建的后核心将其分配一个进程表项和进程标识符，检查同时运行的进程数目，并且拷贝进程项目表项的数据，由子进程继承父进程的所有文件。

1. 参考例程2，编写程序。父进程通过循环语句创建若干子进程。探讨进程的家族树以及子进程继承父进程的资源的关系。

代码如图所示：



编译结果如下：

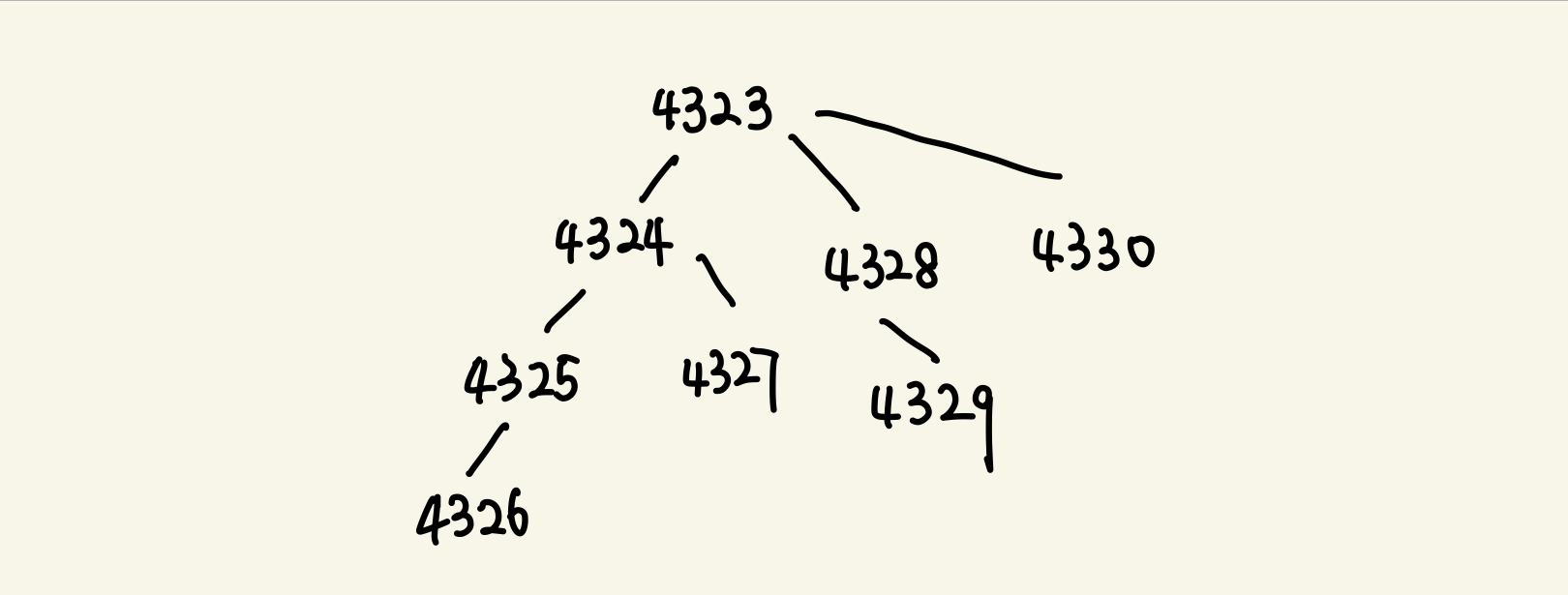


思考：

① 画出进程的家族树。子进程的运行环境是怎样建立的？反复运行此程序

看会有什么情况？解释一下。

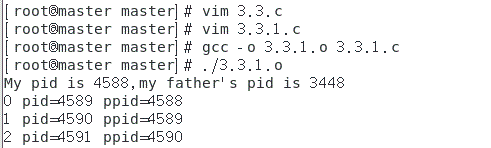
答：根据运行结果，可得进程的家族树如下：



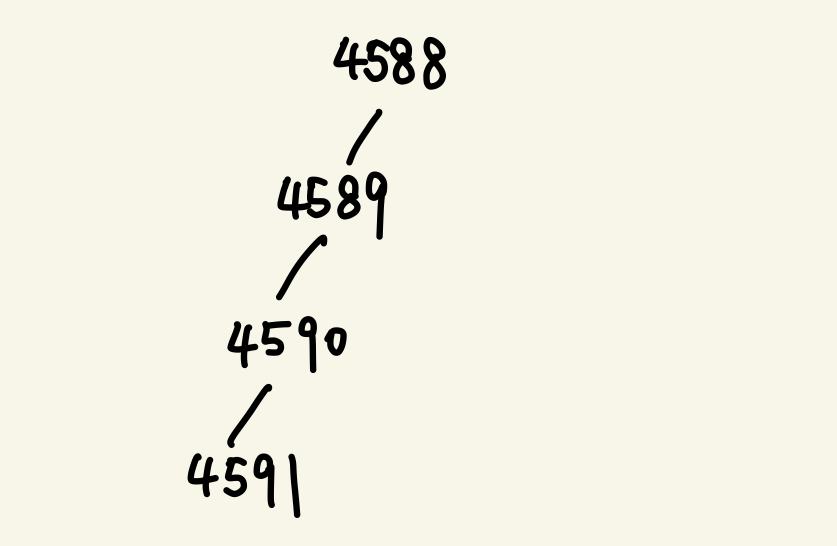
可以看到，每一次运行所返回的进程号都不同，但是最终画出的家族树形状是符合家族进程树的，经过多次运行，可以发现进程号是由系统自身随机分配的。

② 修改程序，使运行结果呈单分支结构，即每个父进程只产生一个子进程。画出进程树，解释该程序。

答：为使得运行结果呈单分支结构，只需在原有代码中的循环部分增加一个exit(0)语句，使得父进程在子进程结束后跳出循环。运行结果如下图所示：



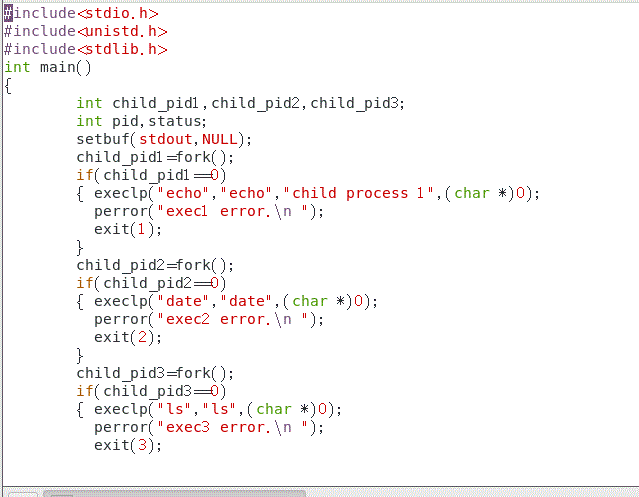
进程树如下：

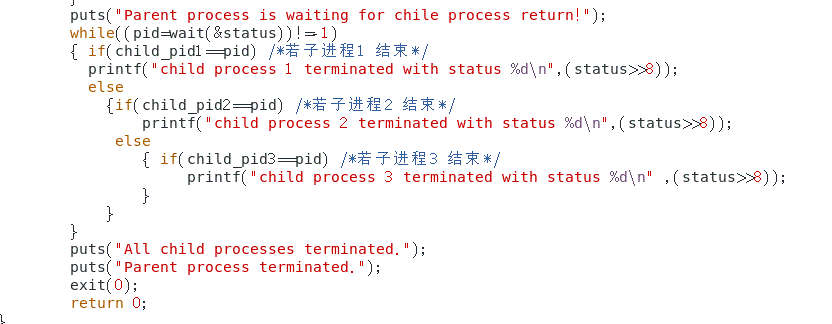


解释：当该进程为父进程时就创建子进程并退出，当该进程为子进程时返回标识号。

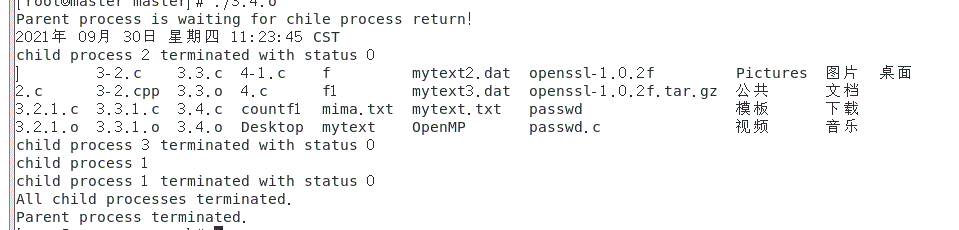
1. 参考例程3 编程，使用fork( )和exec( )等系统调用创建三个子进程。子进程分别启动不同程序，并结束。反复执行该程序，观察运行结果，结束的先后，看是否有不同次序。

代码如下所示：





运行结果如下：

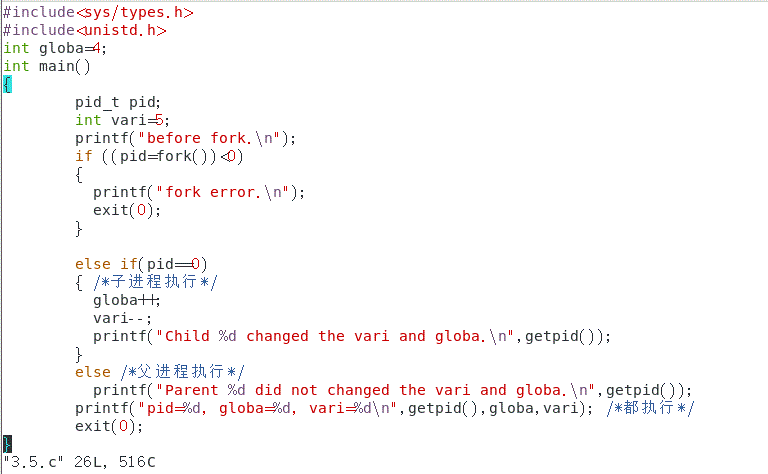
****

思考：子进程运行其它程序后，进程运行环境怎样变化的？反复运行此程序看会有什么情况？解释一下。

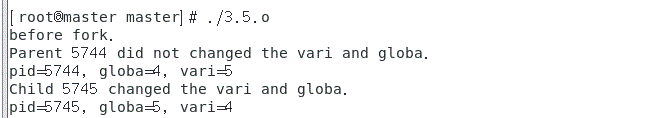
答： 子进程运行其他程序后，这个进程就完全被新程序代替。由于并没有产生新进程所以进程标识号不改变，除此之外的旧进程的其他信息，代码段，数据段，栈段等均被新程序的信息所代替。新程序从自己的main()函数开始运行。反复运行此程序发现结束的先后顺序是不可预知的，每次运行的结果不一样。原因是当每个子进程运行其他程序时，他们的结束随着其他程序的结束而结束，所以结束的先后次序在改变。

1. 参考例程4 编程，验证子进程继承父进程的程序、数据等资源。如用父、子进程修改公共变量和私有变量的处理结果；父、子进程的程序区和数据区的位置。

代码如下所示：



运行结果如下：

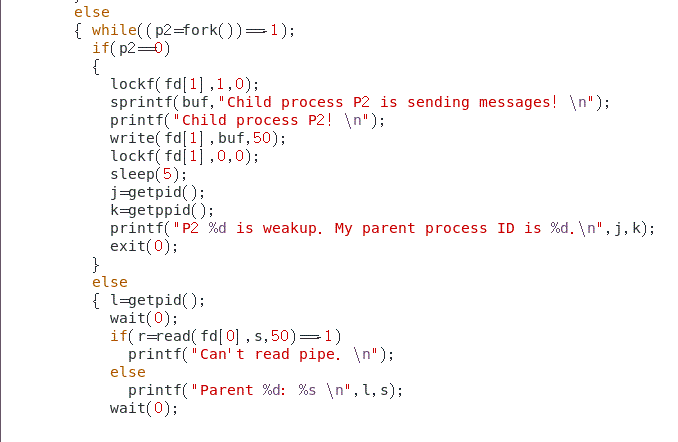
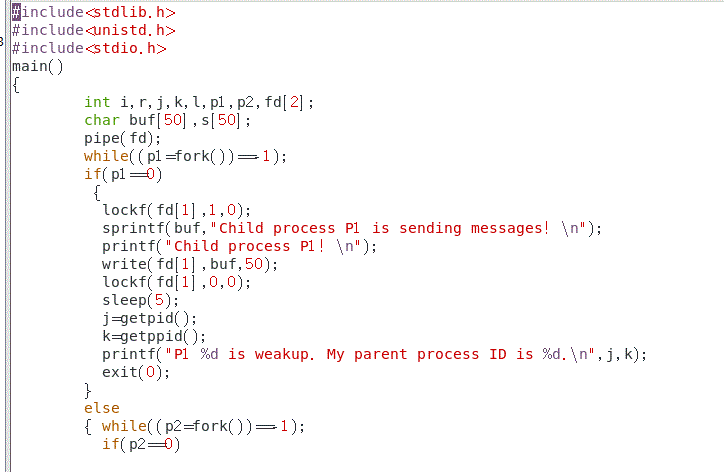


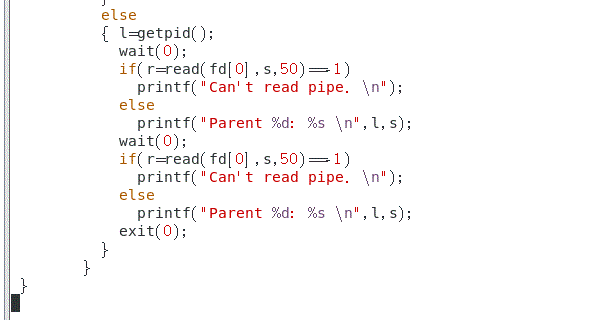
思考：子进程被创建后，对父进程的运行环境有影响吗？解释一下。

答：子进程被创建后，对父进程的运行环境无影响，因为子进程在运行时，他有自己的代码和数据段，这些都可以作修改，但是父进程的代码段和数据段是不会随着子进程的数据段和代码段的改变而改变。

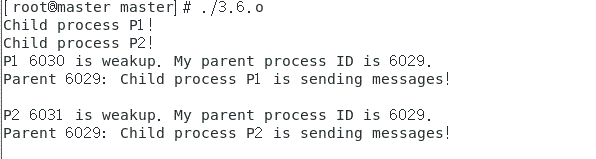
1. 参照《实验指导》第五部分中“管道操作的系统调用”。复习管道通信概念，参考例程5，编写一个程序。父进程创建两个子进程，父子进程之间利用管道进行通信。要求能显示父进程、子进程各自的信息，体现通信效果。

代码如下：



****

运行结果如下：

****

思考：

①什么是管道？进程如何利用它进行通信的？解释一下实现方法。

答：管道是指能够连接一个写进程和一个读进程，并允许他们以生产者—消费者方式进行通信的一个共享文件，又称pipe文件。由写进程从管道的入端将数据写入管道，而读进程则从管道出端读出数据来进行通信。

②修改睡眠时机、睡眠长度，看看会有什么变化。请解释。

答：修改睡眠时间和睡眠长度都会引起进程被唤醒的时间不一，因为睡眠时机决定进程在何时睡眠，睡眠长度决定进程何时被唤醒。

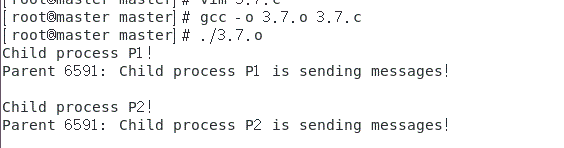
③加锁、解锁起什么作用？不用它行吗？

答：加锁、解锁是为了解决临界资源的共享问题。不用它将会引起无法有效管理数据，即数据会被修改导致读错了数据。

1. 编程验证：实现父子进程通过管道进行通信。进一步编程，验证子进程结束，由父进

程执行撤消进程的操作。测试父进程先于子进程结束时，系统如何处理“孤儿进程”的。

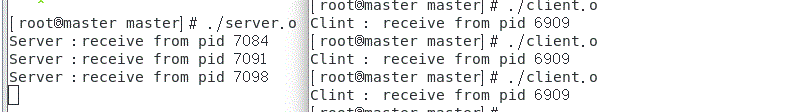
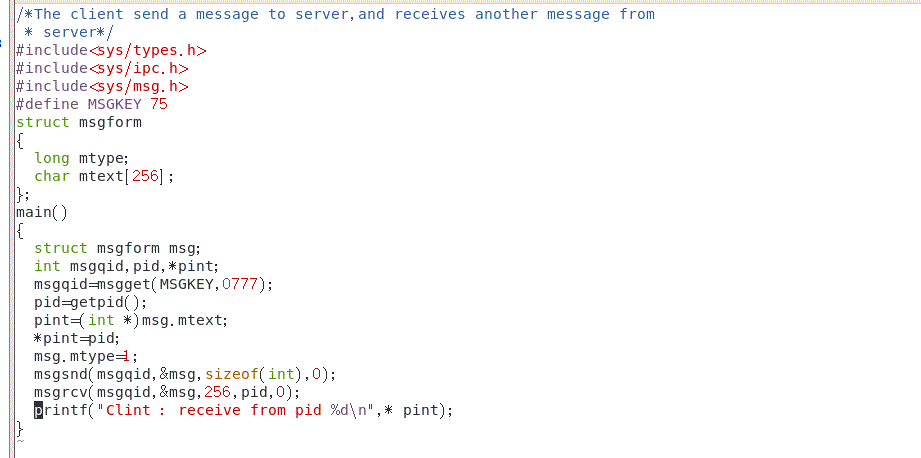
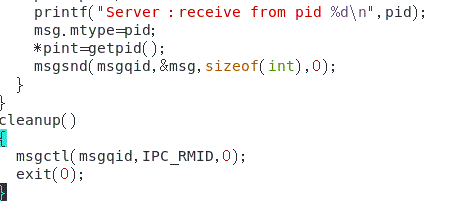
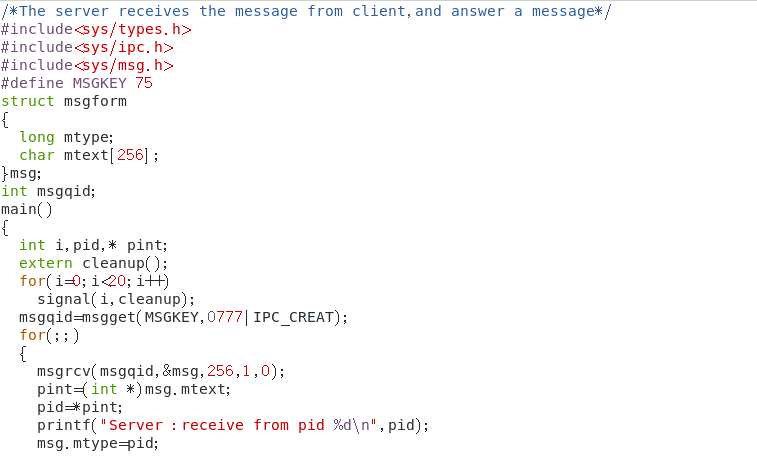
为实现父子进程通过管道进行通信，并验证子进程结束，由父进程执行撤消进程的操作。将上题中的代码进行修改，将wait（）函数删除，最终运行结果如下所示：



思考：对此作何感想，自己动手试一试？解释一下你的实现方法。

答：只要在父进程后加上wait()函数，然后打印“子进程已经结束”，一旦子进程结束，父进程撤销进程。当父进程先于子进程终止时，所有子进程的父进程改变为init进程称为由init进程领养。

1. 编写两个程序一个是服务者程序，一个是客户程序。执行两个进程之间通过消息机制通信。消息标识MSGKEY可用常量定义，以便双方都可以利用。客户将自己的进程标识（pid）通过消息机制发送给服务者进程。服务者进程收到消息后，将自己的进程号和父进程号发送给客户，然后返回。客户收到后显示服务者的pid和ppid，结束。

****

思考：想一下服务者程序和客户程序的通信还有什么方法可以实现？解释一下你的设想，有兴趣试一试吗。

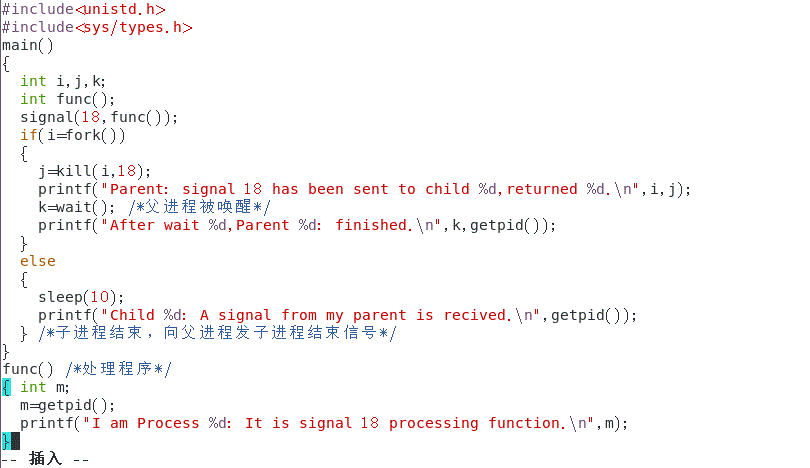
答：还可以用信号量机制来实现。信号量是一个整形计数器，用来控制多个进程对共享资源的访问。或者通过消息队列信号机制，通过向消息队列发送信息、接收信息来实现进程间的通信。

1. 这部分内容涉及《实验指导》第五部分中“有关信号处理的系统调用”。编程实现软

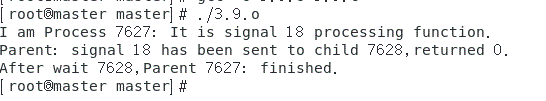
中断信号通信。父进程设定软中断信号处理程序，向子进程发软中断信号。子进程收

到信号后执行相应处理程序。

代码如下：



运行结果如下：



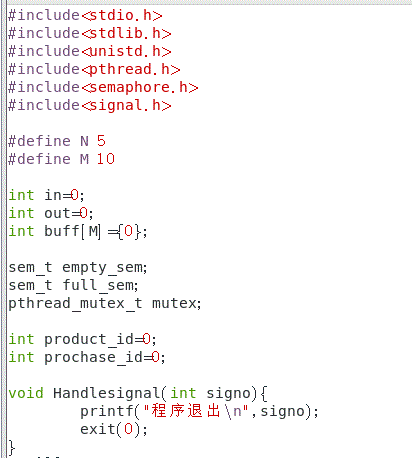
思考：这就是软中断信号处理，有点儿明白了吧？讨论一下它与硬中断有什么区别？看来还挺管用，好好利用它。

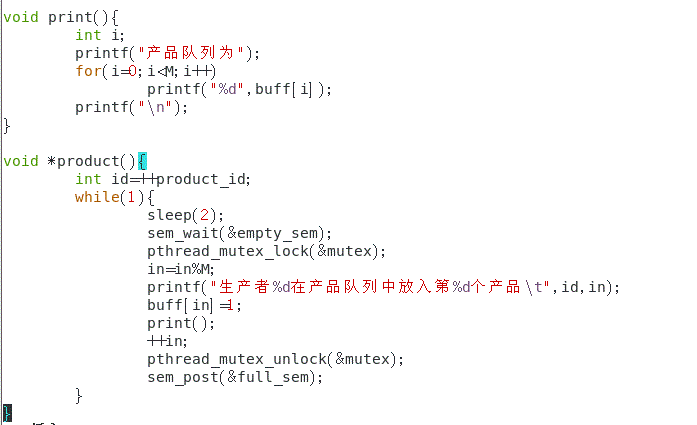
答：硬中断是由外部硬件产生的，而软中断是CPU根据软件的某条指令或者软件对标志寄存器的某个标志位的设臵而产生的。

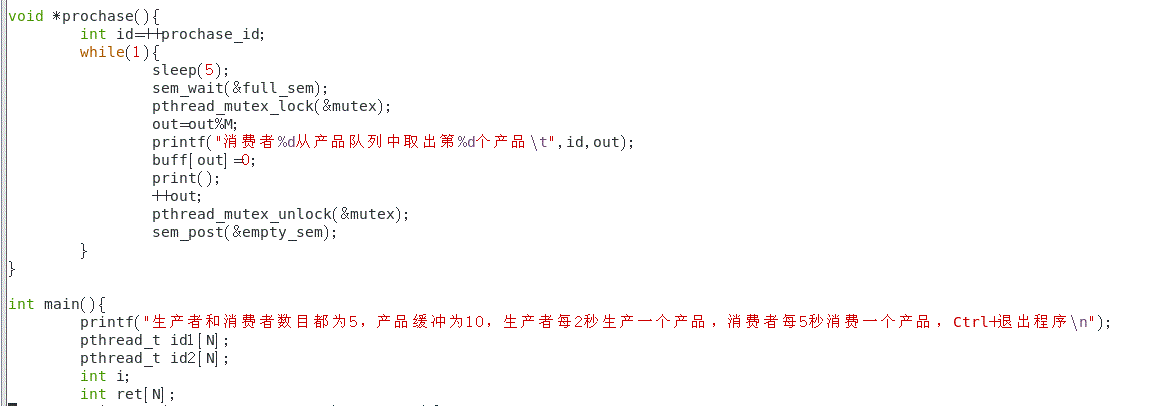
1. 怎么样，试一下吗？用信号量机制编写一个解决生产者—消费者问题的程序，这可是

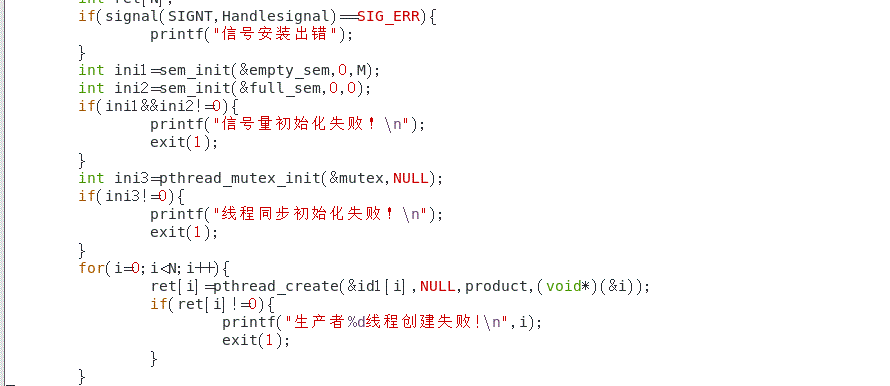
受益匪浅的事。本《实验指导》第五部分有关进程通信的系统调用中介绍了信号量机

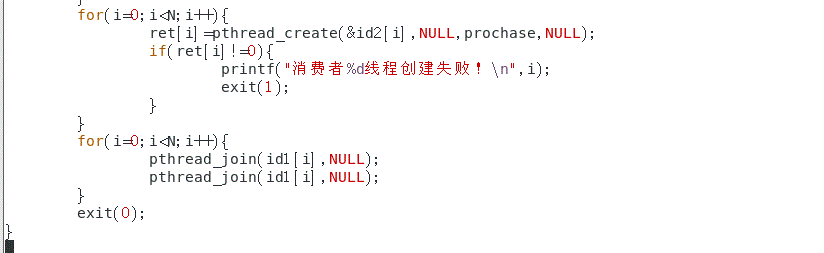
制的使用。

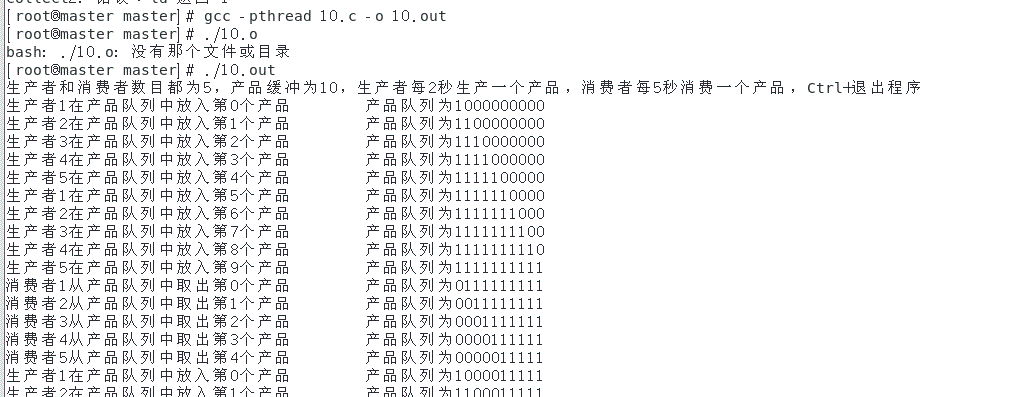












此题代码见附件源代码。

**研究并讨论：**

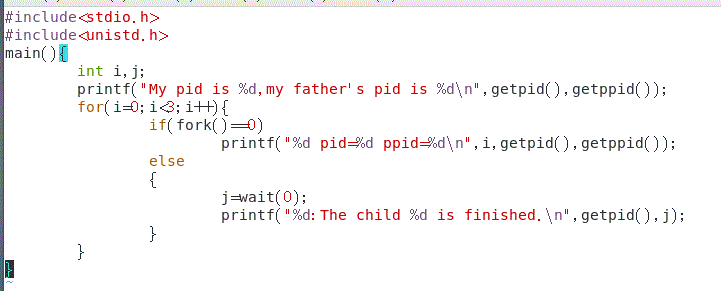
1. 讨论Linux 系统进程运行的机制和特点，系统通过什么来管理进程？

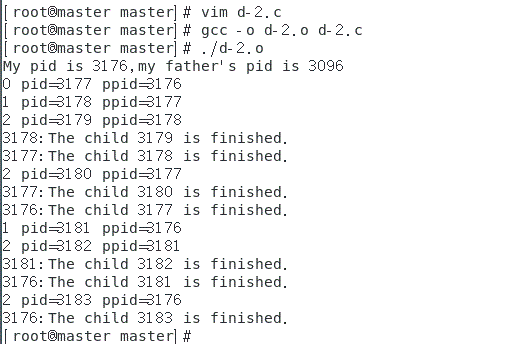
答：在Linux中，每个进程在创建时都会被分配一个数据结构，称为进程控制块（Process Control Block，简称PCB）。PCB中包含了很多重要的信息，供系统调度和进程本身执行使用。所有进程的PCB都存放在内核空间中。PCB中最重要的信息就是进程PID，内核通过这个PID来唯一标识一个进程。PID可以循环使用，最大值是32768。init进程的pid为1，其他进程都是init进程的后代。

除了进程控制块（PCB）以外，每个进程都有独立的内核堆栈（8k），一个进程描述符结构，这些数据都作为进程的控制信息储存在内核空间中；而进程的用户空间主要存储代码和数据。 Linux操作系统使用一些系统调用（如：fork()、wait、exit 等）来实现对进程的管理。

1. C 语言中是如何使用Linux 提供的功能的？用程序及运行结果举例说明。

答：C语言在.c文件中添加调用的.h头文件，调用进程管理所需的相关库函数。以本次实验的部分代码为例：





通过调用getpid()与getppid()函数来在父进程通过循环语句创建若干个子进程后，输出打印每一个进程的进程号，进而可以画出家族数。

1. 什么是进程？如何产生的？举例说明。

答：进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是操作系统动态执行的基本单元，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。

引起进程创建的事件有用户登录、作业调度、提供服务和应用请求。一旦操作系统发现了要求创建新进程的事件后，便调用进程创建原语Creat( )按下述步骤创建一个新进程。申请空白PCB、为新进程分配资源、初始化进程控制块、将新进程插入就绪队列

例如用户登入，在分时系统中，用户在终端键入登录命令后，如果是合法用户，系统将为该终端建立一个进程，并把它插入就绪队列中。

1. 进程控制如何实现的？举例说明。

答：进程控制一般是由OS的内核中的原语来实现的。

例子：唤醒原语wakeup( ) 当被阻塞进程所期待的事件出现时，如I/O完成或其所期待的数据已经到达，则由有关进程(比如用完并释放了该I/O设备的进程)调用唤醒原语wakeup( )，将等待该事件的进程唤醒。

唤醒原语执行的过程是：首先把被阻塞的进程从等待该事件的阻塞队列中移出，将其PCB中的现行状态由阻塞改为就绪，然后再将该PCB插入到就绪队列中。

1. 进程通信方式各有什么特点？用程序及运行结果举例说明。

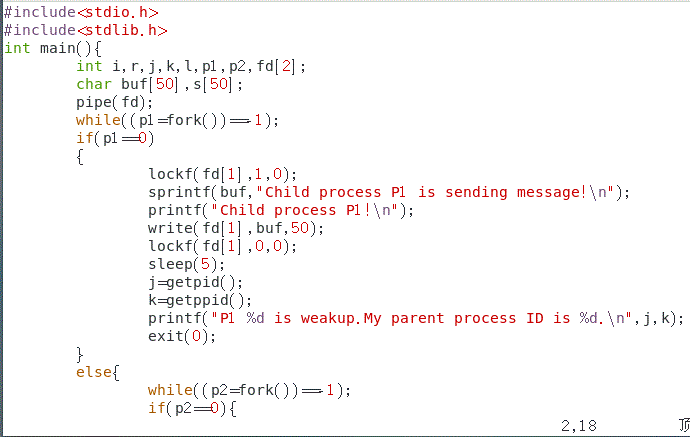
进程间通信可分为4种形式：

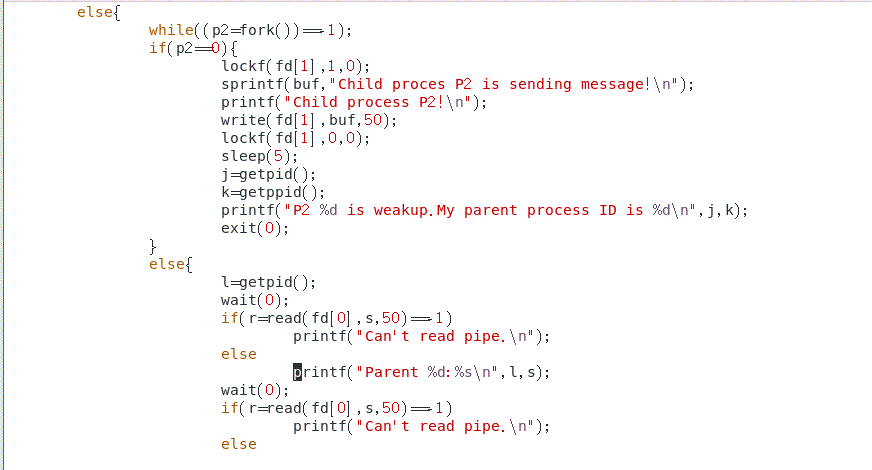
(1) 主从式：① 主进程可自由地使用从进程的资源或数据；② 从进程的动作受主进程的控制；③ 主进程和从进程的关系是固定的。

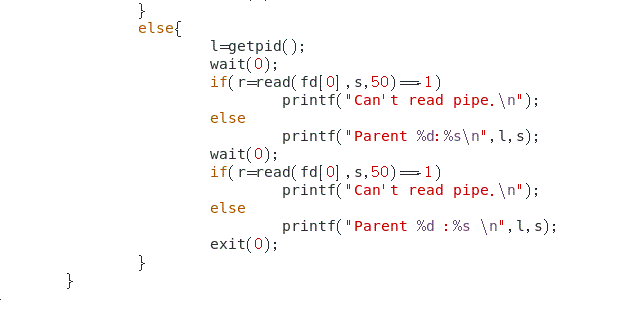
  (2) 会话式：① 使用进程在使用服务进程所提供的服务之前，必须得到服务进程的许可；② 服务进程根据使用进程的要求提供服务，但对所提供服务的控制由服务进程自身完成。③ 使用进程和服务进程在通信时有固定连接关系。

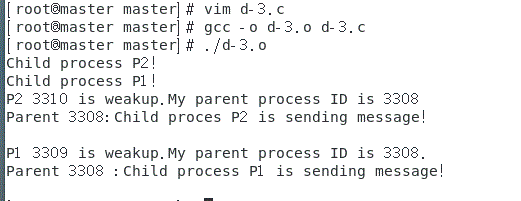
1. 消息或邮箱机制：① 只要存在空缓冲区或邮箱，发送进程就可以发送消息。② 与会话系统不同，发送进程和接收进程之间无直接连接关系，接收进程可能在收到某个发送进程发来的消息之后，又转去接收另一个发送进程发来的消息。③ 发送进程和接收进程之间存在缓冲区或邮箱用来存放被传送消息。
2. 共享存储区方式： ①诸进程可通过对共享存储区中数据的读或写来实现通信。②进程在通信前，先向系统申请获得共享存储区中的一个分区，并指定该分区的关键字；若系统已经给其他进程分配了这样的分区，则将该分区的描述符返回给申请者。③由申请者把获得的共享存储分区连接到本进程上。④可像读、写普通存储器一样地读、写该公用存储分区。

例子：父进程创建两个子进程，父子进程之间利用管道进行通信。









1. 管道通信如何实现？该通信方式可以用在何处？

答：向管道(共享文件)提供输入的发送进程(即写进程)，

以字符流形式将大量的数据送入管道；而接受管道输出的接收进程(即读进程)，则从管道中接收(读)数据。 这种通信方式用于数据传输、资源共享和事件通知。

7. 什么是软中断？软中断信号通信如何实现？

软中断是利用硬件中断的概念，用软件方式进行模拟，实现宏观上的异步执行效果。 利用signal和kill实现软中断通信。

kill(pid,signal): 向进程pid发送信号signal，若pid进程在可中断的优先级（低优先级）上睡眠，则将其唤醒。

signal(sig,ps): 设置sig号软中断信号的处理方式； SIG\_DFL：系统默认方式，一般是终止进程； SIG\_IGN：忽略（屏蔽）；func()： 用制定义函数func()处理。 Signal设臵的处理方式，仅一次有效，处理后即回到默认方式。

**体会：**

在本次进程管理和通信实验中，我实现了用vim文本编译器来编写C语言来实现进程的管理和通信，对操作系统中父进程和子进程的关系有了更深一步的认识。此外，为方便编写程序和相关的调试，我通过vscode上的一个插件remote，通过ssh链接到自己的虚拟机实现在windows上编写linux上的程序，方便调试。

通过对实验中的运行结果进行分析，我也理解了管道通信以及软中断信号通信的原理，同时掌握了一些所调用的库函数的使用。

总而言之，在这次实验中遇到了很多困难，但是凭借自己的努力解决了它们，感觉又成长了一部分。