**操作系统课程设计实验报告**

实验题目：实验三 Linux进程管理

姓 名：张孜远

学 号：20151521

组 号：04

专 业：卓越学院 智能计算与数据科学

班 级：20186211

老师姓名：任彧老师

日 期：2022年12月20日

目 录

[一 题目介绍 1](#_Toc122506112)

[二 实验内容与思路 2](#_Toc122506113)

[三 遇到问题及解决方法 2](#_Toc122506114)

[四 核心代码及实验结果展示 2](#_Toc122506115)

[五 个人实验改进与总结 5](#_Toc122506116)

[5.1 个人实验改进 5](#_Toc122506117)

[5.2 个人实验总结 5](#_Toc122506118)

[六 参考文献 6](#_Toc122506119)

# 一 题目介绍

**实验目的：**

1、通过对Linux进程控制的相关系统调用的编程应用，进一步加深对Linux进程概念的理解，明确进程和程序的联系与区别，理解进程并发执行的具体含义；

2、通过对Linux管道通信机制、消息队列通信机制、共享内存通信机制的应用，加深对不同类型的进程通信方式的理解；

3、通过对Linux的Posix信号量及IPC信号量的应用，加深对信号量同步机制的理解。

**实验内容：**

（1）实现一个模拟的shell：

编写三个不同的程序 cmd1.c，cmd2.c，cmd3.c，每个程序的功能自定，分别编译成可执行文件 cmd1，cmd2，cmd3。然后再编写一个程序，模拟 shell 程序的功能，能根据用户输入的字符串（表示相应的命令名），去为相应的命令创建子进程并让它去执行相应的程序，而父进程则等待子进程结束，然后再等待接收下一条命令。如果接收到的命令为 exit，则父进程结束；如果接收到的命令是无效命令，则显示 Command not found ，继续等待。

（2）实现一个管道通信程序：

由父进程创建一个管道，然后再创建 3 个子进程，并由这三个子进程利用管道与父进程之间进行通信：子进程发送信息，父进程等三个子进程全部发完消息后再接收信息。通信的具体内容可根据自己的需要随意设计，要求能试验阻塞型读写过程中的各种情况，测试管道的默认大小，并且要实现进程间对管道的互斥访问。运行程序，观察各种情况下，进程实际读写的字节数以及进程阻塞唤醒的情况。

（3）利用 linux 的消息队列通信机制实现两个线程间的通信：

编写程序创建三个线程：sender1 线程、sender2 线程和 receive 线程，三个线程的功能描述如下：

① sender1 线程：运行函数 sender1()，它创建一个消息队列，然后，等待用户通过终端输入一串字符，将这串字符通过消息队列发送给 receiver 线程；可循环发送多个消息，直到用户输入 exit 为止，表示它不再发消息，最后向 receiver 线程发送消息 end1 ，并且等待 receiver 的应答，等到应答消息后，将接收到的应答信息显示在终端屏幕上，结束程序的运行。

② sender2 线程：运行函数 sender2()，它创建一个消息队列，然后，等待用户通过终端输入一串字符，将这串字符通过消息队列发送给 receiver 线程；可循环发送多个消息，直到用户输入 exit 为止，表示它不再发消息，最后向 receiver 线程发送消息 end2 ，并且等待 receiver 的应答，等到应答消息后，将接收到的应答信息显示在终端屏幕上，结束程序的运行。

③ receiver 线程运行 receive()，它通过消息队列接收来自 sender1 和 sender2 两个线程的消息，将消息显示在终端屏幕上，当收到内容为 end1 的消息时，就向 sender1 发送一个应答消息 over1 ；当收到内容为 end2 的消息时，就向 sender2 发送一个应答消息 over2 ；消息收完后删除消息队列。使用合适的信号量机制实现三个线程之间的同步与互斥。

（4）利用 linux 的消息队列通信机制实现两个线程间的通信：

编写程序创建两个线程：sender 线程和 receive 线程，其中 sender 线程运行函数 sender()，它创建一个消息队列，然后，循环等待用户通过终端输入一串字符，将这串字符通过消息队列发送给 receiver 线程，直到用户输入 exit 为止；最后，它向 receiver 线程发送消息 end ，并且等待 receiver 的应答，等到应答消息后，将接收到的应答信息显示在终端屏幕上，删除相关消息队列，结束程序的运行。receiver 线程运行 receive()，它通过消息队列接收来自 sender 的消息，将消息显示在终端屏幕上，直至收到内容为 end 的消息为止，此时，它向 sender 发送一个应答消息 over ，结束程序的运行。使用 无名信号量 实现两个线程之间的同步与互斥。

# 二 实验内容与思路

**实验内容：**

1、模拟shell：基本功能 + find、grep命令，并显示相应结果；

2、管道通信：基本功能 + 有名管道通信（独立进程）；

3、共享内存：基本功能 + 双向通信；

# 三 遇到问题及解决方法

**遇到的问题：**

1、实验编译可执行文件失败：

解决方法：编译时使用 gcc xxx.c -pthread指令，添加参数-pthread即可成功生成可执行文件；

2、在线程未创建时，程序已在main函数中结束：

解决方法：添加阻塞进程，调用线程函数pthread\_join；

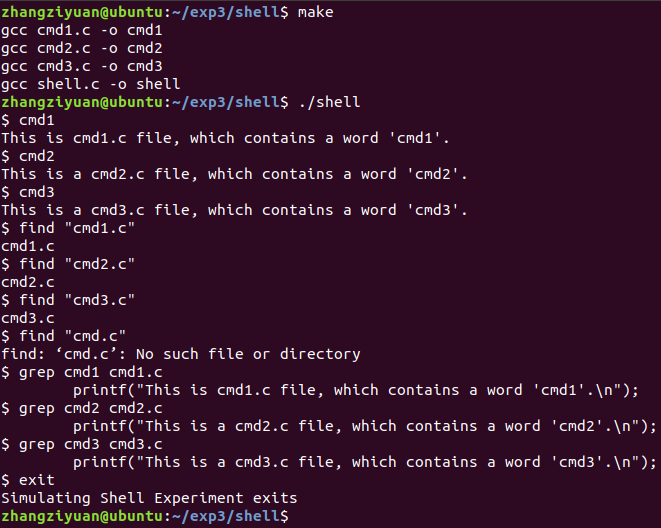
**实验方法：**

在虚拟机中Linux环境下使用C语言进行编程，参照书本与课堂要求完成Linux进程管理中的三个实验。

# 四 核心代码及实验结果展示

**实验过程和结果：**

1. 模拟shell：

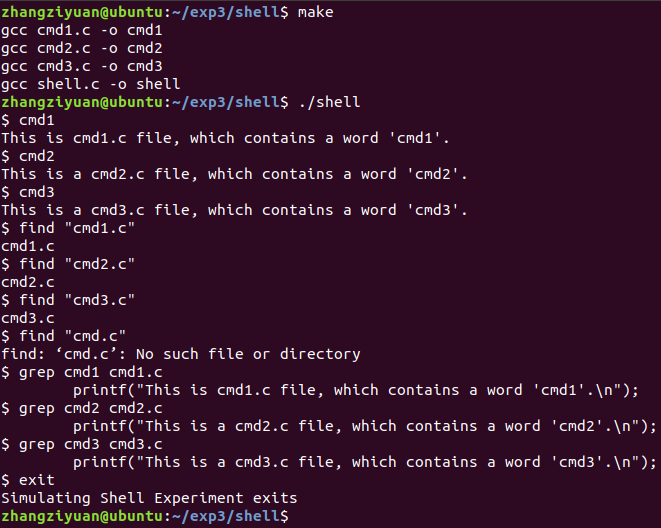
1.1使用make指令，生成cmd1.c、cmd2.c、cmd3.c和shell.c的可执行文件；

1.2使用./shell指令，进入shell.c的可执行文件（类比为main函数），验证功能；

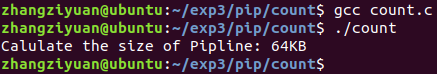
1）分别进入cmd1.c、cmd2.c和cmd3.c的可执行文件cmd1、cmd2和cmd3，正确输出cmd1.c、cmd2.c和cmd3.c中的内容；

2）使用find指令，在当前目录下寻找文件，指令格式为：find “file”；同时如若该文件不存在，则输出报错“find: ‘file’: No such file or dictionary”;

3）使用grep指令，在某个文件中寻找特定字段，指令格式为：grep string file；

 4）使用exit指令，退出模拟shell实验；

1. 管道通信：
2. 统计管道大小：

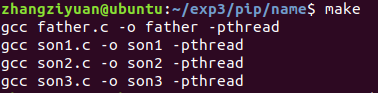
通过非阻塞方式将数据写入管道，每次写1024字节，即1KB，直到管道满为止。

由上图可知，管道大小是64KB；

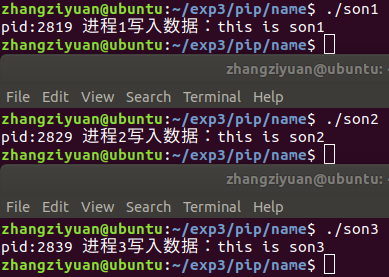
1. 无名管道通信：

通过预编写的pip.c文件，创建三个子进程。分别向进程1、进程2和进程3中写入数据“111”、“222”和“333”，验证父进程接收的数据情况。

1. 有名管道通信：

通过公共的头文件share.h，编写father.c、son1.c、son2.c和son3.c源文件，通过make指令依次生成father、son1、son2和son3可执行文件。

通过./father指令先进入father.c的可执行文件，用于接受来自son1、son2和son3中的数据；通过./son1、./son2和./son3指令依次进入son1.c、son2.c和son3.c的可执行文件，分别在每个子进程中写入数据：this is son1、this is son2和this is son3，观察父进程接收到的数据情况，是否依次为this is son1、this is son2和this is son3。



由上图可知，实验结果正确无误。

1. 共享内存（双向通信）：

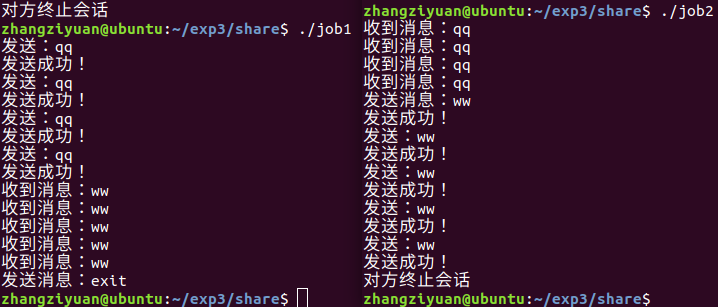
通过make指令依次生成job1.c和job2.c的可执行文件job1和job2。在文件夹下打开两个终端，分别进入可执行文件job1和job2。

先从job1中连续发送消息4次消息字符串“1”，观察job2中接收信息的情况；

再从job2中连续发送消息5次消息字符串“2”，观察job1中接收信息的情况；

最后在job2中输入exit指令，退出“共享”；同时在job1中输出相应提示信息：“对方终止会话”。

由上图可知，实验结果正确无误。

备注：我们也可以在job1中退出“共享”，如下图所示：

# 五 个人实验改进与总结

## 5.1 个人实验改进

|  |  |
| --- | --- |
| 项目实现创新点说明 | 1、根据实验指导书上的要求，以及老师课堂上提出的创新性要求，完成相应的实验工作。 |

## 5.2 个人实验总结

**实验体会：**

相较于之前的操作系统实验，本次实验更加复杂繁琐。我首先补充了大量有关消息队列和信号量机制的原理，牢固自己的理论知识；再通过阅读Linux内核源码，了解了相关Linux内核机制，再开始动手实践。

最开始参考Github上学长的代码，其逻辑是用一个for循环分别创建子进程，但由于其在接收信息后还会再打印一次数据，有bug但我一直没修好，所以后面就打算自己动手实践实现相应功能。

通过此次实验，我对于Linux的管道通信机制和共享内存机制的理解更深了。

管道通信可以轻松地实现两个进程之间的交互，在现实生活中可被用于多个场景（例如智能问答中的人机交互等），但其局限性也十分明显：该管道通信机制只能是单向的；其次如果发送的数据量过大，则会导致管道满和数据丢失的情况。

相较于管道通信机制，共享内存机制可以轻松实现多个进程之间的数据共享，即一个进程向共享内存中写入数据，一个进程向共享内存中读出数据，避免了系统调用的开销，但其局限性也十分明显：共享内存的容量是有限制的；当进程数量变多时，需要通过信号量合理设置同步机制，避免出现冲突的问题。

通过此次实验，我更加充分理解了消息队列和信号量机制的原理，让所学的理论应用于实际，成就感很高。

# 六 参考文献

[1] WEXITSTATUS与WIFEXITED： https://blog.csdn.net/hit\_shaoqi/article/details/53150890

[2] 有关消息队列msgget()、msgsend()、msgrcv()、msgctl()的解释：<https://www.cnblogs.com/52php/p/5862114.html>

[3] 有关stderr()和stdout()的解释：

<https://www.cnblogs.com/mydomain/p/9817320.html>

[4] 有关pthread\_join的解释：

<https://blog.csdn.net/yzy1103203312/article/details/80849831>

[5] 有关sem\_init的解释：

<https://www.cnblogs.com/pipci/p/10179502.html>