# 



**课 程 实 验 报 告**

**课程名称： 操作系统原理**

**专业班级： CS1503班**

**学 号： U201514559**

**姓 名： 周铭昊**

**指导教师： 郑然**

**报告日期： 2018年1月19日**

**计算机科学与技术学院**

**目录**

[**实验一 进程控制 1**](#_Toc502051830)

[**1.1实验目的 1**](#_Toc502051831)

[**1.2 实验内容 1**](#_Toc502051832)

[**1.3 实验过程 1**](#_Toc502051833)

[**1.3.1 编程思路 1**](#_Toc502051834)

[**1.3.2 遇到的问题及解决方式 1**](#_Toc502051835)

[**1.3.3 实验测试与结果分析 2**](#_Toc502051836)

[**1.4 实验总结 2**](#_Toc502051837)

[**实验二 线程同步与通信 3**](#_Toc502051838)

[**2.1 实验目的 3**](#_Toc502051839)

[**2.2 实验内容 3**](#_Toc502051840)

[**2.3 实验过程 3**](#_Toc502051841)

[**2.3.1 编程思路 3**](#_Toc502051842)

[**2.3.2 遇到的问题及解决方式 3**](#_Toc502051843)

[**2.3.3 实验测试与结果分析 3**](#_Toc502051844)

[**2.4 实验总结 3**](#_Toc502051845)

[**实验三 共享内存与进程同步 4**](#_Toc502051846)

[**3.1 实验目的 4**](#_Toc502051847)

[**3.2 实验内容 4**](#_Toc502051848)

[**3.3 实验过程 4**](#_Toc502051849)

[**3.3.1 编程思路 4**](#_Toc502051850)

[**3.3.2 遇到的问题及解决方式 4**](#_Toc502051851)

[**3.3.3 实验测试与结果分析 4**](#_Toc502051852)

[**3.4 实验总结 4**](#_Toc502051853)

[**实验四 Linux文件目录 5**](#_Toc502051854)

[**4.1 实验目的 5**](#_Toc502051855)

[**4.2 实验内容 5**](#_Toc502051856)

[**4.3 实验过程 5**](#_Toc502051857)

[**4.3.1 编程思路 5**](#_Toc502051858)

[**4.3.2 遇到的问题及解决方式 5**](#_Toc502051859)

[**4.3.3 实验测试与结果分析 5**](#_Toc502051860)

[**4.4 实验总结 5**](#_Toc502051861)

# 实验一 进程控制

## **1.1实验目的**

1、加深对进程的理解,进一步认识并发执行的实质；

2、分析进程争用资源现象,学习解决进程互斥的方法；

3、掌握Linux进程基本控制；

4、掌握Linux系统中的软中断和管道通信。

## **1.2 实验内容**

编写程序，演示多进程并发执行和进程软中断、管道通信。

父进程使用系统调用pipe( )建立一个管道,然后使用系统调用fork()创建两个子进程，子进程1和子进程2；

子进程1每隔1秒通过管道向子进程2发送数据:

I send you x times. (x初值为1，每次发送后做加一操作）

子进程2从管道读出信息，并显示在屏幕上。

父进程用系统调用signal()捕捉来自键盘的中断信号（即按Ctrl+C键）；当捕捉到中断信号后，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 Process l is Killed by Parent!

Child Process 2 is Killed by Parent!

父进程等待两个子进程终止后，释放管道并输出如下的信息后终止

Parent Process is Killed!

## **1.3 实验过程**

### 1.3.1 编程思路

要使用管道进行进程间的通信，首先用pipe()函数创建无名管道，pipefd[0]可以看成一个指针，表示管道的读入端，而pipefd [1]则表示管道的写入端。在写进程的时候要通过close()函数关闭读端口，读进程中要关闭写关口，防止出现读写同时进行导致输出错误信息。

然后设置软中断信号，SIGINT表示Ctrl+C的组合键信号，当信号发生后，跳转到两个信号处理函数Int1()、Int2()中，Int1()函数用于传递信号给子进程，Int2()函数用于输出进程结束的信息。

创建两个子进程child1、child2。第一个子进程，向无名管道中写入数据，用sprintf函数可以同时将数据输出到屏幕，然后睡眠1s。第二个子进程，从无名管道中读取数据并输出到屏幕。

最后等待子进程1、2退出，关闭管道。

### 1.3.2 遇到的问题及解决方式

一开始设计的时候，在读进程中没有关闭写管道，写进程中也没有关闭读管道，导致两个进程没有实现同步。解决方式就是在写进程中用close(pipefd[0]);关闭读管道端口，在读进程中用close(pipefd[1]);关闭读管道端口，这样保持一次写入管道，一次读取管道中的数据，两个进程交替执行。

### 1.3.3 实验测试与结果分析

测试结果如图1.1所示，按下ctrl+c后程序结束运行并输出结束信息。程序运行正确。

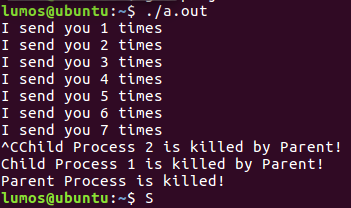


图1.1 实验一测试结果

## **1.4 实验总结**

本次实验学会了如何创建进程，以及通过信号以及信号处理函数的调用来控制进程的软中断，有点像qt中的信号和槽。无名管道就相当于一个缓冲区，一个进程往里面些数据，另一个进程从里面读取数据，两个进程要实现合理的分工同步，就需要一些设计，这里采用的是读进程中不允许写进程执行，两个进程互斥。经过这次实验，对进程间的通信有了进一步的了解。

# 实验二 线程同步与通信

## **2.1 实验目的**

1、掌握Linux下线程的概念；

2、了解Linux线程同步与通信的主要机制；

3、通过信号灯操作实现线程间的同步与互斥。

## **2.2 实验内容**

通过Linux多线程与信号灯机制，设计并实现计算机线程与I/O线程共享缓冲区的同步与通信。

程序要求:两个线程,共享公共变量a

线程1负责计算(1到100的累加，每次加一个数)

线程2负责打印（输出累加的中间结果）

## **2.3 实验过程**

### 2.3.1 编程思路

通过信号灯来控制线程的PV操作，实现线程间的同步和互斥。首先创建2个信号灯，一个初值为0一个初值为1。

再创建两个线程，线程2中，对a进行加1操作，操作前先P(1)，操作结束后再V(0)，表示在进行+1操作后，信号灯1的值从1变为0，信号灯0的值从0变为1。线程1中，对a进行输出，操作前先P(0)，操作结束后再V(1)，表示在进行输出操作后，信号灯0的值从1变为0，信号灯1的值从0变为1。

这样实现两个线程间的同步与互斥，最后等待两个线程运行结束，结束条件为已经从1加到100，然后删除信号灯。

### 2.3.2 遇到的问题及解决方式

本实验中用一个变量a存储从1累加到100的结果，一开始对a赋值为1，结果没有考虑到先执行的是线程2，导致所有输出都大了1，把a的初值改为0，或者修改信号灯使线程1先执行即可。采用的是修改a的初值方法。

### 2.3.3 实验测试与结果分析

程序运行部分截图如图2.1和图2.2所示，程序正确运行。

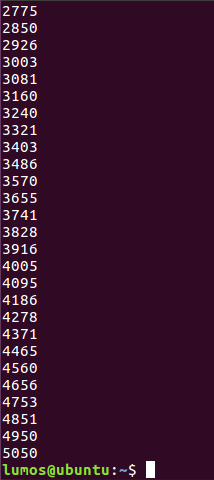
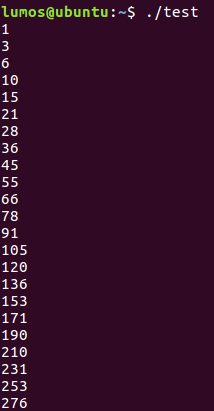


图2.1 图2.2

## **2.4 实验总结**

通过这次实验，我学会了如何使用信号灯来进行PV操作，对PV操作的理解更加深刻。也学会了信号灯的创建使用等操作，在日后实现进程同步互斥时非常实用。

# 实验三 共享内存与进程同步

## **3.1 实验目的**

1、掌握Linux下共享内存的概念与使用方法；

2、掌握环形缓冲的结构与使用方法；

3、掌握Linux下进程同步与通信的主要机制。

## **3.2 实验内容**

利用多个共享内存（有限空间）构成的环形缓冲，将源文件复制到目标文件，实现两个进程的誊抄。

## **3.3 实验过程**

### 3.3.1 编程思路

要实现两个文件的誊抄，建立一个环形缓冲区，在一个进程中把一个文件信息写入缓冲区，另一个进程中把缓冲区中的数据写到新文件中，即可实现两个文件的誊抄。

首先创建环形缓冲区，即为一个共享内存数组shmid[6]，令环形缓冲区大小为6，随后创建两个信号灯，信号灯0的初值为0，信号灯1的初值为6，再创建两个读写进程。

先执行写进程，该进程中pv操作为P(1)、V(0)，表示在写进程中，信号灯1的值从6开始，每次执行就减1，直到信号灯1的值变为0，这样就可以实现一次写入6个数据到环形缓冲中，写完6个数据后，信号灯0的值为6。读进程的pc操作为P(0)、V(1)，信号灯0标识现在缓冲区中有的数据个数，每读取一次信号灯0就-1；信号灯1代表环形缓冲中空闲位置的个数，读取一次信号灯0就+1。这样就实现了两个进程的同步互斥。

### 3.3.2 遇到的问题及解决方式

在从缓冲区读取数据最后写入新文件的时候，一开始没有判断文件结束的条件，解决方法是自己获取要拷贝的文件大小size，设置一个变量i，在一个for语句中将i从1加到size作为进程的执行条件。

### 3.3.3 实验测试与结果分析

程序运行结果如图3.1所示，誊抄后的文件如图3.2所示，先写入6个数据到环形缓冲区中，随后进行读写互斥操作，程序运行正确。

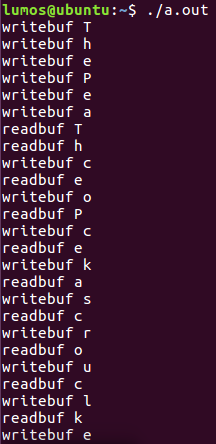


图3.1 实验三运行结果



图3.2 誊抄后的文件

## **3.4 实验总结**

在实验一，用的是无名管道来实现进程间的通信，而本实验中，用的是共享内存来实现，学会了不同的方式进行进程通信。环形缓冲区的实现也提高了进程间通信的效率，对进程通信以及同步互斥有了进一步的了解。

# 实验四 Linux文件目录

## **4.1 实验目的**

1、了解Linux文件系统与目录操作；

2、了解Linux文件系统目录结构；

3、掌握文件和目录的程序设计方法。

## **4.2 实验内容**

1. 功能类似ls -lR；

2. 查询指定目录下的文件及子目录信息；

3. 显示文件的类型、大小、时间等信息；

4. 递归显示子目录中的所有文件信息。

## **4.3 实验过程**

### 4.3.1 编程思路

类似ls -lR输出目标路径下的文件信息，首先输入一个路径，打开这个路径，然后用chdir()函数将输入的路径设置为当前目录，若读取到一个目录项，则以该目录项的名字为参数,调用lstat得到该目录项的相关信息，否则直接判断并输出文件信息，用S\_ISDIR()函数来判断是否为目录文件，类似的，输出信息依次为：

1. 获取并打印文件类型
2. 获取并打印文件所有者的权限
3. 获取并打印与文件所有者同组的用户对该文件的操作权限
4. 获取并打印其他用户对该文件的操作权限
5. 获取并打印文件所有者
6. 获取并打印文件访问者
7. 获取并打印文件大小
8. 获取并打印文件最后修改时间
9. 获取并打印文件名
10. 获取并打印文件深度

### 4.3.2 遇到的问题及解决，方式

输出时间信息时，在结尾自带了一个换行符，导致输出格式有点乱。解决方法是用字符串time保存时间信息，再把最后一个字符‘\0’去除即可。

### 4.3.3 实验测试与结果分析

测试结果如图4.1所示，文件目录如图4.2所示，程序正确运行。

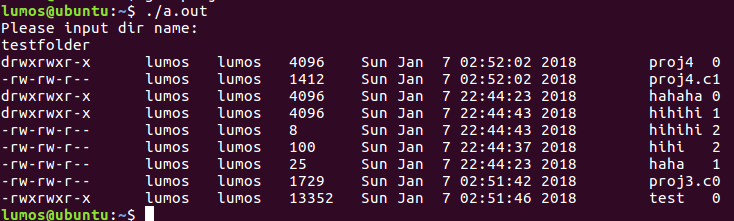


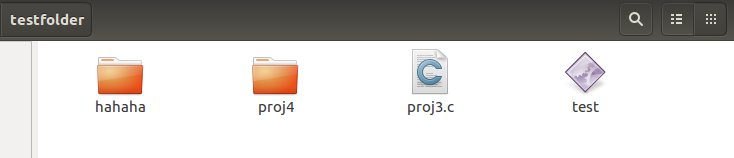
图4.1 实验四测试结果

图4.2 文件目录信息

## **4.4 实验总结**

通过本次实验，我对linux的文件系统更加熟悉，学会了用dir结构体指针来打开一个路径下的目录，也了解了目录项的结构和stat结构体，里面几乎包含了所有的文件状态信息，只要根据需求得到对应子项即可获取文件信息。