《操作系统》思政实验大纲

一、课程基本信息

课程代码：110114

课程名称：操作系统

英文名称:Operation System

实验总学时：12 学时

适用专业：计算机科学专业、软件工程专业、信息管理专业、电子商务专业

课程类别：学科基础课

先修课程：高级程序设计语言、汇编语言、计算机组成原理、数据结构

二、实验教学的总体目的和要求

1、总体目标

毛主席说：“实践是检验真理的唯一标准”，阐明了检验真理的标准只能是社会实践，理论与实践的统一是马克思主义的一个最基本的原则。学习中也是如此，“读万卷书不如行万里路”，而学校教育检验理论知识最有效的方法就是利用教学条件开展实验教学。

通过实验教学，使学生更好地理解操作系统的基本概念、基本原理和实现技术，再以理论指导实践进行深入开发和创新实践。

2、总体要求

（1）对学生的要求

①完成先修课程，尤其是 C 语言；

②实验课前预习，做好知识准备；

③明确实验目的，完成实验内容；

④分析实验结果，总结实验过程；

⑤撰写实验报告，验证理论知识。

（2）对教师的要求

①有一定编程能力，尤其是 C 语言编程；

②熟悉 Linux 的常用命令和系统调用；

③熟悉操作系统工作原理，能够解释实验结果。

（3）对实验条件的要求

Ubuntu 虚拟机、Visual C++ 6.0

三、实验教学内容

《论语·为政》子曰：“温故而知新，可以为师矣。”在新时代的教育背景下，不仅要温故知新，更要推陈出新。因此，通过适当的课后练习不仅可以帮助学生巩固和内化知识，还可以启发学生运用知识解决问题，并培养创新意识和创造能力。

课程配套六个验证型实验，帮助理解抽象的概念和原理。实验一和实验二是文件与目录的常用命令以及文件和目录权限的设置与修改，通过实验帮助学时理解联机命令用户接口的使用以及文件系统的基本操作。实验三是练习 Linux 编辑器 Vi 和 C 语言编译器 Gcc 的使用，为后面的实验做准备。实验四是父进程创建子进程形成进程家族树，通过实验分析各种可能的执行顺序，帮助理解并发执行的概念。实验五是父进程创建子进程和线程，对比分析线程与子进程在资源共享、继承以及运行方面的异同。实验六是进程的高级通信，通过管道通信体会进程高级通信的实现。

	实验项目名称	实验时数	实验要求（必修或选修）	开课周数
实验项目一览	实验准备 虚拟机的安装与启动	0	事先装机，不占学时，用作学习	
	实验一 文件与目录的基本操作	2	必修	2
	实验二 权限的设置与修改	2	必修	4
	实验三 Vi 和 Gcc 的使用	2	必修	6
	实验四 进程的并发执行	2	必修	8
	实验五 进程与线程	2	必修	10
	实验六 进程的高级通信	2	必修	12

实验准备

实验名称：文件与目录的基本操作

实验内容：

1. 安装虚拟机步骤

(1) 需要的软件

Daemon Tools Lite：镜像驱动程序

VMWare Workation：虚拟机软件

Ubuntu Kylin LTS 版：乌班图 Linux

(2) 安装步骤

安装 Daemon Tools Lite

安装 VMWare Workation

安装 Ubuntu

2. Linux 命令的格式

bash 命令的一般格式是：

命令名 [选项] [参数 1] [参数 2] …

示例：cp -f file1.c file2.c

格式说明：

(1) 命令名必须是小写的英文字母；

(2) Linux 的文件名长度不超过 256 个字符，且不能使用如下字符：

! @ # $ ^ & * ( ) [ ] { } ‘ “ 、 / ；

< > 空格

(3) Linux 文件名区分大小写；文件名中如有多个圆点，则最右边一个为分隔符，且多表示文件类型；

(4) Linux 使用“/”为根目录、目录分隔符和目录与文件的分隔符；

(5) 命令中通配符的使用

* 代表任意个任意字符

? 代表一个任意字符

[ ] 只要文件名中[ ]位置处的字符在[ ]中指定的范围内，那么该文件名就与给定的模式相匹配。[!]是对[ ]中内容的排除，不在这个范围就匹配。

3. 超级用户与普通用户

(1) 超级用户是对系统的一切资源均具有访问权限的用户，即系统管理员，用户名为：root，命令提示符为：#；

(2) 普通用户由超级用户创建和删除，仅具有超级用户指定的访问权限，命令提示符为：$。

(3) 超级用户和普通用户可以相互切换；要养成用普通用户登录、使用系统的习惯。

实验性质：综合性

实验学时：0

实验目的与要求：

1. 掌握虚拟机的安装；
2. 了解超级用户和普通用户的切换；
3. 掌握用户的创建和删除；
4. 掌握系统的启动和退出。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1. 尝试安装自己的虚拟机;
2. 创建和删除自己的用户；
3. 熟练登录与退出系统。

理论与实践的辩证思考：

1、实践决定理论，实践是理论的来源，是理论发展的根本动力、是理论的最终目的、是检验真理的唯一标准。

2、理论对实践有能动的反作用。理论产生的最终目的是为了更好地指导实践，真理和科学理论对实践有巨大的推动作用。

3、理论和实践是相辅相成的，缺一不可的，不能任意割裂两者的辩证关系，孤立地强调一个方面。

4、要坚持理论与实践相结合。必须坚持理论和实践的具体的、历史的统一。

实验项目一

实验名称：文件与目录的基本操作

实验内容：

1. 目录操作

（1）查看当前目录 $pwd

（2）显示目录内容 ls［选项］［目录或文件］

（3）改变目录 cd［路径］

（4）创建目录 mkdir［选项］目录名

（5）删除空目录　rmdir［选项］目录名

2. 文件操作

（1）创建空文档 touch［选项］文件名

（2）查看文件内容 cat 文件名

（3）分页查看文件内容 more［选项］文件名 / less［选项］文件名

（4）复制文件 cp［选项］源文件路径名 目标文件路径名

（5）移动文件 mv［选项］源文件路径名 目标文件路径名

（6）删除文件 rm［选项］文件名|目录名

3. 查看系统目录和文件

（1）查看/bin 子目录内容

（2）查看/home 子目录内容

（3）查看/etc 子目录下的文件

（4）查看环境变量设置

$gedit /etc/environment　（用 gedit 显示全局环境变量）

$gedit /etc/profile　（用 gedit 显示登录用户环境设置）

$gedit ~/profile　（用 gedit 显示当前用户环境设置）

（5）认识环境变量 $PATH

在 PATH 中设置命令的搜索路径，在任何目录都可以执行常用命令。

4. 文件的其他操作

（1）查找文件　find 目录名［选项］

（2）按指定模式查找文件　grep［选项］字符串 文件列表

（3）对指定文件按行排序　sort［选项］文件名

5. 群命令及续行符

实验性质：验证性

实验学时：2

实验目的与要求：

1.了解 Linux 文件系统的结构和功能；

2.了解 Linux 文件系统的访问方式；

3.掌握常用的文件和目录操作命令；

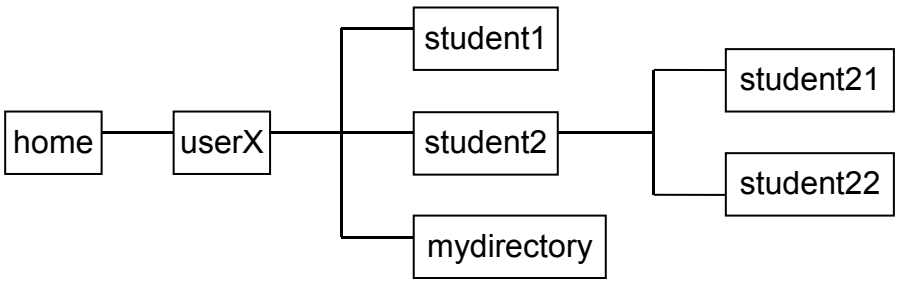
4.熟悉字符界面的联机命令工作方式。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1.Linux 的文件组织与 Windows 有什么不同？

2.创建如下的工作目录，然后尝试删除 student1 和 student2。



3.在用户主目录下创建 myfiel1, myfile2, myfile3 三个文件，总结创建文件的几种方法。

4.总结查看文本文件内容的几种方法。

5.在自己的目录下进行显示、复制、移动、删除等基本操作。

6.了解 Linux 群命令的执行。

实验项目二

实验名称：权限的设置与修改

实验内容：

1. 权限说明

(1) 独立的权限级别

文件拥有者（u—user）

所在组成员（g—group）

其他用户（o—others）

(2) 文件权限

r(read)：查看文件内容

w(write)：修改文件

x(execute)：运行可执行文件

(3) 目录权限

r(read)：列出目录内容

w(write)：在目录中增删文件

x(execute)：进入并访问目录中的文件

2. 文件属性

（1）文件类型

-：普通文件

d：目录

l：符号链接

c：字符型设备节点

b：块设备节点

（2）文件访问权限（3 级独立权限）

（3）文件的硬链接数

（4）文件拥有者的用户名

（5）文件所属用户组名

（6）文件的字符数

（7）文件创建的日期和时间

（8）文件的路径名

3.具体操作

（1）查看文件权限 ls -l

（2）修改文件更改时间 touch -t [时间][文件名]

（3）创建文件链接 ln [-s] 旧文件名 新文件名

硬链接 $ln ./mydirectory/hello my_hello

符号链接 $ln -s ./mydirectory/hello my_symbol

两者的区别 $rm -f ./mydirectory/hello

$cat my_hello 可用硬链接 my_hello 显示其内容

$cat my_symbol 不能显示符号链接 my_symbol 内容

（4）修改文件或目录权限

格式一： chmod who operator permission 文件名/目录名

who 的含义：

u：文件属主权限

g：同组用户权限

o：其他用户权限

a：所有用户权限

operator 的含义：

+：增加权限

-：取消权限

=：设定权限

permission 的含义：

r：读权限

w：写权限

x：执行权限

s：文件属主和组 set-ID

t：粘性位

l：给文件加锁，使其他用户无法访问

格式二： chmod ××× 文件名/目录名

r--：4—读 -w-：2—写 --x：1—执行

rw-：4+2=6 r-x：4+1=5 rwx：4+2+1=7

（5）改变文件或目录所属组　chgrp [选项] 组 文件

（6）改变文件的属主或属组　chown [选项] 用户或属组 文件

实验性质：验证性

实验学时：2

实验目的与要求：

1. 理解文件和目录权限的概念；

2. 掌握权限的设置和修改。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1.（1）在主目录下创建工作目录 mywork，将/usr/bin/cal 以文件名 mycal、将/bin/cat 以文件名 mycat 均复制到 mywork 下，分别查看文件 cal 和 cat 的权限，以及文件 mycal 和 mycat 的权限，有什么不同？

（2）分别运行 cal 和 mycal，是什么结果？思考原因。

（3）要使文件 mycal 运行，应该怎么做？

2.（1）在主目录下用 cat 命令创建内容为“Hello,World!”的文本文件 hello，然后在子目录 mywork 下创建名为 my_hello 的硬链接，以及名为 my_symbol 的符号链接，分别显示这两个链接的内容；

（2）修改主目录下的文件 hello 的内容，再次显示这两个链接的内容，有什么不同？

(3) 然后删除主目录下的文件 hello，再次显示这两个链接，结果又有什么不同？

3. (1) 在主目录下，显示文件 hello 的权限；

(2) 去掉文件属主的 r 权限，可以显示其内容吗？添加 r 权限再试试；

(3) 去掉文件属主的 w 权限，可以修改其内容吗？添加 w 权限再试试。

4. (1) 在主目录下，显示子目录 mywork 的权限；

(2) 去掉文件属主的 x 权限，能否进入该子目录？加上再试试；

(3) 去掉文件属主的 r 权限，能否显示该子目录的内容？加上再试试；

(4) 去掉文件属主的 w 权限，能否在该子目录下创建或删除文件？加上再试试。

5. 尝试用格式二的方法为文件或子目录设置权限。

实验项目三

实验名称：Vi 编辑器和 Gcc 编译器的使用

实验内容：

1. 在当前目录下创建一个 exercise 子目录，并切换到该目录；

2. 用 vi 输入一个名为 hello.c 的源程序

```
#include <stdio.h>
int main ( )
{   printf("Hello world,Linux programming!\n\n");
    return 0;
}
```

3.练习命令模式、插入模式和底线模式的常用命令，存盘退出；

4.再次进入 hello.c，将源程序修改为以下代码

```
#include <stdio.h>
int main ( )
{   char name[10];
    printf("please input a name:\n");
    scanf("%s",name);
    printf("%s%s","Hello ",name);
    printf("\n");
    return 0;
}
```

5.以文件名 nihao.c 存盘退出，用 ls 查看文件信息

6. gcc 的用法 gcc［选项］［文件名］

选项说明：

-E：仅对输入文件进行预处理

-c：把源代码编译为目标文件，不执行汇编和链接的过程

-s：编译产生汇编语言文件后停止编译

-o：编译并直接产生可执行文件

(1) 分四个阶段对该程序进行编译

$gcc -E hello.c -o hello.i

(将 hello.c 的预处理结果保存在 hello.i 中)

$gcc -c hello.i -o hello.o

(将预处理结果 hello.i 编译为目标文件 hello.o)

$gcc -s hello.i -o hello.s

(也可将预处理结果 hello.i 编译为汇编文件 hello.s)

$gcc hello.o -o hello

(将目标文件 hello.o 编译为可执行文件 hello)

(2)运行可执行文件 $./hello (运行可执行文件 hello)

实验性质：验证性

实验学时：2

实验目的与要求：

1. 理解文件和目录权限的概念；
2. 掌握权限的设置和修改。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1.Vi 有几种工作模式？掌握各种模式下的基本操作；

2.如何用 Gcc 实现源程序的编译和调试？

3.如何运行可执行程序？

实验项目四

实验名称：进程的并发执行

实验内容：

(1) 进程的资源继承

对比分析以下两段代码的执行结果和原因。

```
#include<unistd.h>
#include <stdio.h>
void    main()
{ printf("Before fork...");
   if (fork()==0)
      {    printf("Son\n");    }
   else
      {    printf("Father\n");    }
}//A
```

```
#include<unistd.h>
#include <stdio.h>
void    main()
{ printf("Before fork...\n");
   if (fork()==0)
      {    printf("Son\n");    }
   else
      {    printf("Father\n");    }
}//B
```

2.进程的并发执行

编写一段程序，利用 fork（）创建两个子进程，其中父进程显示字符“AAA”，两个子进程分别显示字符“BBB”和“CCC”。观察和记录屏幕上的显示结果，并分析原因。

```
#include<unistd.h>
#include<stdio.h>
main()
{    int p1,p2;
   while ((p1=fork())= =-1);                    /*创建子进程 p1*/
   if (p1= =0)                                  /*子进程 1 创建成功*/
      printf(“BBB\t”);
   else
   {    while ((p2=fork())= =-1);               /*创建子进程 p2*/
        if (p2= =0)                             /*子进程 2 创建成功*/
           printf(“CCC\t”);
        else
           printf(“AAA\t”);    }                    /*父进程执行*/
}//abc
```

3. 进程的并发控制

对比以下两段代码，分析可能的输出结果和原因。

```
#include<unistd.h>
#include<stdio.h>
main()
{   int p1,p2;
  while ((p1=fork())= =-1);                    /*创建子进程 p1*/
  if (p1= =0)                                   /*子进程 1 创建成功*/
     for (i=0;i<50;i++)
        printf("son%d\n",i);                   /*输出 50 个字符串"son"*/
  else
   { while ((p2=fork())= =-1);                  /*创建子进程 p2*/
        if (p2= =0)                              /*子进程 2 创建成功*/
           for (i=0;i<50;i++)
              printf("daughter%d\n",i);        /*输出 50 个字符串"daughter"*/
        else      for (i=0;i<50;i++)                 /*父进程执行*/
                     printf("children%d\n",i);       /*输出 50 个字符串"children"*/
   }   }//A
```

```
#include<unistd.h>
#include<stdio.h>
main()
{   int p1,p2,i;
   while ((p1=fork())= =-1);                    /*创建子进程 p1*/
   if (p1= =0)                                   /*子进程 1 创建成功*/
   {   lockf(1,1,0);
       for (i=0;i<50;i++)   printf("son%d\n",i);
       lockf(1,0,0);   }
   else
   {   while ((p2=fork())= =-1);                  /*创建子进程 p2*/
     if (p2= =0)                                  /*子进程 2 创建成功*/
     {   lockf(1,1,0);
        for (i=0;i<50;i++)   printf("daughter%d\n",i);
        lockf(1,0,0);   }
     else   {   lockf(1,1,0);
                for (i=0;i<50;i++)   printf("children%d\n",i);
                lockf(1,0,0);   }
    }   }//B
```

4.进程的多次创建

对比分析以下两段代码，深入理解进程的循环创建与并发执行。

```
#include<unistd.h>
#include<stdio.h>
void    main( )
{    int i,pid;
     for    (i=1;i<=3;++i)
        { pid=fork( );
          if (pid>0)
             printf(“test\n”);
          else
             { printf(“test\n”);
               exit(0);    }
        }
}//A
```

```
#include<unistd.h>
#include<stdio.h>
void    main( )
{    int i,pid;
     for    (i=1;i<=3;++i)
     {    pid=fork( );
       if (pid>0)
          printf(“test\n”);
       else
          printf(“test\n”);
  }
}//B
```

实验性质：验证性

实验学时：2

实验目的与要求：

1．理解文件和目录权限的概念；

2．掌握权限的设置和修改。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1.父进程如何创建子进程？

2.子进程的执行入口在哪里？

3．父进程和子进程的并发执行是什么含义？

4．并发进程竞争资源如何进行控制？

实验项目五

实验名称：认识进程和线程

实验内容：

1．用 fork( )创建进程

fork 系统调用创建的子进程复制了父亲进程的资源，包括内存的内容 task_struct 内容（两个进程的 pid 不同）。这里是资源的复制不是指针的复制。

```
//testFork.c
#include"stdio.h"
int main()
{ int count = 1;
   int child;
   if(!(child = fork()))
    { //开始创建子进程
      printf("This is son, his count is: %d. and his pid is: %d\n", ++count, getpid());
                                  //子进程的内容                    }
   else
    { printf("This is father, his count is: %d, his pid is: %d\n", count, getpid());
    }
}
```

显示结果:

This is son, his count is: 2. and his pid is: 302

This is father, his count is: 1, his pid is: 301

2．用 vfork( )创建线程

vfork 系统调用创建出来的是一个线程，因为它缺少了自己独立的内存资源。

```
//testVfork.c
#include "stdio.h"
int main( )
{ int count = 1;
   int child;
   printf("Before create son, the father's count is:%d\n", count);
   if(!(child = vfork()))
    { printf("This is son, his pid is: %d and the count is: %d\n", getpid(), ++count);
     exit(1);
     }
    else
     { printf("After son, This is father, his pid is: %d and the count is: %d,
              and the child is: %d\n", getpid(), count, child);
     }  }
```

显示结果：

Before create son, the father's count is:1

This is son, his pid is: 4185 and the count is: 2

After son, This is father, his pid is: 4184 and the count is: 2, and the child is: 4185

3．用 vfork( )创建线程和父进程的同步问题

由 vfork 创造出来的子进程还会导致父进程挂起，除非子进程执行 exit 或者 execve 才会唤起父进程。看下面程序：

```
//testVfork.c
#include "stdio.h"
int main()
 { int count = 1;
   int child;
   printf("Before create son, the father's count is:%d\n", count);
   if(!(child = vfork()))
    { int i;
     for(i = 0; i < 100; i++)
      { printf("This is son, The i is: %d\n", i);
       if(i == 70) exit(1);        }
     printf("This is son, his pid is: %d and the count is: %d\n", getpid(), ++count);
     exit(1);        }
    else
     { printf("After son, This is father, his pid is: %d and the count is: %d,
              and the child is: %d\n", getpid(), count, child);       }
 }
```

实验学时：2

实验目的与要求：

1．理解文件和目录权限的概念；

2．掌握权限的设置和修改。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1. 子进程和线程在资源继承方面有何不同？

2. 子进程、线程和父进程的执行顺序如何控制？

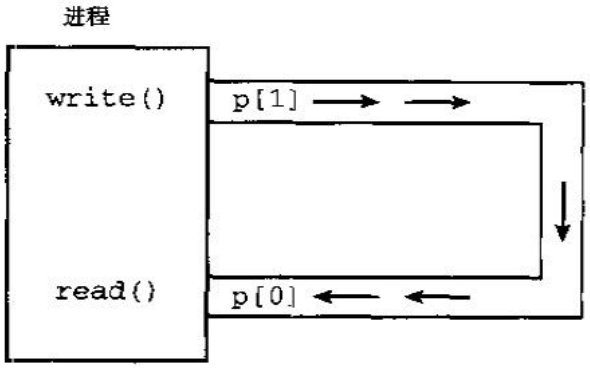
实验项目六

实验名称：进程的高级通信

实验内容：

1. 进程内的串行通信

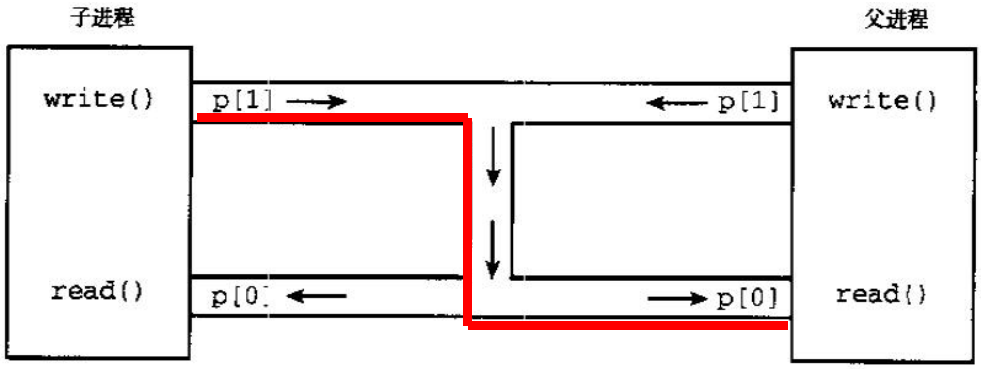
编制程序，利用管道发送三条消息，然后顺序输出。




```
#include<unistd.h>
# include<signal.h>
#include<stdio.h>
#define MSGSIZE 16
char *msg1 = "hello,world #1";
char *msg2 = "hello,world #2";
char *msg3 = "hello,world #3";
void main()
{ char inbuf[MSGSIZE];
   int    fd[2],j;
   if(pipe(fd) == -1)
   { printf("pipe call");        exit(1); }
   write(fd[1],msg1,MSGSIZE);
   write(fd[1],msg2,MSGSIZE);
   write(fd[1],msg3,MSGSIZE);
   for(j=0;j<3;j++)
   { read(p[0],inbuf,MSGSIZE);
      printf("%s\n",inbuf);    }
   exit(0);
}//main
```

2. 进程间的管道通信

编写程序，在父进程中创建一个子进程，子进程向父进程创建的管道中写入消息，父进程从管道中读出消息。



［提示］管道其实是强制用做单向的通信信道，而它本身是允许双向通信的。如果两个进程都同时读写管道，结果将会发生混淆。

```
#include<unistd.h>
#include <stdio.h>
void main( )
{ int   pid  , fd[2]；
 char   buf1[20]，  buf2[20]；
 pipe(fd)；
 while   ((pid=fork())==-1)；
 if   (pid==0)
   { sprintf(buf1,”This is an example\n”) ;
    write(fd[1],buf1,50)；
    exit(0)；  }//if
 else
    { wait(0);
     read(fd[0],buf2,50);
     printf(“%s”,buf2); }//else
}//main
```

3. 进程间的管道互斥

编制程序，父进程创建两个子进程，两个子进程分别向管道写入消息，父进程依次从管道读出消息。为了使两个子进程写入的消息不至于混淆，要实现对管道写入端的互斥访问。

```
# include<unistd.h>
# include<signal.h>
# include<stdio.h>
main( )
{   int fd[2],pid1,pid2;
   char OutPipe[100],InPipe[100];
   pipe(fd);                                          /*创建管道*/
   while ((pid1=fork( ))= =-1);                       /*创建子进程 1*/
   if (pid1= =0)                                      /*子进程 1 创建成功*/
   { lockf(fd[1],1,0);                                /*锁定管道写入端*/
      sprintf(OutPipe,"Child1 is sending message!"); /*定义发送缓冲区*/
      write(fd[1],OutPipe,50);                        /*写入管道*/
      sleep(5);
      lockf(fd[1],0,0);                               /*释放管道写入端*/
      exit(0);   }                                    /*子进程终止*/
   else
   { while ((pid2=fork( ))= =-1);
      if (pid2= =0)
      {   lockf(fd[1],1,0);
         sprintf(OutPipe,"Child2 is sending message!");
         write(fd[1],OutPipe,50);
         sleep(5);
         lockf(fd[1],0,0);
         exit(0);       }
      else
      { wait(0);                                      /*等待子进程终止*/
         read(fd[0],InPipe,50);      /*从管道读出端读消息到接收缓冲区*/
         printf("%s\n",InPipe);                       /*输出读到的消息*/
         wait(0);
         read(fd[0],InPipe,50);
         printf("%s\n",InPipe);
         exit(0);   }
   }//else
}//main
```

实验性质：验证性

实验学时：2

实验目的与要求：

1．理解文件和目录权限的概念；

2．掌握权限的设置和修改。

实验条件：Ubuntu 虚拟机、Visual C++ 6.0

研究与思考：

1.进程通信的本质是什么问题？

2.管道通信时如何实现的？

3.如何实现管道通信的同步与互斥？

四、考核方式

撰写实验报告,实验报告成绩占平时成绩的 40%。

五、推荐实验教材和教学参考书

实验教材：自编教材

参考教材：胡峰松.操作系统原理实验教程(基于 Linux)（第 1 版）.北京：清华大学出版社，2010 年

六、其他需说明的

大纲修订人：白雪梅　　修订日期：2020 年 11 月

大纲审定者：沈永珞　　审定日期：2020 年 12 月