广东开放大学

计算机科学与技术专业

《操作系统原理与应用》

课程实验方案

人工智能学院

2018/9/10

目录

一、课程基本信息

学院：人工智能学院

专业名称：计算机科学与技术

课程名称：操作系统原理与应用

课程编码（课程 ID 号）：10096

课程性质（专业必修课 / 专业选修课）：专业必修课

适用专业：计算机类

先修课程：计算机导论、程序设计语言、数据结构

课程实验（实训）负责教师：肖小红

课程总学分：3

课程总学时：54

课程实验（训）总学时：18

二、课程简介

《操作系统原理与应用》课程是计算机和信息类专业的重要的核心课程，操作系统是计算机系统中最主要的系统软件，是计算机软硬件的管理者，其基本原理和算法是计算机及信息类专业人员必备的知识体系。通过本课程的学习，可以使学生了解计算机操作系统的一些基本术语、概念，掌握计算机操作系统的功能，了解操作系统的工作原理及典型算法，从而进一步提高学生对计算机软硬件系统的认识，

为后续课程的学习和将来的开发工作打下坚实的基础。

三、考核说明或要求

本课程共有 6 个实训，见下表。

实验名称	课时	建议分数
实验一：Linux 基本操作	2	10
实验二：进程控制	4	15
实验三：进程间通信	2	15
实验四：处理器（进程）调度	2	15
实验五：存储管理	2	15
实验六：文件系统设计▲	6	30

四、实验名称：

实验一：Linux 基本操作

预计课时：2 学时

实训时长： 40(分钟)

实验简介： 本实验主要是熟悉 linux 系统中的常见命令以及 vi 编辑器的使用，能够利用 vi 编辑器编辑 C 语言程序，并进行编译和运行。

实验目标：

（1）熟悉 Linux 下的基本操作，学会使用各种 Shell 命令去操作 Linux，对 Linux 有一个感性认识。

（2）学会使用 vi 编辑器编辑简单的 C 语言程序，并能对其编译和调试。

实验内容：

（1）以 root 用户身份登陆，并使用“ls”，“cat”“cd”等命令来实现基本的文件操作并观察 Linux 文件系统的特点；

（2）使用 vi 编辑器编写一 C 程序，并用 gcc 命令进行编译和链接，并用 a.out 来进行输出结果。

实验所需基础：

操作系统：Linux RHEL 6.0

实验是否需要联网：否

实训步骤：

步骤一：Linux 常用命令练习

（1）用 root 账号（超级用户）注册，口令为 computer（注意大小写)。注册成功出现#号（超级用户系统提示符,普通用户的系统提示符为$)。

（2）注销（退出）系统：logout 或 exit

（3）练习使用命令 ls（注意 Linux 命令区分大小写。）

使用 ls 查看当前目录内容；使用 ls 查看指定目录内容，如/目录，/etc 目录

使用 ls –all 查看当前目录内容；使用 dir 查看当前目录内容

（4）使用 cd 改变当前目录

cd .. 回到上层目录 ；cd / 回到根目录

（5）pwd　显示当前路径

（6）建立目录 mkdir

mkdir　目录名 ；　mkdir　/home/s2001/newdir

（7）删除目录：rmdir;

（8）复制文件 cp：　如　cp　文件名 1　文件名 2

（9）移动文件或目录：mv

（10）删除文件　rm

（11）　显示文件内容：more　(分页显示);

（12）显示文件：cat 文件名　建立文件：cat >文件名，ctrl+d 结束输入

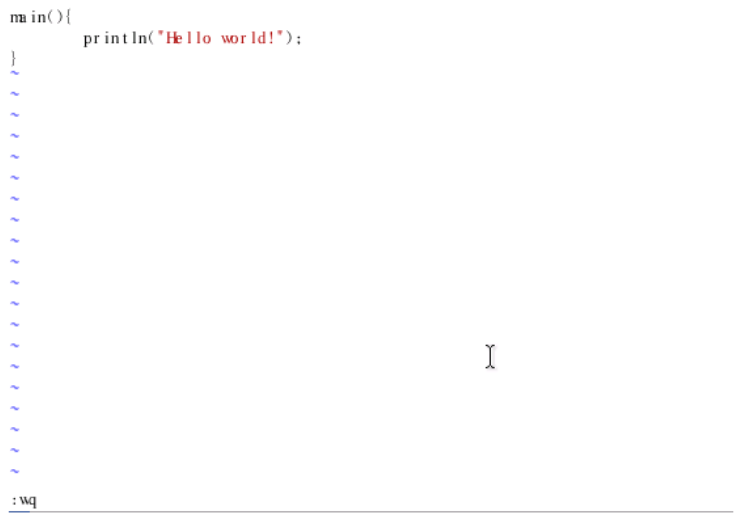
步骤二：使用编辑器 vi 编辑文件

（1）进入 linux 的文本模式之后，在命令行键入 vi filename.c 然后回车。下面作一些简单的解释：首先 vi 命令是打开 vi 编辑器。后面的 filename.c 是用户即将编辑的 c 文件名字，注意扩展名字是.c；当然，vi 编辑器功能很强，可以用它来编辑其它格式的文件，比如汇编文件，其扩展名字是.s；也可以直接用 vi 打开一个新的未命名的文件，当保存的时候再给它命名，只是这样做不很方便。

（2）最基本的命令 I ：当进入刚打开的文件时，不能写入信息，这时按一下键盘上的 I 键（insert），插入的意思，就可以进入编辑模式了。如下图所示：

（3）a 与 i 是相同的用法

（4）当文件编辑完后，需要保存退出，这时需要经过以下几个步骤：1）按一下键盘上的 Esc 键；2）键入冒号(:)，紧跟在冒号后面是 wq（意思是保存并退出)。如果不想保存退出，则在第二步键入冒号之后，键入 q!（不带 w，机尾部保存)。如下图所示：



（5）退出 vi 编辑器的编辑模式之后，要对刚才编写的程序进行编译。编译的命令是：gcc filename.c [-o outputfilename.out]，其中 gcc 是 c 的编译器。参数：filename.c 是要编译的源文件的名称，outputfilename 表示输出文件名称，中括号表示括号内部的内

容可输入也可以不输入（中括号本身不再命令行中出现）。

（6）最后一步是运行程序，方法如下：./outputfilename

实验二：进程控制

预计课时：4 学时

实训时长：80(分钟)

实验简介：本次实验主要理解程序和进程的关系，进程的创建，多进程的运行以及同步互斥的控制。

实验目标：

（1）加深对进程概念的理解，明确进程和程序的区别。

（2）进一步认识并发执行的实质。

（3）分析进程竞争资源现象，学习解决进程互斥的方法。

实验内容：

（1）进程的创建

编写一段源程序，使系统调用 fork()创建两个子进程，当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符：父进程显示字符“a”;子进程分别显示字符“b”和字符“c”。试观察纪录屏幕上的显示结果，并分析原因。

（2）进程的控制

修改已编写的程序,将每个进程输出一个字符改为每个进程输出一句话，在观察程序执行时屏幕出现的现象，并分析原因。

如果在程序中使用调用 lockf()来给每一个子进程加锁，可以实现进程之间的互斥，观察并分析出现的现象。

（3）①编写一段程序，使其现实进程的软中断通信。

要求：使用系统调用 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 DEL 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 Processll is Killed by Parent!

Child Processl2 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在上面的程序中增加语句 signal (SIGNAL, SIG-IGN) 和 signal (SIGQUIT, SIG-IGN)，观察执行结果，并分析原因。

（4）进程的管道通信

编制一段程序，实现进程的管理通信。

使用系统调用 pipe()建立一条管道线；两个子进程 P1 和 P2 分别向管道中写一句话：

Child 1 is sending a message!

Child 2 is sending a message!

而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。

要求父进程先接收子进程 P1 发来的消息，然后再接收子进程 P2 发来的消息。

实验所需基础：

操作系统：Linux RHEL 6.0

实验是否需要联网：否

实训步骤：

步骤一：进程的创建

编写一段程序，使用系统调用 fork( )创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符；父进程显示字符“a”，子进程分别显示字符“b”和“c”。试观察记录屏幕上的显示结果，并分析原因。

〈程序〉

```
#include<stdio.h>
main()
{
int p1,p2;
while((p1=fork())==-1);
if(p1==0)                /*子进程创建成功*/
       putchar('b');
else
 {
      while((p2=fork())==-1);
    if(p2==0)              /*子进程创建成功*/
         putchar('c');
      else
        putchar('a');           /*父进程执行*/
}
}
```

步骤二：进程的控制

修改已编写好的程序，将每个程序的输出由单个字符改为一句话，再观察程序执行时屏幕上出现的现象，并分析其原因。如果在程序中使用系统调用 lockf()来给每个程序加锁，可以实现进程之间的互斥，观察并分析出现的现象。

〈程序 1〉

```
#include<stdio.h>
```

```
main()
{
 int p1,p2,i;
 while((p1=fork())==-1);
   if(p1==0)
     for(i=0;i<50000;i++)
       printf("child %d\n",i);
   else
        {

        while((p2=fork())==-1);
        if(p2==0)
          for(i=0;i<50000;i++)
              printf("son %d\n",i);
        else
          for(i=0;i<50000;i++)
              printf("daughter %d\n",i);
}
}
```

〈程序 2〉

```
include<stdio.h>
main()
{
 int p1,p2,i;
 while((p1=fork())==-1);
   if(p1==0)
     {
       lockf(1,1,0);
       for(i=0;i<50000;i++)
       printf("child %d\n",i);
       lockf(1,0,0);
    }
   else
        {

        while((p2=fork())==-1);
        if(p2==0)
                    {
          lockf(1,1,0);
         for(i=0;i<50000;i++)
              printf("son %d\n",i);
          lockf(1,0,0);
                    }
```

```
        else
                    {
          lockf(1,1,0);
         for(i=0;i<50000;i++)
             printf("daughter %d\n",i);
           lockf(1,0,0);
                 }
}
}
```

比较<程序 1>和<程序 2>的运行结果，分析 lockf()函数的作用。

步骤三：软中断通信

编制一段程序，使用系统调用 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 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 process1 is killed by parent!

child process2 is killed by parent!

父进程等待两个子进程终止后，输出以下信息后终止：

parent process is killed!

<程序流程图>

〈程序〉

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>

void waiting(),stop(),alarming();
int wait_mark;

main()
{
 int p1,p2;
 if(p1=fork())              /*创建子进程 p1*/
```

```
{
    if(p2=fork())          /*创建子进程 p2*/
    {
        wait_mark=1;
        signal(SIGINT,stop);     /*接收到^c 信号，转 stop*/
        signal(SIGALRM,alarming);/*接受 SIGALRM
        waiting();

        kill(p1,16);              /*向 p1 发软中断信号 16*/
            kill(p2,17);           /*向 p2 发软中断信号 17*/

            wait(0);               /*同步*/
        wait(0);
        printf("parent process is killed!\n");
        exit(0);
    }
    else
    {
        wait_mark=1;
        signal(17,stop);
        signal(SIGINT,SIG_IGN);  /*忽略 ^c 信号*/
        while (wait_mark!=0);
        lockf(1,1,0);
        printf("child process2  is killed by parent!\n");
        lockf(1,0,0);
        exit(0);
    }
}
else
{
    wait_mark=1;
    signal(16,stop);
signal(SIGINT,SIG_IGN);  /*忽略^c 信号*/
    while (wait_mark!=0)
    lockf(1,1,0);
    printf("child process1 is killed by parent!\n");
    lockf(1,0,0);
    exit(0);
}
}

void waiting()
{
sleep(5);
```

```
if (wait_mark!=0)
 kill(getpid(),SIGALRM);

}
void alarming()
{
 wait_mark=0;
}
void stop()
{
 wait_mark=0;
}
```

在上面的任务 1 中，增加语句 signal(SIGINT,SIG_IGN)和语句 signal(SIGQUIT,SIG_IGN)，观察执行结果，并分析原因。这里，signal(SIGINT,SIG_IGN)和signal(SIGQUIT,SIG_IGN)分别为忽略键信号以及忽略中断信号。

<程序>

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>

 int pid1,pid2;
 int EndFlag=0;
 int pf1=0;
 int pf2=0;

void IntDelete()
{
 kill(pid1,16);
 kill(pid2,17);

}

void Int1()
{
 printf("child process 1 is killed !by parent\n");
 exit(0);
}
void Int2()
```

```
{
printf("child process 2 is killed !by parent\n");
exit(0);
}

main()
{
int exitpid;
    if(pid1=fork())
    {
      if(pid2=fork())
      {
        signal(SIGINT,IntDelete);
        waitpid(-1,&exitpid,0);
        waitpid(-1,&exitpid,0);
        printf("parent process is killed\n");
        exit(0);
      }
      else
      {
        signal(SIGINT,SIG_IGN);
        signal(17,Int2);
        pause();
      }
    }
    else
    {
       signal(SIGINT,SIG_IGN);
       signal(16,Int1);
       pause();
    }
}
```

步骤四：管道通信

编制一段程序，实现进程的管道通信。使用系统调用 pipe()建立一条管道线。两个子进程 p1 和 p2 分别向通道个写一句话：

child1 process is sending message!

child2 process is sending message!

而父进程则从管道中读出来自两个进程的信息，显示在屏幕上。

〈程序〉

```
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
int pid1,pid2;

main( )
{
int fd[2];
char outpipe[100],inpipe[100];
pipe(fd);                          /*创建一个管道*/
while ((pid1=fork( ))==-1);
if(pid1==0)
  { lockf(fd[1],1,0);
    sprintf(outpipe,"child 1 process is sending message!");
    /*把串放入数组 outpipe 中*/
    write(fd[1],outpipe,50);     /*向管道写长为 50 字节的串*/
    sleep(5);                  /*自我阻塞 5 秒*/
    lockf(fd[1],0,0);
    exit(0);
   }
else
  { while((pid2=fork( ))==-1);
    if(pid2==0)
  {     lockf(fd[1],1,0);           /*互斥*/
        sprintf(outpipe,"child 2 process is sending message!");
        write(fd[1],outpipe,50);
        sleep(5);
        lockf(fd[1],0,0);
        exit(0);
     }
     else
     {   wait(0);               /*同步*/
         read(fd[0],inpipe,50);   /*从管道中读长为 50 字节的串*/
         printf("%s\n",inpipe);
         wait(0);
         read(fd[0],inpipe,50);
         printf("%s\n",inpipe);
         exit(0);
    }
  }
}
```

分析 wait（）系统调用、sleep（）系统调用的作用。

实验三：进程间通信

预计课时：2 学时

实训时长： 40(分钟)

实验简介：此次实验主要是了解消息通信以及共享存储区通信。

实验目标：

（1）了解消息通信。

（2）了解共享存储区通信。

实验内容：

（1）消息的创建，发送和接收

使用系统调用 msgget( )，megsnd( )，msgrev( )及 msgctl()编制一长度为 1K 的消息发送和接收的程序。

（2）共享存储区的创建,附接和断接

使用系统调用 shmget(),sgmat(),smgdt(),shmctl()编制一个长度为 1K 的消息发送和接收的程序。

实验所需基础：

操作系统：Linux RHEL 6.0

实验是否需要联网：否

实训步骤：

步骤一：消息通信

（1） 为了便于操作和观察结果，用一个 程序为“引子”，先后 fork( )两个子进程，SERVER 和 CLIENT，进行通信。

（2） SERVER 端建立一个 Key 为 75 的消息队列，等待其他进程

发来的消息。当遇到类型为 1 的消息，则作为结束信号，取消该队列，并退出 SERVER 。SERVER 每接收到一个消息后显示一句“(server)received”。

（3） CLIENT 端使用 Key 为 75 的消息队列，先后发送类型从 10 到 1 的消息，然后退出。最后的一个消息，既是 SERVER 端需要的结束信号。CLIENT 每发送一条消息后显示一句“(client)sent”。

（4） 父进程在 SERVER 和 CLIENT 均退出后结束。

〈程序〉

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#define MSGKEY 75           /*定义关键词 MEGKEY*/
struct msgform                 /*消息结构*/
{
long mtype;
char mtexe[1030];          /*文本长度*/
}msg;
int msgqid,i;

void CLIENT( )
{
int i;
msgqid=msgget(MSGKEY,0777|IPC_CREAT);
for(i=50;i>=1;i--)
{
     msg.mtype=i;
     printf("(client)sent\n");
        msgsnd(msgqid,&msg,1024,0);        /*发送消息 msg 入 msgid 消息队列*/
}
exit(0);
}

void SERVER( )
{
```

```
  msgqid=msgget(MSGKEY,0777|IPC_CREAT); /*由关键字获得消息队列*/
  do
  {
   msgrcv(msgqid,&msg,1030,0,0);  /*从队列 msgid 接受消息 msg*/
   printf("(server)receive\n");
  }while(msg.mtype!=1);              /*消息类型为 1 时，释放队列*/
   msgctl(msgqid, IPC_RMID,0);
  exit(0);

}

void main()
{
  while ((i=fork())==-1);
    if(!i) SERVER();
  while ((i=fork())==-1);
    if(!i) CLIENT();
  wait(0);
  wait(0);
}
```

步骤二：共享存储区通信

（1）为了便于操作 和观察结果，用一个 程序为“引子”，先后 fork( )两个子进程，SERVER 和 CLIENT，进行通信。

（2）SERVER 端建立一个 KEY 为 75 的共享区,并将第一个字节置为-1.作为数据空的标志.等待其他进程发来的消息.当该字节的值发生变化时,表示收到了该消息,进行处理.然后再次把它 的值设为-1。如果遇到的值为 0,则视为结束信号,取消该队列,并退出 SERVER.SERVER 每接 收到一次数据后显示”(server)received”。

（3）CLIENT 端建立一个为 75 的共享区,当共享取得第一个字节为-1 时，Server 端空闲,可发送请求。CLIENT 随即填入 9 到 0。期间等待Server端再次空闲.进行完这些操作后,CLIENT退出。CLIENT 每发送一次数据后显示”(client)sent”。

（4）父进程在 SERVER 和 CLIENT 均退出后结束。

<程序>

```
#include<sys/types.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#define SHMKEY  75                          /*定义共享区关键词*/
int shmid,i;
int *addr;

 void CLIENT()
{
int i;
shmid=shmget(SHMKEY,1024, 0777|IPC_CREAT);
     /*  获取共享区，长度 1024，关键词 SHMKEY */
addr=shmat(shmid,0,0);                      /*共享区起始地址为 addr*/
for(i=9;i>=0;i--)
{
    while(*addr!= -1);
    printf("(client)sent\n");                    /*打印(client)sent*/

    *addr=i;                                  /*把 i 赋给 addr*/
}
exit(0);
}

void SERVER()
{
   shmid=shmget(SHMKEY,1024,0777|IPC_CREAT);  /*创建共享区*/
addr=shmat(shmid,0,0);                          /*共享区起始地址为
addr*/
do
{
    *addr=-1;
      while(*addr == -1);
    printf("(server)received\n%d",*addr);           /*服务进程使用
共享区*/
} while(*addr);
  shmctl(shmid,IPC_RMID,0);
}

 void main()
{
```

```
  while ((i=fork())==-1);
    if(!i) SERVER();
  while ((i=fork())==-1);
    if(!i) CLIENT();
  wait(0);
  wait(0);
}
```

实验四：处理器（进程）调度

预计课时：2 学时

实训时长： 60(分钟)

实验简介：本实验是采用优先级进程调度算法来模拟演示进程调度。

实验目标：

熟悉优先级进程调度的原理与流程。

实验内容：

（1）设计一个有 N 个进程共行的进程调度程序。每个进程由一个进程控制块 PCB 表示。进程控制块包括以下信息：进程名，进程优先数，进程需要运行的时间，占用 CPU 的时间以及进程的状态等。

（2）本调度程序用优先数调度算法。

（3）编写程序并调试运行。

实验所需基础：

操作系统：Linux RHEL 6.0

实验是否需要联网：否

实训步骤：

步骤一：算法

本程序采用优先数算法对 N 个进程进行调度。每个进程处于 R，就绪 W 和完成 F 三种状态之一，并假定起始状态就是就绪状态 W。

为了方便处理，有如下假设：每个进程的优先数=50-进程完成总耗时。

步骤二：编程

〈程序〉

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
typedef struct node /*创建 PCB*/
{   char name[10];  /*进程标识*/
   int prio;   /*进程优先数*/
  int cputime; /*进程占用 CPU 时间*/
   int needtime; /*进程完成所需时间*/
   int count;  /*计数器*/
   char state; /*进程的状态*/
   struct node *next; /*链指针*/

}PCB;
PCB *finish,*ready,*tail,*run;
int N;

firstin()          /*创建就绪队列对头指针*/
{
   run=ready;
   run->state='R';
   ready=ready->next;
}


void prt(char algo)  /*演示进程调度*/
{
   PCB *p;
   printf("   NAME  CPUTIME   NEEDTIME   PRIORITY   STATUS\n");
   if(run!=NULL)
   printf("  %-10s%-10d%-10d%-10d %c\n",run->name,
       run->cputime,run->needtime,run->prio,run->state);
   p=ready;
   while(p!=NULL)
   {  printf("  %-10s%-10d%-10d%-10d %c\n",p->name,
      p->cputime,p->needtime,p->prio,p->state);
      p=p->next;
   }
   p=finish;
   while(p!=NULL)
   {  printf("  %-10s%-10d%-10d%-10d %c\n",p->name,
```

```
      p->cputime,p->needtime,p->prio,p->state);
      p=p->next;
   }
   getch();
}

insert(PCB *q)
{
   PCB *p1,*s,*r;
   int b;
   s=q;
   p1=ready;
   r=p1;
   b=1;
   while((p1!=NULL)&&b)
      if(p1->prio>=s->prio)
      {
     r=p1;
     p1=p1->next;
      }
      else
     b=0;
   if(r!=p1)
   {
      r->next=s;
      s->next=p1;
   }
   else
   {
      s->next=p1;
      ready=s;
   }
}

void create(char alg)   /*创建各个进程*/
{
   PCB *p;
   int i,time;
   char na[10];
   ready=NULL;
   finish=NULL;
   run=NULL;
    for(i=1;i<=N;i++)
   {
```

```
        p=malloc(sizeof(PCB));
        printf("Enter NAME of process:\n");
        scanf("%s",na);
        printf("Enter TIME of process(less than 50):\n");
        scanf("%d",&time);
        strcpy(p->name,na);
        p->cputime=0;
        p->needtime=time;
        p->state='w';
        p->prio=50-time;  /*假设优先级与耗时之和为 50*/
        if(ready!=NULL)
       insert(p);
        else
        {
      p->next=ready;
      ready=p;
       }
   }
   clrscr();
   printf("          DISPLAY OF THE PROGRESS:\n");
   printf("************************************************\n");
   prt(alg);
   run=ready;
   ready=ready->next;
   run->state='R';
}

priority(char alg)   /*优先级算法调度*/
{
   while(run!=NULL&&run->prio>=0)
   {
      run->cputime=run->cputime+1;
      run->needtime=run->needtime-1;
      run->prio=run->prio-3;
      if(run->needtime==0)
      {
     run->next=finish;
     finish=run;
     run->state='F';
     run=NULL;
     if(ready!=NULL)
       firstin();
      }
      else
```

```
        if((ready!=NULL)&&(run->prio<ready->prio))
        {
           run->state='W';
           insert(run);
           firstin();
        }
         prt(alg);
      }
   }

   main()
   {  char algo;
      clrscr();
   loop:     printf("Enter THE TOTAL NUMBER of PCB(less than 10 is
better):\n");
      scanf("%d",&N);
      if(N>10)
      {printf("it's too big,and select a small number.\n");
      goto loop;}
      create(algo);
      priority(algo);}
```

实验五：存储管理

预计课时：2 学时

实训时长： 60(分钟)

实验简介：使用先进先出算法（FIFO)、最近最少使用算法（LRU)、最佳淘汰算法(OPT)、最少访问页面算法(LFU)、最近最不经常使用算法(NUR)计算页式存储管理中的内存页面命中率。

实验目标：

（1）熟悉先进先出算法（FIFO)、最近最少使用算法（LRU)、最佳淘汰算法(OPT)、最少访问页面算法(LFU)、最近最不经常使用算法(NUR)。

（2）了解上述算法的优缺点。

实验内容：

设计一个虚拟存储区和内存工作区,并使用下列算法计算访问命中率。

（1） 进先出的算法（FIFO）

（2） 最近最少使用的算法（LRU）

（3） 最佳淘汰算法(OPT)

（4） 最少访问页面算法(LFU)

（5） 最近最不经常使用算法(NUR)

命中率=(1-页面失效次数)/页地址流长度**实验所需基础：**

操作系统：Linux RHEL 6.0

实验是否需要联网：否

实训步骤：

步骤一：数据结构设计

（1）页面类型

```
typedef struct{
                int pn,pfn,counter,time;
              }pl-type;
```

其中 pn 为页号,pfn 为面号, counter 为一个周期内访问该页面的次数, time 为访问时间。

（2）页面控制结构

```
pfc-struct{
           int pn,pfn;
           struct pfc_struct *next;
         }
typedef   struct pfc_struct pfc_type;
pfc_type  pfc_struct[total_vp],*freepf_head,*busypf_head;
pfc_type  *busypf_tail;
```

其中 pfc[total_vp]定义用户进程虚页控制结构
*freepf_head 为空页面头的指针
*busypf_head 为忙页面头的指针
*busypf_tail 为忙页面尾的指针

步骤二：函数定义

（1）void initialize( ):初始化函数,给每个相关的页面赋值。

（2）void FIFO( ):计算使用 FIFO 算法时的命中率。

（3）void LRU( ):计算使用 LRU 算法时的命中率。

（4）void OPT( ):计算使用 OPT 算法时的命中率。

（5）void LFU( ):计算使用 LFU 算法时的命中率。

（6）void NUR( ):计算使用 NUR 算法时的命中率。

步骤三：变量定义

（1）int a[total_instruction]: 指令流数据组

（2）int page[total_instruction]：每条指令所属的页号

（3）int offset[total_instruction]：每页装入 10 条指令后取模运算页号偏移值

（4）int total_pf：用户进程的内存页面数

（5）int disaffect：页面失效次数

步骤四：程序设计

<程序>

```
#define TRUE 1
#define FALSE 0
#define INVALID -1
#define NULL  0

#define  total_instruction 320     /*指令流长*/
#define  total_vp  32               /*虚页长*/
#define  clear_period  50           /*清 0 周期*/

typedef struct                      /*页面结构*/
{
    int pn;      //页号 logic number
    int pfn;     //页面框架号 physical frame number
    int counter; //计数器
    int time;    //时间
}pl_type;

pl_type pl[total_vp];                       /*页面线性结构---指令序列需要使用地址*/

typedef struct pfc_struct                   /*页面控制结构，调度算法的控制结构*/
{
    int pn;
    int pfn;
    struct pfc_struct *next;
}pfc_type;


pfc_type pfc[total_vp], *freepf_head, *busypf_head, *busypf_tail;
```

```
int diseffect,  a[total_instruction]; /* a[]为指令序列*/

int page[total_instruction],  offset[total_instruction];/*地址信息*/

void initialize(int);
float  FIFO(int);
float  LRU(int);
float  LFU(int);
float  NUR(int); //not use recently
float  OPT(int);

int main( )
{
    int s,i,j;
   float sfifo, slru,slfu,snur,sopt;
   sfifo=slru=slfu=snur=sopt=0;
    srand(10*getpid());                    /*由于每次运行时进程号不同，故可用来
作为初始化随机数队列的“种子”*/

    s=(float)319*rand( )/32767/32767/2+1;  /*正态分布*/

    for(i=0;i<total_instruction;i+=4)        /*产生指令队列*/
    {
        if(s<0||s>319)
        {
            printf("When i==%d,Error,s==%d\n",i,s);
            exit(0);
        }
        a[i]=s;                                    /*任选一指令访问点 m*/
        a[i+1]=a[i]+1;                             /*顺序执行一条指令*/
        a[i+2]=(float)a[i]*rand( )/32767/32767/2; /*执行前地址指令 m*/
        a[i+3]=a[i+2]+1;                           /*顺序执行一条指令*/

        s=(float)(318-a[i+2])*rand( )/32767/32767/2+a[i+2]+2;
        if((a[i+2]>318)||(s>319))

            printf("a[%d+2],a number which is :%d and s==%d\n",i,a[i+2],s);

    }
    for (i=0;i<total_instruction;i++) /*将指令序列变换成页地址流*/
    {
        page[i]=a[i]/10;
        offset[i]=a[i]%10;
    }
```

```
    for(i=4;i<=32;i++)    /*用户内存工作区从 4 个页面到 32 个页面*/
    {
        printf("---%2d page frames---",i);
        sfifo=sfifo+FIFO(i);
        slru=slru+LRU(i);
        slfu=slfu+LFU(i);
        snur=snur+NUR(i);
        sopt=sopt+OPT(i);
      printf("\n");

    }
   printf("Average hit rate: ");
    printf("FIFO:%6.4f ",sfifo/29);
   printf("LRU:%6.4f ",slru/29);
   printf("LFU:%6.4f ",slfu/29);
   printf("NUR:%6.4f ",snur/29);
   printf("OPT:%6.4f ",sopt/29);
   printf("\n");
   return 0;
}

/*初始化相关数据结构 total_pf 表示内存的块数 */

void initialize(int total_pf)
{
    int i;
    diseffect=0;
    for(i=0;i<total_vp;i++)
    {
      pl[i].pn=i;
        pl[i].pfn=INVALID;        /*置页面控制结构中的页号，页面为空*/
        pl[i].counter=0;          /*页面控制结构中的访问次数为 0*/
        pl[i].time=-1;            /*访问的时间*/
    }

    for(i=0;i<total_pf-1;i++)    /*建立 pfc[i-1]和 pfc[i]之间的链接*/
    {
        pfc[i].next=&pfc[i+1];
        pfc[i].pfn=i;
    }

    pfc[total_pf-1].next=NULL;
    pfc[total_pf-1].pfn=total_pf-1;
    freepf_head=&pfc[0];          /*空页面队列的头指针为 pfc[0]*/
```

```
}

float FIFO(int total_pf)              /*先进先出算法 total_pf:用户进程的内存页面
数*/
{
    int i,j;
    pfc_type *p;                      /*中间变量*/
    initialize(total_pf);            /*初始化相关页面控制用数据结构*/
    busypf_head=busypf_tail=NULL; /*忙页面队列头，队列尾链接*/
    for(i=0;i<total_instruction;i++)
    {
        if(pl[page[i]].pfn==INVALID)   /*页面失效*/
        {
            diseffect+=1;                  /*失效次数*/
            if(freepf_head==NULL)          /*无空闲页面*/
            {
                p=busypf_head->next;
                pl[busypf_head->pn].pfn=INVALID;
                freepf_head=busypf_head;  /*释放忙页面队列的第一个页面*/
                freepf_head->next=NULL;  /*表明还是缺页*/
                busypf_head=p;
            }
            p=freepf_head->next;
         freepf_head->next=NULL; /*使 busy 的尾为 null*/
            freepf_head->pn=page[i]; /*填入虚页号 */
            pl[page[i]].pfn=freepf_head->pfn;  /* 填入实页面号 */
            if(busypf_tail==NULL)
            {
                busypf_head=busypf_tail=freepf_head;
            }
            else
            {
                busypf_tail->next=freepf_head;  /* free 页面减少一个 */
                busypf_tail=freepf_head;
            }
            freepf_head=p;
        }
    }
    printf("FIFO:%6.4f ",1-(float)diseffect/320);
    return 1-(float)diseffect/320;
}

float LRU (int total_pf)        /*最近最久未使用算法 least recently used*/
```

```
{
    int min,minj,i,j,present_time; /*minj 为最小值下标*/
    initialize(total_pf);
    present_time=0;
    for(i=0;i<total_instruction;i++)
    {
        if(pl[page[i]].pfn==INVALID)              /*页面失效*/
        {
            diseffect++;
            if(freepf_head==NULL)               /*无空闲页面*/
            {
                min=32767;                          /*设置最大值*/
                for(j=0;j<total_vp;j++)             /*找出 time 的最小值*/
                {
                    if(min>pl[j].time&&pl[j].pfn!=INVALID)
                    {
                        min=pl[j].time;
                        minj=j;
                    }
                }
                freepf_head=&pfc[pl[minj].pfn];   //腾出一个单元
                pl[minj].pfn=INVALID;
                pl[minj].time=0;
                freepf_head->next=NULL;
            }
            pl[page[i]].pfn=freepf_head->pfn;   //有空闲页面,改为有效
            pl[page[i]].time=present_time;
            freepf_head=freepf_head->next;      //减少一个 free 页面
        }
        else
        {
            pl[page[i]].time=present_time;         //命中则增加该单元的访问次数
            present_time++;
        }
    }
    printf("LRU:%6.4f ",1-(float)diseffect/320);
    return 1-(float)diseffect/320;
}

float NUR(int  total_pf )        /*最近未使用算法 Not Used recently count 表示*/
{
int i,j,dp,cont_flag,old_dp;
pfc_type *t;
initialize(total_pf);
```

```
dp=0;

for(i=0;i<total_instruction;i++)
{
    if (pl[page[i]].pfn==INVALID)          /*页面失效*/
    {
        diseffect++;
        if(freepf_head==NULL)                /*无空闲页面*/
        {
            cont_flag=TRUE;
            old_dp=dp;

            while(cont_flag)
            {

               if(pl[dp].counter==0&&pl[dp].pfn!=INVALID)
                    cont_flag=FALSE;
                else
                {
                    dp++;
                    if(dp==total_vp)
                        dp=0;
                    if(dp==old_dp)
                        for(j=0;j<total_vp;j++)
                         pl[j].counter=0;
                }
            }
            freepf_head=&pfc[pl[dp].pfn];
            pl[dp].pfn=INVALID;
            freepf_head->next=NULL;
        }

        pl[page[i]].pfn=freepf_head->pfn;

        freepf_head->pn=page[i];

        freepf_head=freepf_head->next;
    }
    else
        pl[page[i]].counter=1;
    if(i%clear_period==0)
        for(j=0;j<total_vp;j++)
            pl[j].counter=0;
}
```

```
printf("NUR:%6.4f ",1-(float)diseffect/320);
return 1-(float)diseffect/320;
}

float OPT(int total_pf)        /*最佳置换算法*/
{
    int i,j, max,maxpage,d,dist[total_vp];
    pfc_type *t;
    initialize(total_pf);
    for(i=0;i<total_instruction;i++)
    {
        if(pl[page[i]].pfn==INVALID)       /*页面失效*/
        {
            diseffect++;
            if(freepf_head==NULL)          /*无空闲页面*/
            {
                for(j=0;j<total_vp;j++)
                {
                    if(pl[j].pfn!=INVALID)
                        dist[j]=32767;
                    else
                        dist[j]=0;
                }
            d=1;
                for(j=i+1;j<total_instruction;j++)
                {
                    if(pl[page[j]].pfn!=INVALID)
                    {
                        dist[page[j]]=d;
                        }
                   d++;
                }
                max=-1;
                for(j=0;j<total_vp;j++)
                    if(max<dist[j])
                    {
                        max=dist[j];
                        maxpage=j;
                    }
                    freepf_head=&pfc[pl[maxpage].pfn];
                    freepf_head->next=NULL;
                    pl[maxpage].pfn=INVALID;
            }
            pl[page[i]].pfn=freepf_head->pfn;
```

```
            freepf_head=freepf_head->next;}
    }
    printf("OPT:%6.4f ",1-(float)diseffect/320);
    return 1-(float)diseffect/320;
}
/*该算法时根据已知的预测未知的，least frequency  Used是最不经常使用置换法*/
float  LFU(int total_pf)
{
    int i,j,min,minpage;
    pfc_type *t;
    initialize(total_pf);
    for(i=0;i<total_instruction;i++)
    {
        if(pl[page[i]].pfn==INVALID)      /*页面失效*/
        {
            diseffect++;
            if(freepf_head==NULL)          /*无空闲页面*/
            {
                min=32767;
                /*获取counter的使用用频率最小的内存*/
                for(j=0;j<total_vp;j++)
                {
                    if(min>pl[j].counter&&pl[j].pfn!=INVALID)
                    {
                        min=pl[j].counter;
                        minpage=j;}
                }
                freepf_head=&pfc[pl[minpage].pfn];
                pl[minpage].pfn=INVALID;
                pl[minpage].counter=0;
                freepf_head->next=NULL;
            }
            pl[page[i]].pfn=freepf_head->pfn;   //有空闲页面,改为有效
            pl[page[i]].counter++;
            freepf_head=freepf_head->next;      //减少一个free 页面
        }
        else
           pl[page[i]].counter=pl[page[i]].counter+1;}
    printf("LFU:%6.4f ",1-(float)diseffect/320);
    return 1-(float)diseffect/320; }
```

步骤五：分析结果

分析几种算法的页面命中率，说明产生结果的可能原因。

实验六：文件系统设计▲

预计课时：6 学时

实训时长： 160(分钟)

实验简介：本实验的目的是通过一个简单多用户文件系统的设计，加深理解文件系统的内部功能和内部实现。

实验目标：

为 DOS 系统设计一个简单的二级文件系统。要求做到以下几点：

（1）可以实现下列几条命令

LOGIN　　用户登陆

DIR　　列文件目录

CREATE　创建文件

DELETE　删除文件

OPEN　　打开文件

CLOSE　　关闭文件

READ　　读文件

WRITE　　写文件

（2）列目录时要列出文件名，物理地址，保护码和文件长度。

（3）源文件可以进行读写保护。

实验内容：

（1）本文件系统采用两级目录，其中第一级对应于用户账号，第二级对应于用户账号下的文件。

（2）确定文件系统的数据结构：主目录、子目录及活动文件等。

主目录和子目录都以文件的形式存放于磁盘，这样便于查找和修改。

（3）用户创建的文件，可以编号存储于磁盘上。如：file0,file1,file2…并以编号作为物理地址，在目录中进行登记。

实验所需基础：

操作系统：Windows XP 及以上版本

软件：wintc

实验是否需要联网：否

实训步骤：

步骤一：主要数据结构设计

（1）OSFILE 结点

```
typedef struct  /*the structure of OSFILE*/
      {int   fpaddr;                    /*file physical address*/
        int  flength;                   /*file length*/
        int   fmode;    /*file mode:0-Read Only;1-Write Only;2-Read and Write(default);*/
        char fname[MAXNAME];            /*file name*/
       } OSFILE;
```

（2）OSUFD 结点

```
typedef struct     /*the structure of OSUFD*/
      { char ufdname[MAXNAME];   /*ufd name*/
        OSFILE ufdfile[MAXCHILD];   /*ufd own file*/
```

```
        }OSUFD;
```

（3）用户密码

```
typedef struct  /*the structure of OSUFD'LOGIN*/
     {char ufdname[MAXNAME];        /*ufd name*/
      char ufdpword[8];             /*ufd password*/
      } OSUFD_LOGIN;
```

（4）文件打开模式

```
typedef struct     /*file open mode*/
    {int ifopen;     /*ifopen:0-close,1-open*/
     int openmode;    /*0-read only,1-write only,2-read
and write,3-initial*/
    }OSUFD_OPENMODE;
```

（5）i 结点

```
struct dinode
{
 unsigned short di_number;         /*关联文件数*/
 unsigned short di_mode;           /*存取权限*/
 unsigned short di_uid;
 unsigned short di_gid;
unsigned long di_size;                 /*文件大小*/
unsigned int di_addr[NADDR];           /*物理块号*/1
}
```

步骤二：主要函数设计

（1）登陆文件系统函数 LoginF()

（2）目录操作函数 DirF()

（3）改变当前目录函数 CdF()

（4）创建文件函数 CreateF()

（5）删除文件函数 DeleteF()

（6）修改文件权限函数 ModifyFM()

（7）打开文件函数 OpenF()

（8）关闭文件函数 CloseF()

（9）读文件函数 ReadF()

（10）写文件函数 WriteF()

（11）退出文件系统函数 QuitF()

（12）帮助函数 help()

步骤三：程序设计

参考程序见下（本程序需要在 c:下建一个名为 osfile 的目录及一个名为 file 的子目录）：

```
#include "stdio.h"
#include "string.h"
#include "conio.h"
#include "stdlib.h"
#define MAXNAME 25  /*the largest length of mfdname,ufdname,filename*/
#define MAXCHILD 50 /*the largest child*/
#define MAX (MAXCHILD*MAXCHILD) /*the size of fpaddrno*/

typedef struct  /*the structure of OSFILE*/
  {int  fpaddr;               /*file physical address*/
   int  flength;              /*file length*/
   int  fmode;   /*file mode:0-Read Only;1-Write Only;2-Read and
```

```
Write(default);*/
   char fname[MAXNAME];        /*file name*/
  } OSFILE;

typedef struct     /*the structure of OSUFD*/
  {char ufdname[MAXNAME];   /*ufd name*/
   OSFILE ufdfile[MAXCHILD];   /*ufd own file*/
  }OSUFD;

typedef struct  /*the structure of OSUFD'LOGIN*/
  {char ufdname[MAXNAME];       /*ufd name*/
   char ufdpword[8];            /*ufd password*/
  } OSUFD_LOGIN;

typedef struct     /*file open mode*/
  {int ifopen;     /*ifopen:0-close,1-open*/
   int openmode;   /*0-read only,1-write only,2-read and write,3-initial*/
  }OSUFD_OPENMODE;

OSUFD *ufd[MAXCHILD];   /*ufd and ufd own files*/
OSUFD_LOGIN ufd_lp;

int ucount=0;  /*the count of mfd's ufds*/
int fcount[MAXCHILD];  /*the count of ufd's files*/
int loginsuc=0; /*whether login successfully*/
char username[MAXNAME];  /*record login user's name22*/
char dirname[MAXNAME];/*record current directory*/
int fpaddrno[MAX];  /*record file physical address num*/
OSUFD_OPENMODE ifopen[MAXCHILD][MAXCHILD]; /*record file open/close*/
int wgetchar; /*whether getchar()*/

FILE *fp_mfd,*fp_ufd,*fp_file_p,*fp_file;

void main()
{int i,j,choice1;
 char choice[50];
 /*choice operation:dir,create,open,delete,modify,read,write*/
 int choiceend=1;  /*whether choice end*/
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 void LoginF();  /*LOGIN FileSystem*/
 void DirF();  /*Dir FileSystem*/
 void CdF();  /*Change Dir*/
 void CreateF();  /*Create File*/
```

```
void DeleteF(); /*Delete File*/
void ModifyFM(); /*Modify FileMode*/
void OpenF();  /*Open File*/
void CloseF();  /*Close File*/
void ReadF(); /*Read File*/
void WriteF(); /*Write File*/
void QuitF(); /*Quit FileSystem*/
void help();
if((fp_mfd=fopen("c:\\osfile\\mfd","rb"))==NULL)
  {fp_mfd=fopen("c:\\osfile\\mfd","wb");
   fclose(fp_mfd);
  }

for(i=0;i<MAX;i++) fpaddrno[i]=0;
textattr(BLACK*16|WHITE);
clrscr();   /*clear screen*/
LoginF();   /*user login*/
clrscr();

if(loginsuc==1)  /*Login Successfully*/
{while (1)
{wgetchar=0;
 if (choiceend==1)
 {printf("\n\nC:\\%s>",strupr(dirname));}
 else printf("Bad command or file name.\nC:\\%s>",strupr(username));
  gets(choice);
  strcpy(choice,ltrim(rtrim(strlwr(choice))));
  if (strcmp(choice,"dir")==0) choice1=1;
  else if(strcmp(choice,"creat")==0) choice1=2;
  else if(strcmp(choice,"delete")==0) choice1=3;
  else if(strcmp(choice,"attrib")==0) choice1=4;
  else if(strcmp(choice,"open")==0) choice1=5;
  else if(strcmp(choice,"close")==0) choice1=6;
  else if(strcmp(choice,"read")==0) choice1=7;
  else if(strcmp(choice,"modify")==0) choice1=8;
  else if(strcmp(choice,"exit")==0) choice1=9;
  else if(strcmp(choice,"cls")==0) choice1=10;
  else if(strcmp(choice,"cd")==0) choice1=11;
  else if(strcmp(choice,"help")==0) choice1=20;
  else choice1=12;
    switch(choice1)
      {case 1:DirF();choiceend=1;break;
   case 2:CreateF();choiceend=1;if(!wgetchar) getchar();break;
   case 3:DeleteF();choiceend=1;if(!wgetchar)getchar();break;
```

```
    case 4:ModifyFM();choiceend=1;if(!wgetchar) getchar();break;
    case 5:choiceend=1;OpenF();if (!wgetchar) getchar();break;
    case 6:choiceend=1;CloseF();if (!wgetchar) getchar();break;
    case 7:choiceend=1;ReadF();if (!wgetchar) getchar();break;
    case 8:choiceend=1;WriteF();if (!wgetchar) getchar();break;
    case 9:printf("\nYou have exited this system.");
           QuitF();exit(0);break;
    case 10:choiceend=1;clrscr();break;
    case 11:CdF();choiceend=1;break;
    case 20:help();choiceend=1;break;
    default:choiceend=0;
       }
      }
}
else printf("\nAccess denied.");
}

void help(void)
{
printf("\nThe Command List\n");
printf("\nCd  Attrib  Creat  Modify  Read  Open  Cls  Delete  Exit  Close\n");
}
char *rtrim(char *str)  /*remove the trailing blanks.*/
{int n=strlen(str)-1;
 while(n>=0)
 {if(*(str+n)!=' ')
   {*(str+n+1)='\0';
    break;
   }
   else n--;
 }
 if (n<0) str[0]='\0';
 return str;
}

char *ltrim(char *str) /*remove the heading blanks.*/
{char *rtrim(char *str);
 strrev(str);
 rtrim(str);
 strrev(str);
 return str;
}

void LoginF()  /*LOGIN FileSystem*/
```

```
{char loginame[MAXNAME],loginpw[9],logincpw[9],str[50];
 int i,j,flag=1;
 char a[25];
 int findout; /*login user not exist*/
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 void InputPW(char *password);  /*input password,use '*' replace*/
 void SetPANo(int RorW);  /*Set physical address num*/
 while(1)
 {findout=0;
 printf("\n\nLogin Name:");
 gets(loginame);
 ltrim(rtrim(loginame));
 fp_mfd=fopen("c:\\osfile\\","rb");
 for(i=0;fread(&ufd_lp,sizeof(OSUFD_LOGIN),1,fp_mfd)!=0;i++)
  if (strcmp(strupr(ufd_lp.ufdname),strupr(loginame))==0)
    {findout=1;
     strcpy(logincpw,ufd_lp.ufdpword);
    }
 fclose(fp_mfd);
 if (findout==1)  /*user exist*/
    {printf("Login Password:");
     InputPW(loginpw); /*input password,use '*' replace*/
      if (strcmp(loginpw,logincpw)==0)
     {strcpy(username,strupr(loginame));
      strcpy(dirname,username);
      fp_mfd=fopen("c:\\osfile\\","rb");
      for(j=0;fread(&ufd_lp,sizeof(OSUFD_LOGIN),1,fp_mfd)!=0;j++)
        {strcpy(str,"c:\\osfile\\");
         strcat(str,ufd_lp.ufdname);
         ufd[j]=(OSUFD*)malloc(sizeof(OSUFD));
         strcpy(ufd[j]->ufdname,strupr(ufd_lp.ufdname));
         fp_ufd=fopen(str,"rb");
         fcount[j]=0;

for(i=0;fread(&ufd[j]->ufdfile[i],sizeof(OSFILE),1,fp_ufd)!=0;i++,fcount[j]++)
           {ifopen[j][i].ifopen=0;
           ifopen[j][i].openmode=4;}
         fclose(fp_ufd);}
      fclose(fp_mfd);
      ucount=j;
         SetPANo(0);
     printf("\n\nLogin successful! Welcome to this FileSystem\n\n");
     loginsuc=1;
```

```
     return;}
      else
     {printf("\n\n");
      flag=1;
      while(flag)
      {printf("Login Failed!  Password Error.  Try Again(Y/N):");
      gets(a);
      ltrim(rtrim(a));
      if (strcmp(strupr(a),"Y")==0) {loginsuc=0;flag=0;}
      else if(strcmp(strupr(a),"N")==0){loginsuc=0;flag=0;return;}
      }
     }
      }
else
      {printf("New Password(<=8):");
       InputPW(loginpw); /*input new password,use '*' replace*/
       printf("\nConfirm Password(<=8):"); /*input new password,use '*' replace*/
       InputPW(logincpw);
      if (strcmp(loginpw,logincpw)==0)
     {strcpy(ufd_lp.ufdname,strupr(loginame));
      strcpy(ufd_lp.ufdpword,loginpw);
      fp_mfd=fopen("c:\\osfile\\","ab");
      fwrite(&ufd_lp,sizeof(OSUFD_LOGIN),1,fp_mfd);
      fclose(fp_mfd);
      strcpy(username,strupr(loginame));
      strcpy(dirname,loginame);
      strcpy(str,"c:\\osfile\\");
      strcat(str,username);
      if((fp_ufd=fopen(str,"rb"))==NULL)
        {fp_ufd=fopen(str,"wb");
         fclose(fp_ufd);
        }
      fp_mfd=fopen("c:\\osfile\\","rb");
      for(j=0;fread(&ufd_lp,sizeof(OSUFD_LOGIN),1,fp_mfd)!=0;j++)
        {strcpy(str,"c:\\osfile\\");
         strcat(str,ufd_lp.ufdname);
         ufd[j]=(OSUFD*)malloc(sizeof(OSUFD));
         strcpy(ufd[j]->ufdname,strupr(ufd_lp.ufdname));
         fp_ufd=fopen(str,"rb");

for(i=0;fread(&ufd[j]->ufdfile[i],sizeof(OSFILE),1,fp_ufd)!=0;i++,fcount[j]++)
            {ifopen[j][i].ifopen=0;
            ifopen[j][i].openmode=4;}
         fclose(fp_ufd);}
```

```
      fclose(fp_mfd);
      ucount=j;
      SetPANo(0);
      printf("\n\nLogin Successful! Welcome to this System\n\n");
      loginsuc=1;
      return;
      }
      else
      {printf("\n\n");
      flag=1;
      while(flag)
      {printf("Login Failed! Password Error. Try Again(Y/N):");
      gets(a);
      ltrim(rtrim(a));
      if (strcmp(strupr(a),"Y")==0) {loginsuc=0;flag=0;}
      else if(strcmp(strupr(a),"N")==0){loginsuc=0;flag=0;return;}
      }
      }
 }
 }
 }

void SetPANo(int RorW)  /*Set physical address num,0-read,1-write*/
{int i,j;
 if (RorW==0)
    {if((fp_file_p=fopen("c:\\osfile\\file\\file_p","rb"))==NULL)
        {fp_file_p=fopen("c:\\osfile\\file\\file_p","wb");
         fclose(fp_file_p);
        }
     fp_file_p=fopen("c:\\osfile\\file\\file_p","rb");
     for(i=0;fread(&j,sizeof(int),1,fp_file_p)!=0;i++)
       fpaddrno[j]=1;
     /*for(i=1;i<MAX;i++)
       if ((i%13)==0) fpaddrno[i]=1;*/
    }
 else
    {fp_file_p=fopen("c:\\osfile\\file\\file_p","wb");
     /*for(i=1;i<MAX;i++)
       if((i%13)==0) fpaddrno[i]=0;*/
     for(i=0;i<MAX;i++)
     if (fpaddrno[i]==1)
        fwrite(&i,sizeof(int),1,fp_file_p);
     }
 fclose(fp_file_p);
```

```
}

void InputPW(char *password)  /*input password,use '*' replace*/
{int j;
 for(j=0;j<=7;j++)
    {password[j]=getch();
     if ((int)(password[j])!=13)
       {if((int)(password[j])!=8)
          putchar('*');
        else
          {if (j>0)
        {j--;j--;
         putchar('\b');putchar(' ');putchar('\b');
        }
        else j--;
          }
        }
     else
      {password[j]='\0';
       break;
      }
    }
 password[j]='\0';
}

void DirF()  /*Dir FileSystem*/
{int i,j,count=0;
 char sfmode[25],sfpaddr[25],str[25];
 int ExistD(char *dirname);  /*Whether DirName Exist,Exist-i,Not Exist-0*/
 clrscr();
 if (strcmp(strupr(ltrim(rtrim(dirname))),"")!=0)
 {printf("\n\nC:\\%s>dir\n",dirname);

printf("\n%14s%16s%14s%10s%18s\n","FileName","FileAddress","FileLength","Type",
"FileMode");
  j=ExistD(dirname);
 for(i=0;i<fcount[j];i++)
   {if ((i%16==0)&&(i!=0))
     {printf("\nPress any key to continue..");
     getch();
     clrscr();

printf("\n%14s%16s%14s%10s%18s\n","FileName","FileAddress","FileLength","Type",
"FileMode");
```

```
     }
    itoa(ufd[j]->ufdfile[i].fpaddr,str,10);
    strcpy(sfpaddr,"file");
    strcat(sfpaddr,str);
    if (ufd[j]->ufdfile[i].fmode==0) strcpy(sfmode,"Read Only");
    else if(ufd[j]->ufdfile[i].fmode==1) strcpy(sfmode,"Write Only");
    else if(ufd[j]->ufdfile[i].fmode==2)strcpy(sfmode,"Read And Write");
    else strcpy(sfmode,"Protect");
    printf("%14s%16s%14d%10s%18s\n",ufd[j]->ufdfile[i].fname,sfpaddr,ufd[j]->uf
dfile[i].flength,"<FILE>",sfmode);
   }
 printf("\n %3d file(s)\n",fcount[j]);}
 else
 {printf("\n\nC:\\>dir\n");
  printf("\n%14s%18s%8s\n","DirName","OwnFileCount","Type");
  for(i=0;i<ucount;i++)
   {if ((i%16==0)&&(i!=0))
     {printf("\nPress any key to continue...");
     getch();
     clrscr();
       printf("\n%14s%18s%8s\n","DirName","OwnFileCount","Type");
    }
    printf("%14s%18d%8s\n",ufd[i]->ufdname,fcount[i],"<UFD>");
    count=count+fcount[i];
   }
  printf("\n %3d dir(s),%5d file(s)\n",ucount,count);
 }
}

int ExistD(char *dirname)  /*Whether DirName Exist,Exist-i,Not Exist-0*/
{int i;
 int exist=0;
 for(i=0;i<ucount;i++)
   if (strcmp(strupr(ufd[i]->ufdname),strupr(dirname))==0)
      {exist=1;
       break;
      }
 if (exist) return(i);
 else return(-1);
}

void CdF() /*Exchange Dir*/
{char dname[MAXNAME];
 char *rtrim(char *str);  /*remove the trailing blanks.*/
```

```
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistD(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 printf("\nPlease input DirName (cd..-Previous dir; DirNAME-cd [DirNAME]):");
 gets(dname);
 ltrim(rtrim(dname));
 if (ExistD(dname)>=0)  strcpy(dirname,strupr(dname));
 else if(strcmp(strupr(dname),"CD..")==0)  strcpy(ltrim(rtrim(dirname)),"");
 else printf("\nError.\'%s\' does not exist.\n",dname);
}

void CreateF()  /*Create File*/
{int fpaddrno,flag=1,i;
 char fname[MAXNAME],str[50],str1[50],strtext[255],a[25];
 char fmode[25];

 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int FindPANo();  /*find out physical address num*/
 int WriteF1(); /*write file*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(dirname),strupr(username))!=0)
 {printf("\nError. You must create file in your own dir.\n");wgetchar=1;}
 else
 {
 printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 if (ExistF(fname)>=0)
   {printf("\nError. Name \'%s\' has already existed.\n",fname);
    wgetchar=1;
   }
 else
   {printf("Please input FileMode(0-Read Only, 1-Write Only, 2-Read and Write,
3-Protect):");
    gets(fmode);
    ltrim(rtrim(fmode));
    if((strcmp(fmode,"0")==0)||(strcmp(fmode,"1")==0)||(strcmp(fmode,"2")==0)||
(strcmp(fmode,"3")==0))
      {fpaddrno=FindPANo();
       if (fpaddrno>=0)
     {i=ExistD(username);
      strcpy(ufd[i]->ufdfile[fcount[i]].fname,fname);
      ufd[i]->ufdfile[fcount[i]].fpaddr=fpaddrno;
```

```
      ufd[i]->ufdfile[fcount[i]].fmode=atoi(fmode);
      ifopen[i][fcount[i]].ifopen=0;
      ifopen[i][fcount[i]].openmode=4;
      strcpy(str,"c:\\osfile\\file\\file");
      itoa(fpaddrno,str1,10);
      strcat(str,str1);
      fp_file=fopen(str,"wb");
      fclose(fp_file);
      fcount[i]++;
      while(flag)
      {printf("Input text now(Y/N):");
       gets(a);
       ltrim(rtrim(a));
       ufd[i]->ufdfile[fcount[i]-1].flength=0;
       if(strcmp(strupr(a),"Y")==0)
        {fp_file=fopen(str,"wb+");
         ufd[i]->ufdfile[fcount[i]-1].flength=WriteF1();
         flag=0;
        }
       else if(strcmp(strupr(a),"N")==0){flag=0;wgetchar=1;}
      }
      printf("\n\'%s\' has been created successfully!\n",fname);
     }
       else
    {printf("\nFail!No Disk Space. Please format your disk.\n");wgetchar=1;}
      }
      else {printf("\nError. FileMode\'s Range is 0-3\n");wgetchar=1;}
   }}
}

int ExistF(char *filename)  /*Whether FileName Exist,Exist-i,Not Exist-0*/
{int i,j;
 int exist=0;
 int ExistD(char *dirname);
 j=ExistD(dirname);
 for(i=0;i<fcount[j];i++)
   if (strcmp(strupr(ufd[j]->ufdfile[i].fname),strupr(filename))==0)
      {exist=1;
       break;
      }
 if (exist) return(i);
 else return(-1);
}
```

```
int FindPANo()  /*find out physical address num*/
{int i;
 for(i=0;i<MAX;i++)
   if (fpaddrno[i]==0) {fpaddrno[i]=1;break;}
 if (i<MAX) return(i);
 else return(-1);
}

int WriteF1()  /*write file*/
{int length=0;
 char c;
 printf("Please input text(\'#\' stands for end):\n");
 while((c=getchar())!='#')
 {fprintf(fp_file,"%c",c);
  if (c!='\n') length++;
 }
 fprintf(fp_file,"\n");
 fclose(fp_file);
 return(length);
}

void DeleteF() /*Delete File*/
{char fname[MAXNAME];
 char str[50],str1[50];
 int i,j,k,flag=1;
 char a[25]; /*whether delete*/
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(dirname),strupr(username))!=0)
 {printf("\nError. You can only delete file in your own dir.\n");wgetchar=1;}
 else
 {printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if (i>=0)
    {k=ExistD(username);
     if(ifopen[k][i].ifopen==1)
      {printf("\nError. \'%s\' is in open status. Close it before
delete.\n",fname);wgetchar=1;}
     else
     {
```

```
      while(flag)
       {printf("\'%s\' will be deleted. Are you sure(Y/N):",fname);
    gets(a);
    ltrim(rtrim(a));
    if(strcmp(strupr(a),"Y")==0)
    {fpaddrno[ufd[k]->ufdfile[i].fpaddr]=0;
    itoa(ufd[k]->ufdfile[i].fpaddr,str,10);
    for(j=i;j<fcount[k]-1;j++)
      {strcpy(ufd[k]->ufdfile[j].fname,ufd[k]->ufdfile[j+1].fname);
       ufd[k]->ufdfile[j].fpaddr=ufd[k]->ufdfile[j+1].fpaddr;
       ufd[k]->ufdfile[j].flength=ufd[k]->ufdfile[j+1].flength;
       ufd[k]->ufdfile[j].fmode=ufd[k]->ufdfile[j+1].fmode;
       ifopen[k][j]=ifopen[k][j+1];
      }
    fcount[k]--;
    strcpy(str1,"c:\\osfile\\file\\file");
    strcat(str1,str);
    remove(str1);
    flag=0;
    printf("\n\'%s\' has been deleted successfully.\n",fname);
    wgetchar=1;
       }
     else if(strcmp(strupr(a),"N")==0)
     {printf("\nError. \'%s\' hasn\'t been deleted.\n",fname);
      wgetchar=1;
      flag=0;}
      }}}
 else
   {printf("\nError. \'%s\' does not exist.\n",fname);wgetchar=1;}}
}

void ModifyFM() /*Modify FileMode*/
{char fname[MAXNAME],str[50];
 int i,j,k,flag;
 char fmode[25]; /*whether delete*/
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 void InputPW(char *password);  /*input password,use '*' replace*/
 void SetPANo(int RorW);  /*Set physical address num*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(dirname),strupr(username))!=0) {printf("\nError.You can only
modify filemode in yourself dir.\n");wgetchar=1;}
 else
```

```
{ printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if (i>=0)
    {k=ExistD(username);
     if(ifopen[k][i].ifopen==1)
     {printf("\nError.\'%s\' is in open status. Close it before
modify.\n",fname);wgetchar=1;}
     else
     {
     if(ufd[k]->ufdfile[i].fmode==0) strcpy(str,"read only");    /*FileMode*/
     else if(ufd[k]->ufdfile[i].fmode==1) strcpy(str,"write only");
     else if(ufd[k]->ufdfile[i].fmode==2) strcpy(str,"read and write");
     else strcpy(str,"Protect");

     printf("\'%s\' filemode is %s.\n",fname,strupr(str));
     printf("Modify to(0-read only,1-write only,2-read and write,3-Protect):");
     gets(fmode);
     ltrim(rtrim(fmode));
     if(strcmp(fmode,"0")==0)
       {ufd[k]->ufdfile[i].fmode=0;
    printf("\n\'%s\' has been modified to READ ONLY mode successfully.\n",fname);
    wgetchar=1;
       }
     else if(strcmp(fmode,"1")==0)
       {ufd[k]->ufdfile[i].fmode=1;
    printf("\n\'%s\' has been modified to WRITE ONLY mode successfully.\n",fname);
    wgetchar=1;
       }
     else if(strcmp(fmode,"2")==0)
       {ufd[k]->ufdfile[i].fmode=2;
    printf("\n\'%s\' has been modified to READ AND WRITE mode
successfully.\n",fname);
    wgetchar=1;
       }
     else if(strcmp(fmode,"3")==0)
       {ufd[k]->ufdfile[i].fmode=3;
    printf("\n\'%s\' has been modified to FORBID mode successfully.\n",fname);
    wgetchar=1;
       }
     else {printf("\nError.\'%s\' is not modified.\n",fname);wgetchar=1;}
    }
  }
```

```
 else
   {printf("\nError. \'%s\' dose not exist.\n",fname);wgetchar=1;}}
}


void OpenF() /*Open File*/
{char fname[MAXNAME];
 char str[25],str1[25],fmode[25];
 int i,k;
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(ltrim(rtrim(dirname))),"")==0)
 {printf("\nError. Please change to ufd dir before open.\n");wgetchar=1;return;}
 printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if (i>=0)
  {k=ExistD(dirname);
   if(!ifopen[k][i].ifopen)
    {if (ufd[k]->ufdfile[i].fmode==3)
    {printf("\nError. The file\'s mode is FORBID. Can not open.\n");wgetchar=1;}
     else
     {printf("Please input FileOpenMode(0-Read Only,1-Write Only,2-Read and
Write):");
     gets(fmode);
     ltrim(rtrim(fmode));
     if((strcmp(fmode,"0")==0)||(strcmp(fmode,"1")==0)||(strcmp(fmode,"2")==0))
     {if(fmode[0]=='0')   /*open file with read only mode*/
    {strcpy(str,"read only");
     if((ufd[k]->ufdfile[i].fmode==0)||(ufd[k]->ufdfile[i].fmode==2))
ifopen[k][i].ifopen=1;
    }
      else if(fmode[0]=='1') /*open file with write only mode*/
    {strcpy(str,"write only");
     if((ufd[k]->ufdfile[i].fmode==1)||(ufd[k]->ufdfile[i].fmode==2))
ifopen[k][i].ifopen=1;
    }
      else if(fmode[0]=='2')  /*open file with read and write mode*/
    {strcpy(str,"read and write");
     if(ufd[k]->ufdfile[i].fmode==2) ifopen[k][i].ifopen=1;
    }
```

```
      if(ufd[k]->ufdfile[i].fmode==0) strcpy(str1,"read only");    /*FileMode*/
      else if(ufd[k]->ufdfile[i].fmode==1) strcpy(str1,"write only");
      else if(ufd[k]->ufdfile[i].fmode==2) strcpy(str1,"read and write");
      if(ifopen[k][i].ifopen==1)
    {ifopen[k][i].openmode=atoi(fmode);
     if (ifopen[k][i].openmode==0) strcpy(str,"read only");
     else if(ifopen[k][i].openmode==1) strcpy(str,"write only");
     else if(ifopen[k][i].openmode==2) strcpy(str,"read and write");
     printf("\n\'%s\' has been opened. OpenMode is %s,FileMode
is %s\n",fname,strupr(str),strupr(str1));
     wgetchar=1;
    }
       else
       {printf("\nError. \'%s\' hasn\'t been opened. OpenMode Error. OpenMode
is %s,but FileMode is %s\n",fname,strupr(str),strupr(str1));wgetchar=1;}
     }
     else {printf("\nError. FileOpenMode\'s Range is 0-2\n");wgetchar=1;}
     }}
    else {printf("\nError. \'%s\' is in open status.\n",fname);wgetchar=1;}
   }
 else
   {printf("\nError. \'%s\' does not exist.\n",fname);wgetchar=1;}
}

void CloseF() /*Close File*/
{int i,k,n=0;
 char fname[MAXNAME];
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(ltrim(rtrim(dirname))),"")==0)
 {printf("\nError. Please convert to ufd dir before close.\n");wgetchar=1;return;}
 k=ExistD(dirname);
 printf("\nOpen File(s) In This Ufd:\n");/*display openned file*/
 for(i=0;i<fcount[k];i++)
   {if (ifopen[k][i].ifopen==1) {printf("%15s",ufd[k]->ufdfile[i].fname);n++;}
    if((n%4==0)&&(n!=0)) printf("\n");
   }
 printf("\n%d files openned.\n",n);
 if (n==0) wgetchar=1;
 if(n!=0)
{printf("\nPlease input FileName:");
 gets(fname);
```

```
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if(i>=0)
  {if(ifopen[k][i].ifopen==1)
   {ifopen[k][i].ifopen=0;
    ifopen[k][i].openmode=4;
    printf("\n\'%s\' has been closed successfully.\n",fname);
    wgetchar=1;
   }
   else {printf("\nError.\'%s\' is in closing status.\n",fname);wgetchar=1;}
  }
 else {printf("\nError. \'%s\' is not exist.\n",fname);wgetchar=1;}
}
}

void ReadF() /*Read File*/
{int i,k,n=0;
 char fname[MAXNAME];
 char str[255],str1[255],c;
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 if (strcmp(strupr(ltrim(rtrim(dirname))),"")==0) {printf("\nError.Please
convert to ufd dir before read.\n");wgetchar=1;return;}
 printf("\nCaution:Open file first\n");
 printf("Opened File(s) List:\n");
 k=ExistD(dirname);
 for(i=0;i<fcount[k];i++)
   {if (ifopen[k][i].ifopen==1)
     if ((ifopen[k][i].openmode==0) ||(ifopen[k][i].openmode==2))
{printf("%15s",ufd[k]->ufdfile[i].fname);n++;}
    if((n%4==0)&&(n!=0)) printf("\n");
   }
 printf("\n%d files openned.\n",n);
 if (n==0) wgetchar=1;
 if(n!=0)
{printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if(i>=0)
  {if(ifopen[k][i].ifopen==1)
   {if((ifopen[k][i].openmode==0) ||(ifopen[k][i].openmode==2))
```

```
     {itoa(ufd[k]->ufdfile[i].fpaddr,str,10);
      strcpy(str1,"file");
      strcat(str1,str);
      strcpy(str,"c:\\osfile\\file\\");
      strcat(str,str1);
      fp_file=fopen(str,"rb");
      fseek(fp_file,0,0);
      printf("\nThe text is:\n\n");
      printf("   ");
      while(fscanf(fp_file,"%c",&c)!=EOF)
       if (c=='\n') printf("\n   ");
       else printf("%c",c);
      printf("\n\n%d Length.\n",ufd[k]->ufdfile[i].flength);
      fclose(fp_file);
      wgetchar=1;
     }
    else
    {printf("\nError.\'%s\' has been opened with WRITE ONLY mode. It isn\'t
read.\n",fname);wgetchar=1;}
   }
   else {printf("\nError.\'%s\' is in closing status. Please open it before
read\n",fname);wgetchar=1;}
  }
 else {printf("\nError. \'%s\' does not exist.\n",fname);wgetchar=1;}
}
}

void WriteF() /*Write File*/
{int i,k,n=0;
 char fname[MAXNAME];
 char str[50],str1[50],a[50];
 char *rtrim(char *str);  /*remove the trailing blanks.*/
 char *ltrim(char *str);  /*remove the heading blanks.*/
 int ExistF(char *filename);  /*Whether FileName Exist,Exist-i,Not Exist-0*/
 int ExistD(char *dirname);
 int WriteF1(); /*write file*/
 if (strcmp(strupr(ltrim(rtrim(dirname))),"")==0) {printf("\nError. Please
convert to ufd dir before write.\n");wgetchar=1;return;}
 k=ExistD(dirname);
 printf("\nOpen File(s) with write only mode or read and write mode:\n");/*display
openned files with writable mode*/
 for(i=0;i<fcount[k];i++)
   {if (ifopen[k][i].ifopen==1)
     if ((ifopen[k][i].openmode==1) ||(ifopen[k][i].openmode==2))
```

```
{printf("%15s",ufd[k]->ufdfile[i].fname);n++;}
    if((n%4==0)&&(n!=0)) printf("\n");
   }
 printf("\n%d files open.\n",n);
 if (n==0) wgetchar=1;
 if(n!=0)
{printf("\nPlease input FileName:");
 gets(fname);
 ltrim(rtrim(fname));
 i=ExistF(fname);
 if(i>=0)
  {if(ifopen[k][i].ifopen==1)
   {if((ifopen[k][i].openmode==1) ||(ifopen[k][i].openmode==2))
     {itoa(ufd[k]->ufdfile[i].fpaddr,str,10);
      strcpy(str1,"file");
      strcat(str1,str);
      strcpy(str,"c:\\osfile\\file\\");
      strcat(str,str1);
      if (ufd[k]->ufdfile[i].flength!=0)
    {printf("\n\'%s\' has text. Overwrite or Append(O-overwrite,A-Append,else-not
write):",fname);
     gets(a);
     ltrim(rtrim(a));
     if (fp_file!=NULL) fclose(fp_file);
     if (strcmp(strupr(a),"O")==0)
       {printf("\nOverwrite\n");
        fp_file=fopen(str,"wb");
        ufd[k]->ufdfile[i].flength=0;
        ufd[k]->ufdfile[i].flength=WriteF1();
       }
     else if(strcmp(strupr(a),"A")==0)
       {printf("\nAppend\n");
        fp_file=fopen(str,"ab");
        ufd[k]->ufdfile[i].flength=ufd[k]->ufdfile[i].flength+WriteF1();
       }
     else
       {printf("\nError.\'%s\' has not been written.\n",fname);
        fclose(fp_file);
        wgetchar=1;
       }
    }
       else
     {fp_file=fopen(str,"wb");
      ufd[k]->ufdfile[i].flength=WriteF1();
```

```
     }
     }
    else
    {printf("\nError. \'%s\' has been opened with read only mode.It isn\'t
writed.\n",fname);wgetchar=1;}
   }
   else
    {printf("\nError. \'%s\' is in closing status. Please open it before
write\n",fname);wgetchar=1;}
  }
 else
  {printf("\nError. \'%s\' does not exist.\n",fname);wgetchar=1;}
}
}

void QuitF() /*Quit FileSystem*/
{int i,j;
 char str[50];
 void SetPANo(int RorW);  /*Set physical address num,0-read,1-write*/
 SetPANo(1);
 if (fp_mfd!=NULL) fclose(fp_mfd);
 if (fp_ufd!=NULL) fclose(fp_ufd);
 if (fp_file!=NULL) fclose(fp_file);
 for(j=0;j<ucount;j++)
{strcpy(str,"c:\\osfile\\");
 strcat(str,ufd[j]->ufdname);
 ltrim(rtrim(str));
 fp_ufd=fopen(str,"wb");
 fclose(fp_ufd);
 fp_ufd=fopen(str,"ab");
 for(i=0;i<fcount[j];i++)
 fwrite(&ufd[j]->ufdfile[i],sizeof(OSFILE),1,fp_ufd);
 fclose(fp_ufd);}
}
```

步骤四：结果验证

（1）登陆界面，见图1

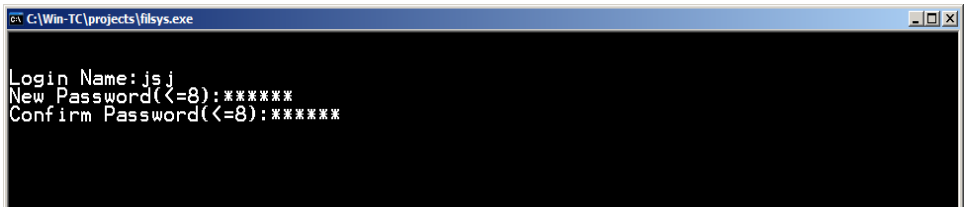


图 1 登陆界面

（2）创建文件界面，见图2

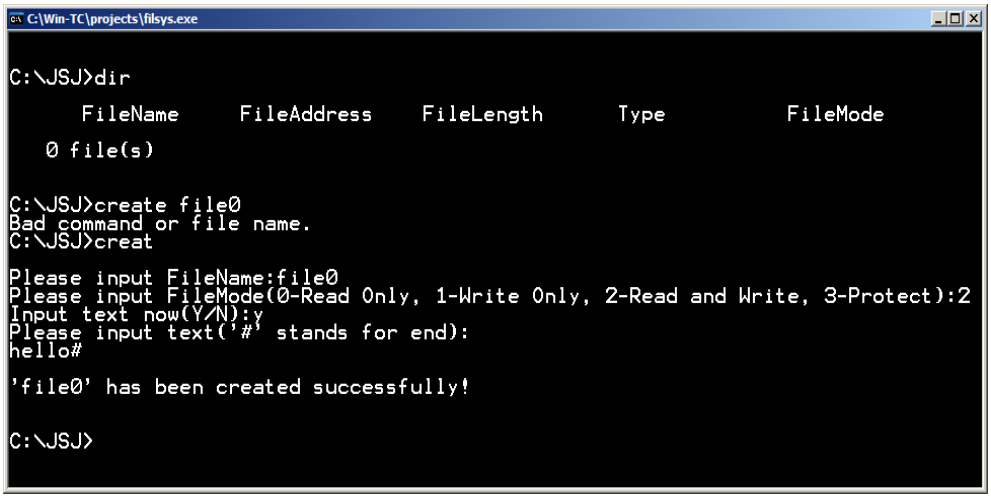


图 2 创建文件界面

（3）删除文件界面，见图3

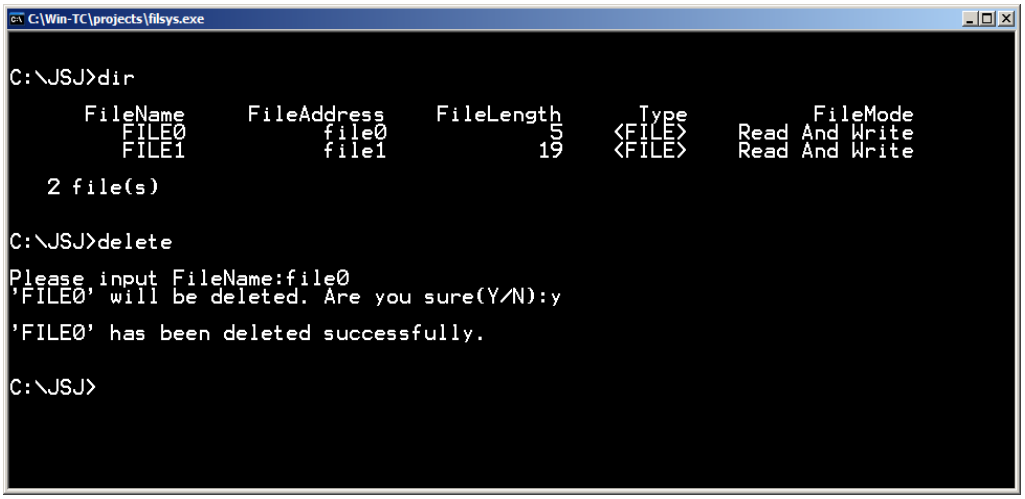


图 3 删除文件界面

（4）打开文件界面，见图4

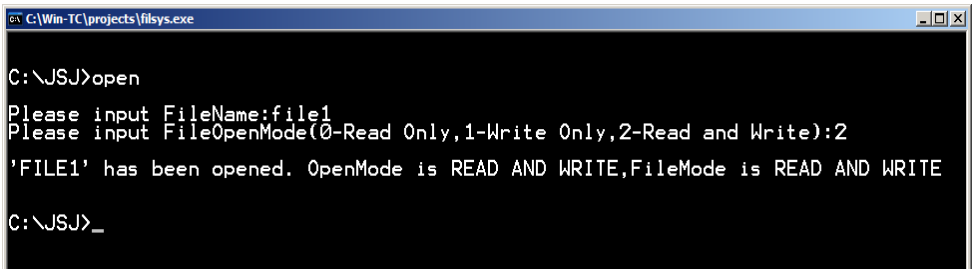


图 4 打开文件界面

（5）修改文件界面，见图5

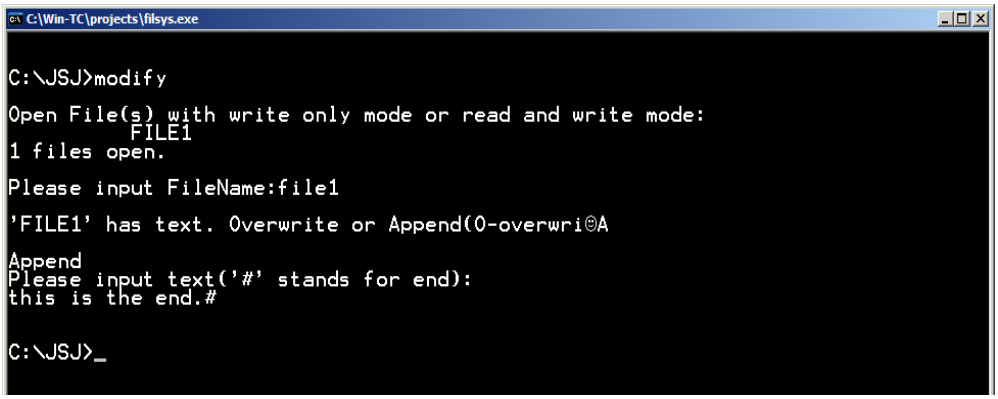


图 5 修改文件界面

（6）读文件界面，见图6

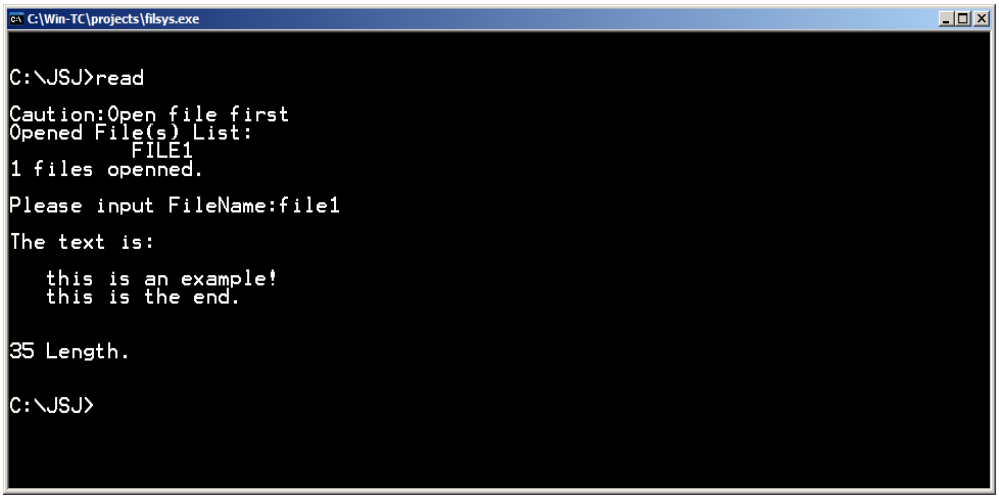


图 6 读文件界面