学习完《操作系统》课程，通过一个具体的课程设计实践，可使学生对操作系统原理有更进一步的理解与应用，通过设计对操作系统有一个实现的尝试和创新的思维，同时发挥团队协作精神和个人创造能力。

**选题：**以下两题可任选一个，也可以自由选择其他操作系统相关的内容做课程设计，难度不能与所出题目差别太大，也可以进行LINUX的代码分析。有能力的同学可以加上界面的设计。

**分组**：每组不可超过两人，一人也可以。每人设计系统中的一部分。

**要求：**界面自行设计，语言自行选择。每组需写一份课程设计报告，内容包括：小组成员及分工，系统采用的语言、支撑平台、程序的模块功能分析、程序的流程图、程序功能段的说明及代码解释；实验体会与软件评价。报告提交后，将系统代码及说明压缩成文件以所有组员学号组合命名发给班长，再由班长统一发给我。我的邮箱是（xchliu@chd.edu.cn）。（如：2016240303＋0302（班号＋两个组员学号）。

**题目一**：支持多个进程（线程）并发运行的简单进程（线程）管理模拟系统

1．实验内容

学习进程管理的设计与实现，学习和运用操作系统原理，设计一个操作系统子系统的模拟系统。通过该系统的设计调试可增加对操作系统实现的感知性。设计一个允许n个进程并发运行的进程管理的模拟系统。该系统包括简单的进程控制、同步及通信机构，其进程调度算法可任意选择，除此之外，还需要用到银行家算法来预防进程的死锁。分析系统所需的数据结构、算法的流程以及划分各个功能模块。

2．实验原理

进程管理包括进程的控制、阻塞、唤醒与撤消。进程在运行过程中状态在不断的变化。进程运行过程中，因为竞争资源而需对它们进行同步控制。所有这些在操作系统中用数据结构PCB来记录，系统通过PCB控制进程的运行。

在单处理机系统中，多道程序的引入，需要进程的调度程序来选择一个进程进行运行。比如常用的先来先服务、短进程优先和优先级优先等等，也可以选择它们的结合调度算法。

**题目二：**文件系统设计

1. 实验内容

通过一个简单的二级文件系统设计，加深对文件系统的内部功能以及内部实现的理解。要求模拟采用二级目录结构的磁盘文件系统的文件操作，能实现以下几条命令，用输入命令来模拟用户程序中调用的文件操作：

Login 用户登录

Dir 列文件目录（列出文件名、物理地址、保护码和文件长度）

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

源文件可以进行读写保护

2．实验原理

文件系统管理中用到的数据结构有：

①首先应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。

②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。

**例：**//下例只是参考（每组可自行设计数据结构和程序功能，只需能演示进程管理的功能，如用多线程）

**1．进程管理模拟系统**。

（1）问题描述

本系统的同步机构采用的信号量上的P、V操作的机制；控制机构包括阻塞和唤醒操作；时间片中断处理程序处理模拟的时间片中断；进程调度程序负责为各进程分配处理机。

系统中设计了3个并发进程．它们之间有如下同步关系：3个进程需要互斥使用临界资源s2，进程1和进程2又需互斥使用临界资源s1。本系统在运行过程中随机打印出各进程的状态变换过程，系统的调度过程及公共交量的变化情况。

（2）算法

系统为进程设置了5种运行状态：e——执行态；r——高就绪态；t——低就绪态(执行进程因时间片到限而转入)：w——等待态；c——完成态。各进程的初始状态均设置为r。

系统分时执行各进程。通过产生随机数x来模拟时间片（每一个时间片并不相同）。当进程processl访问随机数x时，若x≥0．33；当进程proccss2访问x时，若x<0.33或x≥0.66；当进程process3访问x时，若x<0.66，则分别认为各进程的执行时间片到限，产生“时间片中断”面转入低就绪态t。

进程调度算法采用剥夺式最高优先数法。各进程的优先数通过键盘输入予以静态设置。调度程序每次总是选择优先数最小(优先权最高)的就绪进程投入执行。先从r状态进程中选择，再从t状态进程中选择。当现行进程唤醒某个等待进程，且被唤醒进程的比先数小于现行进程时，则剥夺现行进程的执行权。

各进程在使用临界资源s1和s2时，通过调用信号量sem1和sem2上的P、V操作来实现同步。阻塞和唤醒操作负责完成从进程的执行态到等待态以及从就绪态的转变。

系统启动后，在完成必要的系统初始化后便执行进程调度程序。当执行因“时间片中断”，或者被排斥使用临界资源，或唤醒某个等待进程时，立即进行进程调度。当3个进程都处于完成状态后，系统退出运行。

（3）数据结构

①每个进程一个PCB，内容：

id　　　　　进程标识

status　　　 进程状态

priority　　　进程优先数

nextwr　　　等待队链指针，指示在同一信号量上等待的下一个进程的标识。

②信号量。对应于临界资源s1和s2分别有sem1和sem2，均为互斥号量，内容包括：

value　　　　信号量值，初值为1

firstwr　　　等待 链指针，指示在该信号量上第一个等待进程的标识数。

③保留区。用数组saveaera[3][4]表示。即每个进程都有一个大小为4个单元的保留区，用来保存被“中断”时的现场信息，如通用寄存器的内容和断点地址等。

④全程变量。

exe　　　　　　　执行进程指针，其值为进程标识数

i　　　　　　　　用来模拟一个通用寄存器。

（4）用Turbo　C编写的程序代码：

#include <stdlib.h>

#include <time.h>

#include <stdio.h>

#define TRUE 1

#define FALSE 0

#define MAXPRI 100

#define NIL -1

struct{

int id;

char status;

int nextwr; //\*等待链指针，指示在同一信号量上等待的下一个进行进程的标识符\*

int priority;

}pcb[3];

struct

{

int value;

int firstwr; /\*等待链首指针，指示该信号量上第一个等待进程的标识数\*/

}sem[2];

char savearea[3][4],addr;

int i,s1=0,s2=0,seed,exe=NIL;

init( )

{ int j;

for(j=0;j<3;j++)

{

pcb[j].id=j;

pcb[j].status='r';

pcb[j].nextwr=NIL;

printf("\nprocess%dpriority?",j+1);

scanf("%d",&i);

pcb[j].priority=i;

}

sem[0].value=1; sem[0].firstwr=NIL;

sem[1].value=1; sem[1].firstwr=NIL;

for(i=0;i<3;i++)

for(j=0;j<4;j++)

savearea[i][j]='0';

}

float random()

{

float m;

srand(time(0));

m=(1+rand()%3)%3-0.1;

printf("random m=%f\n",m);

getchar();

return(m);

}

timeint(ad) /\*time slice interupt\*/

char ad;

{ float x;

x=random();

if((x<0.33)&&(exe==0))return(FALSE);

if((x<0.66)&&(exe==1))return(FALSE);

if((x<1.0)&&(exe==2))return(FALSE);

savearea[exe][0]=i;

savearea[exe][1]=ad;

pcb[exe].status='t';

printf("This times slice interrupt process%d enter into ready\n",exe+1);

exe=NIL;

return(TRUE);

}

scheduler( )

{

int pd;

if((pd=find())==NIL&&exe==NIL)

return(NIL); /\*quit system\*/

if(pd!=NIL)

{

if(exe==NIL)

{

pcb[pd].status='e';

exe=pd;

printf("proccess%d is executing\n",exe+1);

}

else if(pcb[pd].priority<pcb[exe].priority)

{

pcb[exe].status='r';

printf("process%d enter into ready\n",exe+1);

pcb[pd].status='e';

exe=pd;

printf("process%d is executing\n", exe+1);

}

}

i=savearea[exe][0];

addr=savearea[exe][1];

return(exe);

}

find( )

{

int j,pd=NIL,w=MAXPRI;

for(j=0;j<3;j++)

{

if(pcb[j].status=='r')

if(pcb[j].priority<w)

{

w=pcb[j].priority;

pd=j;

}

}

if (pd==NIL)

for(j=0;j<3;j++)

{

if(pcb[j].status=='t')

if(pcb[j].priority<w)

{

w=pcb[j].priority;

pd=j;

}

}

return(pd);

}

p(se,ad)

int se;

char ad;

{

if(--sem[se].value>=0)

return(FALSE);

block(se);

savearea[exe][0]=i;

savearea[exe][1]=ad;

exe=NIL;

return(TRUE);

}

block(se)

int se;

{

int w;

printf("process%d is blocked\n",exe+1);

pcb[exe].status='w';

pcb[exe].nextwr=NIL;

if((w=sem[se].firstwr)==NIL)

sem[se].firstwr=exe;

else

{

while(pcb[w].nextwr!=NIL)

w=pcb[w].nextwr;

pcb[w].nextwr=exe;

}

}

v(se,ad)

int se;

char ad;

{

if(++sem[se].value>0)

return(FALSE);

wakeup(se);

savearea[exe][1]=ad;

savearea[exe][0]=i;

return(TRUE);

}

wakeup(se)

int se;

{

int w;

w=sem[se].firstwr;

if(w!=NIL)

{

sem[se].firstwr=pcb[w].nextwr;

pcb[w].status='r';

printf("process%d is waken up\n",w+1);

}

}

process1()

{

if(addr=='a') goto a1;

if(addr=='b') goto b1;

if(addr=='c') goto c1;

if(addr=='d') goto d1;

if(addr=='e') goto e1;

if(addr=='f') goto f1;

for(i=0; i<3;i++)/\*如果程序执行超过５次，则\*/

{

printf("process1 calls P on the semaphore 1\n");

if(p(0,'a')) break; /\*process 1 is blocked\*/

a1: printf("process1 is executing in the cretical section 1\n");

if(timeint('b')) break; /\*time silce interrupt\*/

b1: printf("s1=%d\n",++s1);

printf("process1 calls V on semaphorel and quit cretical section 1.\n");

if(v(0,'c')) break; /\*wake up a blocked process\*/

c1: printf("process1 calls P on semaphorel 2.\n");

if(p(1,'d')) break;

d1: printf("process1 is execting cretical section 2.\n");

if(timeint('e')) break;

e1: printf("s2=%d\n",++s2);

printf("process1 calls V on semaphore2 and quit cretical section2.\n");

if(v(1,'f')) break; /\*wake up a block process\*/

f1: printf("process1 cycle count=%d\n",i);

}

if(i<3) return;

eexit(0);

}

process2()

{

if(addr=='a') goto a2;

if(addr=='b') goto b2;

if(addr=='c') goto c2;

if(addr=='d') goto d2;

if(addr=='e') goto e2;

if(addr=='f') goto f2;

for(i=1;i<6;++i)

{

printf("process2 call P on semaphore2\n");

if(p(1,'a')) break; /\*process2 is blocked\*/

a2: printf("process2 is executing on the cretical setion2\n");

if(timeint('b')) break;

b2: printf("s2=%d\n",++s2);

printf("process2 is calls V on semaphore2 and quit cretical section2.\n");

if(v(1,'c')) break; /\*wake up a blocked process\*/

c2: printf("process2 call P on semphore 1.\n");

if(p(0,'d')) break; /\*process2 is blocked\*/

d2: printf("process2 is executing cretical setion1\n");

if(timeint('e')) break;

e2: printf("s1=%d\n",++s1);

printf("process2 call V on semaphorel and quit cretical setion2\n");

if(v(0,'f')) break; /\*wkup up a block process\*/

f2: printf("process2 cycle count=%d\n",i);

}

if(i<6) return;

eexit(1);

}

process3()

{

if(addr=='a') goto a3;

if(addr=='b') goto b3;

if(addr=='c') goto c3;

for(i=1;i<6;++i)

{

printf("process3 call P on semaphore2\n");

if(p(1,'a')) break; /\*process3 is blocked\*/

a3: printf("process3 is executing on its cretical section.\n");

if(timeint('b')) break;

b3: printf("s2=%d\n",++s2);

printf("process3 calls V on semaphore and quit cretical section.\n");

if(v(1,'c')) break; /\*wake up a blocked process \*/

c3: printf("process3 cyclen count=%d\n",i);

}

if(i<6) return;

eexit(2);

}

eexit(n)

int n;

{

pcb[n].status='c';

printf("process%d is completed!\n",n+1);

exe=NIL;

}

main()

{

int j=0;

int k;

char m;

printf("\* \* \* \* process management \* \* \* \*\n\n");

init();

printf("s1=%d, s2=%d\n",s1,s2);

printf("process1,process2,process3 are all in ready!\n");

for(;;)

{

j=j+1;

printf("\n%dth Scheduler Press ENTER to continue\n",j);

//scanf("%s",&m);

getchar();

if((k=scheduler())!=NIL)

{

switch(k)

{

case 0: process1();

break;

case 1: process2();

break;

case 2: process3();

break;

default: printf("process identifer error\n");

break;

}

}

else break;

}

printf("s1=%d, s2=%d\n",s1,s2);

printf("\n\* \* \* \* END \* \* \* \*\n");

}

**2．文件系统**

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,delete,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]->ufdfile[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);}

}