**鲁东大学信息与电气工程学院**

**实验报告**

（ 2022 — 2023 学年第 1 学期）

2220180106

**课程名称**  操作系统

**实验题目** 文件系统

**专 业** 软件工程

**班 级**  2101

**姓 名**  梅英豪

**学 号**  20212203341

2023 年 6 月 10 日

|  |  |  |  |
| --- | --- | --- | --- |
| 实验题目 | 实验6 文件管理 | | |
| 实验类型 | 验证、设计 | 实验日期 | 2023 年 5 月 27 日 |
| 题目来源 | ✔1.必修 2.选修 3.自拟(设计) 4.专题 | | |
| **一、实验目的及要求**  1．**熟悉**Linux文件系统的文件和目录结构；  2．**掌握**文件系统的基本特征；  3．**掌握**常用的文件操作函数；  4．**理解**文件存储空间的管理、文件的物理结构和目录结构以及文件操作的实现；  5．加深对文件系统内部功能和实现过程的**理解**。  **二、实验仪器设备与软件环境**  1．一台装有Windows的微型计算机（Win7或Win10）；  2．在微型计算机上（通过虚拟机）安装了Linux环境；  3．Linux环境下的编辑器（vi/vim、gedit或vscode），编译器（gcc），调试器（gdb）。  **三、**实验内容  1．文件备份实验：编写C程序，模拟实现Linux文件系统的简单I/O流操作：备份文件，将源文件source.dat 备份为target.dat文件。实验要求如下：  (1)使用Ｃ语言库函数实现文件备份；  (2)使用系统调用函数实现文件备份。  2．简单文件系统的模拟：模拟实现一个简单的二级文件管理系统，要求做到以下几点：  (1) 可以实现常用文件目录和文件操作，比如：login用户登录、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、创建文件夹、删除文件夹；  (2) 在列文件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。  四、实验指导  **1、**文件备份实验：  对于实验要求(1)，涉及的C语言库函数有fopen()、fclose()、fread()和fwrite()， 需要执行的操作步骤如下。  (1)使用fopen()函数以只读方式打开想要备份的源文件source，并以只写方式打开想要写入内容的目标文件target。  (2)使用fread()函数循环读取源文件中一个缓冲区大小的内容，然后使用fwrite()函数将读取的内容写入目标文件。  (3)读取与写入完毕后，使用fclose()函数关闭读写文件流。  示例代码如下：  #include <sys/types.h>  #include <stdio.h>  #include <stdlib.h>  int main()  {  char buf;  FILE \*source, \*backup;  printf("This program backup file based on C Library.\n");  source=fopen("./source.dat", "r") ;//打开源文件  backup=fopen("./target.dat", "w") ;//打开目标文件  if(source==NULL)  {  printf("Error in opening file.\n") ;  exit(1) ;  }  if(backup==NULL)  {  printf("Error in creating backup file.\n") ;  exit(1) ;  }  while(fread(&buf, sizeof(buf) , 1, source) ==1)  {  if(!fwrite(&buf, sizeof(buf) , 1, backup) )  {  printf("Error in writing file.\n") ;  exit(1) ;  }  }  if(ferror(source) !=0)  {  printf("Error in reading sourcefile.\n") ;  exit(1) ;  }  else  printf("Success in reading sourcefile.\n") ;  if(fclose(source) )  {  printf("Error in close sourcefile.\n") ;  exit(1) ;  }  else  printf("Success in close sourcefile.\n") ;  if(fclose(backup) )  {  printf("Error in close target file.\n") ;  exit(1) ;  }  else  printf("Success in close target file.\n") ;  exit(0) ;  }  对于实验要求(2)，涉及的与Linux相关的系统调用函数有open()、close()、read()和write()，并且需要执行的操作步骤如下：  (1) 使用open()系统调用函数以只读方式打开想要备份的源文件source，并以只写方式打开想要写入内容的目标文件target。  (2) 使用read()系统调用函数循环读取源文件中一个缓冲区大小的内容，然后使用write() 系统调用函数将读取的内容写入目标文件。  (3) 读取与写入完毕后，使用close()系统调用函数关闭读写文件流。  示例代码如下：  #include <sys/types.h>  #include <stdio.h>  #include <stdlib.h>  #include <fcntl.h>  #include <sys/stat.h>  #define MAXSIZE 1024  int main()  {  char buf[MAXSIZE] ;  int source, backup;  int size;  printf("This program backup file based on Linux system calls.\n") ;  source=open("./source.dat", O\_RDONLY) ;//打开源文件  backup=open("./target.dat", O\_WRONLY |O\_CREAT,0644) ;//打开目标文件  if(source==-1)  {  perror("Error in opening file.\n") ;  exit(1) ;  }  if(backup==-1)  {  perror("Error in opening backup file.\n") ;  exit(1) ;  }  //备份文件  while((size=read(source, buf, MAXSIZE) ) >0)  {  if(write(backup, buf, size) !=size)  {  printf("Error in writing file.\n") ;  exit(1) ;  }  }  if(size<0)  {  printf("Error in reading sourcefile.\n") ;  exit(1) ;  }  else  printf("Success in reading sourcefile.\n") ;  //关闭文件  if(close(source) <0)  {  perror("Error in close sourc file.") ;  exit(1) ;  }  else  printf("Success in close sourcefile.\n") ;  if(close(backup) <0)  {  perror("Error in close target file.") ;  exit(1) ;  }  else  printf("Success in close target file.\n") ;  exit(0) ;  }  **2、**简单文件系统的模拟：  设计思路：本文件系统采用两级目录，其中第一级对应用户账号，第二级对应用户账号下的文件。由于只进行简单的模拟实现，因此不考虑文件共享、文件系统安全及特殊文件等内容。  设计时，首先应确定文件系统的数据结构，即主目录、子目录和活动文件等。主目录和子目录都以链表的形式存放，用户创建的文件以编号形式存储在磁盘上，并且需要在目录中进行登记。  在实现这个文件系统时，需要设计如下数据结构：  (1)磁盘块结构体  typedef struct distTable  {  int maxlength； //容量  int start； //起始地址  int useFlag； //是否被使用  struct distTable \*next； //指向下一磁盘块的指针  } disk Node；  (2) 文件块结构体(即FCB)  typedef struct fileTable  {  char fileName[10] ； //文件名  int strat； //文件在磁盘中的起始地址  int length； //文件内容长度  int maxlength； //文件的最大长度  char fileKind[3] ； //文件的属性——读写方式  struct tm \*timeinfo； //文件相关的时间信息  bool openFlag；//判断是否有进程打开了该文件  } fileTableN；  (3) 用户文件目录(user file directory，UFD)  typedef struct user\_file\_directory  {  struct fileTable \*file； //文件  struct user\_file\_directory \*next；//下一个用户文件目录  } UFD；  (4) 主文件目录(master file directory， MFD) 。  typedef struct master\_file\_directory  {  char userName[10] ； //用户账号  char password[10] ； //用户密码  UFD \*user； //用户文件目录  } MFD；  示例代码：本实验比较复杂，涉及内容较多，实现方案也有多种。下面提供了部分示例代码，并没有实现文件系统的全部功能，而且并不完全运行正确，在某些情况下还可能存在错误，同学们需要在此基础上仔细修改、调试和完善。  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <string.h>  #include <unistd.h>  #include <stdbool.h>  #define MaxUser 100 //定义最大主目录文件  #define MaxDisk 512\* 1024 //模拟最大磁盘空间  #define commandAmount 12 //对文件操作的指令数  //存储空间管理有关结构体和变量  char disk[MaxDisk] ; //模拟512KB的磁盘存储空间  typedef struct distTable //磁盘块结构体  {  int maxlength;  int start;  int useFlag;  struct distTable \*next;  } diskNode;  diskNode \*diskHead;  typedef struct fileTable //文件块结构体， 即FCB  {  char fileName[10] ;  int strat;//文件在磁盘中的起始地址  int length;//文件内容长度  int maxlength;//文件的最大长度  char fileKind[3] ;//文件的属性——读写方式  struct tm \*timeinfo;  bool openFlag;//判断是否有进程打开了该文件  //fileTable \*next；  } fileTableN;  //两级目录结构体  typedef struct user\_file\_directory //用户文件目录  {  //char fileName[10] ；  struct fileTable \*file;  struct user\_file\_directory \*next;  } UFD;  //UFD \*headFile；  typedef struct master\_file\_directory //主文件目录  {  char userName[10] ;  char password[10] ;  UFD \*user;  } MFD;  MFD userTable[MaxUser] ;  int used=0;//定义主文件目录中已有的用户数  //文件管理  void fileCreate(char fileName[] , int length, char fileKind[] ) ;//创建文件  void fileWrite(char fileName[] ) ;//写文件  void fileCat(char fileName[] ) ;//读文件  void fileRen(char fileName[] , char rename[] ) ;//重命名文件  void fileFine(char fileName[] ) ;//查询文件  void fileDir(char UserName[] ) ;//显示某一用户的所有文件  void fileClose(char fileName[] ) ;//关闭已打开的文件  void fileDel(char fileName[] ) ;//删除文件  void chmod(char fileName[] , char kind[] ) ;//修改文件的读写方式  int requestDist(int startPostion, int maxLength) ;//磁盘分配査询  void initDisk() ;//初始化磁盘  void freeDisk(int startPostion) ;//释放磁盘空间  void diskShow() ;//显示磁盘使用情况  //用户管理  void userCreate() ;  int login() ;  int userID=-1;//用户登录的ID号，值为-1时表示没有用户登录  int main()  {  char order[commandAmount][10] ;  strcpy(order[0] , "create") ;  strcpy(order[1] , "rm") ;  strcpy(order[2] , "cat") ;  strcpy(order[3] , "write") ;  strcpy(order[4] , "fine") ;  strcpy(order[5] , "chmod") ;  strcpy(order[6] , "ren") ;  strcpy(order[7] , "dir") ;  strcpy(order[8] , "close") ;  strcpy(order[9] , "return") ;  strcpy(order[10] , "exit") ;  strcpy(order[11] , "df") ;  char command[100] , command\_str1[10] , command\_str2[10] , command\_str3[5] ,command\_str4[3] ;  int i, k, j;  int length;  initDisk() ;//初始化磁盘  for(i=0; i<MaxUser; i++)//初始化用户文件目录的头指针  {  userTable[i] .user=(UFD\*) malloc(sizeof(UFD) ) ;  userTable[i] .user->next=NULL;  }  while(1)  {  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("1、Creat user\n") ;  printf("2、login\n") ;  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("Please choose the function key：>") ;  int choice;  scanf("%d", &choice) ;  if (choice==1) userCreate() ;  else if (choice==2) userID=login() ;  else printf("您的输入有误， 请重新选择\n") ;  while(userID!=-1)  {  printf("---------------------------\n");  printf("create-创建格式：create al1000rw， 将创建名为al、长度为1000字节、可读可写的文件\n");  printf("rm-删除格式：rmal， 将删除名为a 1的文件\n") ;  printf("cat-查看文件内容格式：catal， 将显示文件al的内容\n") ;  printf("write-写入格式：write al\n") ;  printf("fine-查询格式：fine a 1， 将显示文件al的属性\n") ;  printf("chmod-修改格式：chmod alr， 将文件al的权限改为只读方式\n") ;  printf("ren-重命名格式：renal bl， 将文件al改名为bl\n") ;  printf("dir-显示文件格式：dir aaa， 将显示用户aaa的所有文件\n") ;  printf("df-显示磁盘空间使用情况格式：df\n") ;  printf("close-关闭文件格式：close al， 将关闭文件al\n") ;  printf("return-退出用户， 返回到登录界面\n") ;  printf("exit-退出程序\n") ;  printf("----------------------------\n");  printf("please imput your command：>") ;  gets(command) ;  int select;  //command\_str1字符串存储命令的操作类型  for(i=0; command[i] != ' ' &&command[i] !='\0'; i++)  command\_str1[i] =command[i] ;  k=i;  command\_str1[k] ='\0';  for(i=0; i<commandAmount; i++)  {  if(!strcmp(command\_str1, order[i] ) )  {  select=i;  break;  }  }  if(i==commandAmount)  {  printf("您输入的命令有误， 请重新输入\n") ;  continue;  }  //command\_str2字符串存储文件名或用户名  for(i=k+1, k=0; command[i] != ' ' && command[i] !='\0'; i++, k++)  command\_str2[k] =command[i] ;  command\_str2[k] ='\0';  k=i;  UFD \*p;  switch(select)  {  case 0:for(i=k+1, k=0; command[i] != ' '; i++, k++)  command\_str3[k] =command[i] ;  command\_str3[k] ='\0';  k=i;  j=1;  length=0;//初始化文件长度  for(i=strlen(command\_str3) -1; i>=0; i--)  { //把字符串转换为十进制数  length+=(command\_str3[i] -48) \*j;  j\*=10;  }  for(i=k+1, k=0; command[i] !=' '&&command[i] !='\0'; i++, k++)  command\_str4[k] =command[i] ;  command\_str4[k] ='\0';  fileCreate(command\_str2, length, command\_str4) ;break;  case 1:fileDel(command\_str2);break;  case 2:fileCat(command\_str2);break;  case 3:fileWrite(command\_str2) ; break;  case 4:fileFine(command\_str2) ; break;  case 5:for(i=k+1, k=0; command[i] !=' '&&command[i] !='\0'; i++, k++)  command\_str3[k] =command[i] ;  command\_str3[k] ='\0';  chmod(command\_str2, command\_str3) ; break;  case 6:for(i=k+1, k=0; command[i] !='\0'; i++, k++)  command\_str3[k] =command[i] ;  command\_str3[k] ='\0';  fileRen(command\_str2, command\_str3) ; break;  case 7:fileDir(command\_str2);break;  case 8:fileClose(command\_str2);break;  case 9://退出用户之前关闭所有打开的文件  for(p=userTable[userID] .user->next; p!=NULL; p=p->next)  if(p->file->openFlag) p->file->openFlag=false;  system("cls") ;  userID=-1; break;  case 10:exit(0);break;  case 11:diskShow();break;  }  }  }  return 0;  }  /\*从键盘读取按键信息，但不在屏幕上回显该信息\*/  int getch(void)  {  char c;  system("stty -echo") ;  system("stty -icanon") ;  c=getchar() ;  system("stty icanon") ;  system("stty echo") ;  return c;  }  void userCreate()  {  char c;  char userName[10] ;  int i;  if(used<MaxUser)  {  while (getchar ()!='\n');//清空流缓冲区  printf("请输入用户名：") ;  for(i=0; c=getch() ; i++)  {  if(c=='\n') break;  else  userName[i] =c;  //printf("%c"， c) ；  }  userName[i] ='\0';  for(i=0; i<used; i++)  {  if(!strcmp(userTable[i] .userName, userName) )  {  printf("\n") ;  printf("该用户名已存在， 创建用户失败\n") ;  sleep(1) ;  return;  }  }  strcpy(userTable[used].userName, userName) ;  printf("\n") ;  printf("请输入密码：") ;  for(i=0; c=getchar() ; i++)  {  if(c=='\n') break;  else  userTable[used].password[i] =c;  printf("\*") ;  }  userTable[userID].password[i] ='\0';  printf("\n") ;  printf("创建用户成功\n") ;  used++;  sleep(1) ;  }  else  {  printf("创建用户失败， 用户数已达到上限\n") ;  sleep(1) ;  }  }  int login()  {  char name[10] , psw[10] ;  char c;  int i, times;  while (getchar ()!='\n');//清空流缓冲区  printf("请输入用户名：") ;  for(i=0; c=getchar() ; i++)  {  if(c=='\n') break;  else  name[i] =c;  printf("%c", c) ;  }  name[i] ='\0';  for(i=0; i<used; i++)  {  if(!strcmp(userTable[i] .userName, name) )  break;  }  if(i==used)  {  printf("\n您输入的用户名不存在\n") ;  sleep(1) ;  return -1;  }  for(times=0; times<3; times++)  {  memset(psw, '\0', sizeof(psw) ) ;  printf("\n 请输入密码：") ;  for(i=0; c=getchar() ; i++)  {  if(c=='\n') break;  else  psw[i] =c;  printf("\*") ;  }  printf("\n") ;  for(i=0; i<used; i++)  {  if(!strcmp(psw, userTable[i] .password) )  {  printf("用户登录成功\n") ;  sleep(1) ;  break;  }  }  if(i==used)  {  printf("您输入的密码错误， 您还有%d次输入机会\n", 2-times) ;  if(times==2) exit(0) ;  }  else break;  }  return i;  }  void initDisk()  {  diskHead=(diskNode\*) malloc(sizeof(diskNode) ) ;  diskHead->maxlength=MaxDisk;  diskHead->useFlag=0;  diskHead->start=0;  diskHead->next=NULL;  }  int requestDist(int startPostion, int maxLength)  {  int flag=0;//标记是否分配成功  diskNode\*p, \*q, \*temp;  p=diskHead;  while(p)  {  if(p->useFlag==0&&p->maxlength>maxLength)  {  startPostion=p->start;  q=(diskNode\*) malloc(sizeof(diskNode) ) ;  q->start=p->start;  q->maxlength=maxLength;  q->useFlag=1;  q->next=NULL;  diskHead->start=p->start+maxLength;  diskHead->maxlength=p->maxlength-maxLength;  flag=1;  temp=p;  if(diskHead->next==NULL) diskHead->next=q;  else  {  while(temp->next) temp=temp->next;  temp->next=q;  }  break;  }  p=p->next;  }  return flag;  }  void fileCreate(char fileName[] , int length, char fileKind[] )  {  //int i， j；  time\_t rawtime;  int startPos;  UFD \*fileNode, \*p;  for(p=userTable[userID] .user->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  {  printf("文件重名， 创建文件失败\n") ;  sleep(1) ;  return;  }  }  if(requestDist(startPos, length) )  {  fileNode=(UFD\*) malloc(sizeof(UFD) ) ;  fileNode->file=(fileTableN\*) malloc(sizeof(fileTableN) ) ;  //这一步必不可少， 因为file Node里面的指针也需要申请地址， 否则file Node->file指向会出错  strcpy(fileNode->file->fileName, fileName) ;  strcpy(fileNode->file->fileKind, fileKind) ;  fileNode->file->maxlength=length;  fileNode->file->openFlag=false;  time(&rawtime) ;  fileNode->file->timeinfo=localtime(&rawtime) ;  fileNode->next=NULL;  if(userTable[userID].user->next==NULL)  userTable[userID].user->next=fileNode;  else  {  p=userTable[userID].user->next;  while(p->next)p=p->next;  p->next=fileNode;  }  printf("创建文件成功\n") ;  sleep(1) ;  }  else  {  printf("磁盘空间已满或所创建文件超出磁盘空闲容量， 磁盘空间分配失败\n") ;  sleep(1) ;  }  }  void freeDisk(int startPostion)  {  diskNode\*p;  for(p=diskHead; p!=NULL; p=p->next)  {  if(p->start==startPostion)  break;  }  p->useFlag=false;  }  void fileDel(char fileName[] )  {  UFD \*p, \*q, \*temp;  q=userTable[userID].user;  p=q->next;  while(p)  {  if(!strcmp(p->file->fileName, fileName) ) break;  else  {  p=p->next;  q=q->next;  }  }  if(p)  {  if(p->file->openFlag!=true)//先判断是否有进程打开该文件  {  temp=p;  q->next=p->next;  freeDisk(temp->file->strat) ;//磁盘空间回收  free(temp) ;  printf("文件删除成功\n") ;  sleep(1) ;  }  else  {  printf("该文件已被进程打开，删除失败\n") ;  sleep(1) ;  }  }  else  {  printf("没有找到该文件，请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void fileCat(char fileName[] )  {  int startPos, length;  int i,k=0;  UFD \*p, \*q;  q=userTable[userID] .user;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  startPos=p->file->strat;  length=p->file->length;  p->file->openFlag=true;//文件打开标记  printf("\*\*\*\*\*\*\*\*\*\*\*\*\n");  //for(inti=startPos； k<length； i++， k++)  i=startPos;  while(k<length)  {  if(i%50==0)printf("\n") ;//一行多于50个字符后就换行  printf("%c", disk[i] ) ;  i++;  k++;  }  printf("\n\n\*\*\*\*\*\*\n");  printf("%s已被read进程打开， 请用close命令将其关闭\n",p->file->fileName) ;  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void fileWrite(char fileName[] )  {  UFD \*p, \*q;  q=userTable[userID].user;int i, k, startPos;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  if(!strcmp(p->file->fileKind, "r") )//判断文件类型  {  printf("该文件是只读文件， 写入失败\n") ;  sleep(1) ;  return;  }  char str[500] ;  printf("please input content：\n") ;  //gets(str) ；  scanf("%s", str) ;  startPos=p->file->strat;  p->file->openFlag=true;//文件打开标记  p->file->length=strlen(str) ;  if(p->file->length>p->file->maxlength)  {  printf("要写入的字符串的长度大于该文件的总长度， 写入失败\n") ;  sleep(1) ;  return;  }  for(i=startPos, k=0; k<(int) strlen(str) ; i++, k++)  disk[i] =str[k] ;  printf("文件写入成功， 请用close命令将该文件关闭\n") ;  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void fileFine(char fileName[] )  {  UFD \*p,\*q;  q=userTable[userID].user;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  printf("\*\*\*\*\n");  printf("文件名：%s\n", p->file->fileName) ;  printf("文件长度：%d\n", p->file->maxlength) ;  printf("文件在磁盘中的起始地址：%d\n", p->file->strat) ;  printf("文件类型：%s\n", p->file-> fileKind) ;  printf("创建时间：%s\n", asctime(p->file->timeinfo) ) ;  Printf("\*\*\*\*\*\*\*\*\n");  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void chmod(char fileName[] , char kind[] )  {  UFD \*p, \*q;  q=userTable[userID].user;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  strcpy(p->file->fileKind, kind) ;  printf("修改文件类型成功\n") ;  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void fileRen(char fileName[] , char name[] )  {  UFD \*p, \*q;  q=userTable[userID].user;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  while(q->next)  {  if(!strcmp(q->next->file->fileName, name) )  {  printf("您输入的文件名已存在， 重命名失败\n") ;  sleep(1) ;  return;  }  q=q->next;  }  strcpy(p->file->fileName, name) ;  printf("重命名成功\n") ;  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  void fileDir(char userName[] )  {  UFD \*p;  int i, k;  for(i=0; i<MaxUser; i++)  {  if(!strcmp(userTable[i].userName, userName) )  {  k=i; break;  }  }  if(i==MaxUser)  {  printf("没有找到该用户， 请检查输入的用户名是否正确\n") ;  return;  }  else  {  p=userTable[k].user->next;  printf("\*\*\*\*\*\n");  printf("文件名 文件长度 文件在磁盘中的起始地址 文件类型 创建时间\n");  for(; p!=NULL; p=p->next)  printf("%s %d %d %s %s", p->file->fileName,p->file->maxlength, p->file->strat, p->file->fileKind, asctime (p->file->timeinfo) ) ;  printf("\*\*\*\*\*\*\*\*\*\*\n");  }  }  void diskShow()  {  diskNode \*p;  int i=0, unusedDisk=0;  Printf("\*\*\*\*\n");  printf("盘块号起始地址容量(bit)是否已被使用\n");  for(p=diskHead;p!=NULL; p=p->next, i++)  {  if(p->useFlag==false) unusedDisk+=p->maxlength;  printf("%d %d %d %d",i, p->start, p->maxlength, p->useFlag) ;  }  printf("\*\*\*\*\*\n");  printf("磁盘空间总容量：512x1024 bit 已使用：%d bit 未使用：%d bit\n\n",  MaxDisk-unusedDisk, unusedDisk) ;  sleep(1) ;  }  void fileClose(char fileName[] )  {  UFD \*p, \*q;  q=userTable[userID].user;  for(p=q->next; p!=NULL; p=p->next)  {  if(!strcmp(p->file->fileName, fileName) )  break;  }  if(p)  {  p->file->openFlag=false;  printf("%s文件已关闭\n", p->file->fileName) ;  sleep(1) ;  }  else  {  printf("没有找到该文件， 请检查输入的文件名是否正确\n") ;  sleep(1) ;  }  }  六、实验过程（80分）  本实验要求学生按照以下操作步骤（含代码、数据、图表等）完成实验，并提交实验结果截图。  （一）文件备份实验  1．使用C语言库函数备份文件（20分）  解题思路：  打开源文件和目标文件：使用fopen函数分别以只读方式打开源文件和以只写方式打开目标文件。  逐块读取源文件内容：使用fread函数循环读取源文件中一块固定大小的内容，存储在缓冲区中。  将读取的内容写入目标文件：使用fwrite函数将读取的内容写入目标文件。  关闭文件：使用fclose函数关闭源文件和目标文件。  代码：  #include <stdio.h>  #define BUFFER\_SIZE 1024  int main() {  FILE \*source, \*target;  char buffer[BUFFER\_SIZE];  size\_t bytesRead;  // 打开源文件和目标文件  source = fopen("source.dat", "rb");  target = fopen("target.dat", "wb");  if (source == NULL || target == NULL) {  printf("无法打开文件\n");  return 1;  }  // 读取源文件并写入目标文件  while ((bytesRead = fread(buffer, 1, BUFFER\_SIZE, source)) > 0) {  fwrite(buffer, 1, bytesRead, target);  }  // 关闭文件流  fclose(source);  fclose(target);  printf("文件备份完成\n");  return 0;  }  运行结果：  %OLHZ2N7(}@`]LBDJUL`{6A  G%}{N3JU30C1D4)33Z1]5IB  2．使用系统调用备份文件（20分）  解题思路：  打开源文件和目标文件：使用open系统调用函数分别以只读方式打开源文件和以只写方式打开目标文件。这会返回文件描述符。  逐块读取源文件内容：使用read系统调用函数循环读取源文件中一块固定大小的内容，存储在缓冲区中。这需要使用之前获取的源文件的文件描述符。  将读取的内容写入目标文件：使用write系统调用函数将读取的内容写入目标文件。同样，这需要使用之前获取的目标文件的文件描述符。  关闭文件：使用close系统调用函数关闭源文件和目标文件。同样，这需要使用之前获取的文件描述符。  代码：  #include <fcntl.h>  #include <unistd.h>  #include <stdio.h>  #define BUFFER\_SIZE 1024  int main() {  int source, target;  char buffer[BUFFER\_SIZE];  ssize\_t bytesRead;  // 打开源文件和目标文件  source = open("source.dat", O\_RDONLY);  target = open("target.dat", O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);  if (source == -1 || target == -1) {  printf("无法打开文件\n");  return 1;  }  // 读取源文件并写入目标文件  while ((bytesRead = read(source, buffer, BUFFER\_SIZE)) > 0) {  write(target, buffer, bytesRead);  }  // 关闭文件描述符  close(source);  close(target);  printf("文件备份完成\n");  return 0;  }  运行结果：  %OLHZ2N7(}@`]LBDJUL`{6AG%}{N3JU30C1D4)33Z1]5IB  （二）简单文件系统的模拟（40分）  首先需要创建用户，然后登录，登录成功后即可执行各类操作。  解题思路：  文件结构定义：首先定义文件结构体，包含文件名、路径、保护码、文件长度等属性。可以使用结构体数组来维护文件表，记录系统中存在的文件信息。  用户登录：实现登录功能，可以简单地进行模拟登录，例如输入用户名和密码进行验证。  列文件目录：实现dir命令，遍历文件表，输出文件名、物理地址、保护码和文件长度等信息。  创建文件：实现create命令，根据输入的文件名、路径和保护码创建文件。需要检查文件是否已经存在，并进行相应的错误处理。  删除文件：实现delete命令，根据输入的文件名和路径删除文件。需要检查文件是否存在并且未打开，然后从文件表中移除相应的文件。  打开文件：实现open命令，根据输入的文件名和路径打开文件。需要检查文件是否存在，并设置文件的打开状态。  关闭文件：实现close命令，根据输入的文件名和路径关闭文件。需要检查文件是否已经打开，并设置文件的关闭状态。  读取文件：实现read命令，根据输入的文件名和路径读取文件内容。需要检查文件是否已经打开，并进行相应的读取操作。  写入文件：实现write命令，根据输入的文件名和路径写入文件内容。需要检查文件是否已经打开，并进行相应的写入操作。  创建文件夹：实现create\_folder命令，根据输入的文件夹名创建文件夹。  删除文件夹：实现delete\_folder命令，根据输入的文件夹名删除文件夹。  命令解析和交互：在主程序中，通过命令行输入相应的命令，根据命令调用对应的函数进行文件和目录操作。可以使用循环结构，允许用户多次输入命令，直到用户选择退出。  代码：  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_FILENAME\_LENGTH 50  #define MAX\_PATH\_LENGTH 100  #define MAX\_FILE\_COUNT 100  typedef struct {  char filename[MAX\_FILENAME\_LENGTH];  char path[MAX\_PATH\_LENGTH];  int protection\_code;  int file\_length;  int is\_open;  } File;  File file\_table[MAX\_FILE\_COUNT];  int file\_count = 0;  int login() {  // 模拟登录操作  printf("用户登录成功！\n");  return 1;  }  void dir() {  // 列出文件目录  printf("文件名\t\t物理地址\t保护码\t文件长度\n");  for (int i = 0; i < file\_count; i++) {  printf("%s\t\t%s\t%d\t%d\n", file\_table[i].filename, file\_table[i].path,  file\_table[i].protection\_code, file\_table[i].file\_length);  }  }  void create(char \*filename, char \*path, int protection\_code) {  // 创建文件  if (file\_count >= MAX\_FILE\_COUNT) {  printf("文件数量已达上限，无法创建新文件\n");  return;  }  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  printf("文件已存在，无法创建\n");  return;  }  }  File new\_file;  strcpy(new\_file.filename, filename);  strcpy(new\_file.path, path);  new\_file.protection\_code = protection\_code;  new\_file.file\_length = 0;  new\_file.is\_open = 0;  file\_table[file\_count++] = new\_file;  printf("文件创建成功\n");  }  void delete(char \*filename, char \*path) {  // 删除文件  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  // 找到要删除的文件  if (file\_table[i].is\_open) {  printf("文件处于打开状态，无法删除\n");  return;  }  for (int j = i; j < file\_count - 1; j++) {  file\_table[j] = file\_table[j + 1];  }  file\_count--;  printf("文件删除成功\n");  return;  }  }  printf("文件不存在，无法删除\n");  }  void open(char \*filename, char \*path) {  // 打开文件  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  // 找到要打开的文件  if (file\_table[i].is\_open) {  printf("文件已经处于打开状态\n");  return;  }  file\_table[i].is\_open = 1;  printf("文件打开成功\n");  return;  }  }  printf("文件不存在，无法打开\n");  }  void close(char \*filename, char \*path) {  // 关闭文件  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  // 找到要关闭的文件  if (!file\_table[i].is\_open) {  printf("文件已经处于关闭状态\n");  return;  }  file\_table[i].is\_open = 0;  printf("文件关闭成功\n");  return;  }  }  printf("文件不存在，无法关闭\n");  }  void read(char \*filename, char \*path) {  // 读取文件  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  // 找到要读取的文件  if (!file\_table[i].is\_open) {  printf("文件未打开，无法读取\n");  return;  }  printf("读取文件：%s\n", file\_table[i].filename);  // 进行读取操作...  return;  }  }  printf("文件不存在，无法读取\n");  }  void write(char \*filename, char \*path) {  // 写入文件  for (int i = 0; i < file\_count; i++) {  if (strcmp(file\_table[i].filename, filename) == 0 &&  strcmp(file\_table[i].path, path) == 0) {  // 找到要写入的文件  if (!file\_table[i].is\_open) {  printf("文件未打开，无法写入\n");  return;  }  printf("写入文件：%s\n", file\_table[i].filename);  // 进行写入操作...  return;  }  }  printf("文件不存在，无法写入\n");  }  void create\_folder(char \*foldername) {  // 创建文件夹  printf("创建文件夹：%s\n", foldername);  // 进行文件夹创建操作...  }  void delete\_folder(char \*foldername) {  // 删除文件夹  printf("删除文件夹：%s\n", foldername);  // 进行文件夹删除操作...  }  int main() {  char command[20];  char filename[MAX\_FILENAME\_LENGTH];  char path[MAX\_PATH\_LENGTH];  int protection\_code;  if (!login()) {  printf("登录失败\n");  return 1;  }  while (1) {  printf("\n请输入命令：");  scanf("%s", command);  if (strcmp(command, "dir") == 0) {  dir();  } else if (strcmp(command, "create") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  printf("请输入保护码：");  scanf("%d", &protection\_code);  create(filename, path, protection\_code);  } else if (strcmp(command, "delete") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  delete(filename, path);  } else if (strcmp(command, "open") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  open(filename, path);  } else if (strcmp(command, "close") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  close(filename, path);  } else if (strcmp(command, "read") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  read(filename, path);  } else if (strcmp(command, "write") == 0) {  printf("请输入文件名：");  scanf("%s", filename);  printf("请输入文件路径：");  scanf("%s", path);  write(filename, path);  } else if (strcmp(command, "create\_folder") == 0) {  printf("请输入文件夹名：");  scanf("%s", filename);  create\_folder(filename);  } else if (strcmp(command, "delete\_folder") == 0) {  printf("请输入文件夹名：");  scanf("%s", filename);  delete\_folder(filename);  } else if (strcmp(command, "exit") == 0) {  printf("退出文件管理系统\n");  break;  } else {  printf("无效的命令\n");  }  }  return 0;  }  运行结果：  RM)8}B@)MKKWWF859ZC69BW  六．实验结果分析（20分）  1．实验思考  （1）使用系统调用函数open() 、read() 、write() 和close() 实现简单文件备份的原理是什么？  使用open()函数打开原始文件和备份文件。原始文件以只读模式打开，备份文件以写入模式打开。这将返回两个文件描述符，用于后续的读取和写入操作。  使用read()函数从原始文件读取数据。读取的数据可以存储在一个缓冲区中。  使用write()函数将读取的数据写入备份文件。数据从缓冲区写入备份文件。  重复步骤2和步骤3，直到读取完整个原始文件或者读取的数据大小为零。  使用close()函数关闭原始文件和备份文件。关闭文件将释放文件描述符，并确保数据被正确写入备份文件。  （2）使用C语言库函数fopen() 、fread() 、fwrite() 和fclose() 实现简单文件备份的原理是什么？  使用fopen()函数打开原始文件和备份文件。原始文件以二进制只读模式打开（"rb"），备份文件以二进制写入模式打开（"wb"）。这将返回两个文件指针，用于后续的读取和写入操作。  使用fread()函数从原始文件读取数据。指定要读取的数据大小和数量，并将其存储在一个缓冲区中。  使用fwrite()函数将读取的数据写入备份文件。指定要写入的数据大小和数量，并将数据从缓冲区写入备份文件。  重复步骤2和步骤3，直到读取完整个原始文件或者读取的数据大小为零。  使用fclose()函数关闭原始文件和备份文件。关闭文件将确保数据被正确写入备份文件，并释放文件指针。  （3）上述两种方式的区别是什么？  抽象级别：使用系统调用函数（如open()、read()、write()和close()）是直接与操作系统交互的低级别方式。它们提供了对底层系统资源（如文件描述符）的直接控制。而使用C语言库函数（如fopen()、fread()、fwrite()和fclose()）是在系统调用的基础上提供的更高级别的抽象，它们隐藏了底层的细节，提供了更方便的文件操作接口。  错误处理：使用系统调用函数需要显式地处理错误情况。每个系统调用函数在发生错误时通常返回一个特定的错误代码，需要对这些错误代码进行检查和处理。而使用C语言库函数，例如fopen()和fclose()，它们在遇到错误时会返回一个空指针，而后续的读取和写入操作可以使用feof()和ferror()等函数进行错误检查。  可移植性：使用C语言库函数提供的抽象层可以使代码更具可移植性，因为这些库函数会在不同的操作系统上提供相同的接口。这意味着可以使用相同的代码在不同的操作系统上进行文件备份。而使用系统调用函数可能涉及到特定于操作系统的调用和参数，因此在不同的操作系统上可能需要编写不同的代码。  （4）实验2中示例代码中使用了gets()函数，但该函数并不是Linux C语言下的标准函数，因此上述代码在编译时会出现警告，请修改这部分代码，并想办法替代gets()函数。  2．分析、总结  文件系统的问题：  文件系统是操作系统中管理和组织文件的一种机制。  文件系统提供了对文件的创建、读取、写入、删除等操作，以及目录管理、文件保护等功能。  文件系统可以使用库函数或系统调用函数来实现文件操作。  使用系统调用函数和C语言库函数实现文件备份的问题：  使用系统调用函数可以直接调用操作系统提供的底层接口来进行文件备份。  使用C语言库函数则是通过库函数封装的高层接口来实现文件备份。  使用系统调用函数需要使用文件描述符进行操作，而C语言库函数则使用文件指针。  简单文件管理系统的问题：  简单文件管理系统是一个模拟的二级文件系统，提供常见的文件和目录操作。  实现了用户登录、列文件目录、创建文件、删除文件、打开文件、关闭文件、读取文件、写入文件、创建文件夹、删除文件夹等功能。  文件管理系统使用文件结构体数组来维护文件表，记录文件的属性和状态。  通过命令行输入命令来调用相应的操作函数，实现文件和目录的管理。  3．体会  多样的解决途径：针对相同的问题，可以有不同的解决途径和实现方式。例如在文件备份中，可以选择使用C语言库函数或系统调用函数来实现；在简单文件管理系统中，可以选择不同的数据结构和算法来管理文件和目录。  细节的重要性：在处理文件系统和文件操作时，细节是非常重要的。例如在打开和关闭文件时需要注意资源的释放，文件的状态管理，错误处理等。这些细节的处理直接影响到程序的正确性和稳定性。  实践的意义：通过回答你的问题，我不仅加深了对文件系统和文件操作的理解，还提供了具体的代码示例和解决方案。这有助于巩固知识、培养编程能力，并将理论知识应用到实际问题中。 | | | |
| 七、指导教师评语及成绩  教师签名：  **年 月 日**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 优秀 | 良好 | 中等 | 及格 | 不及格 | | | | |