

hzau-1

**实验4**

**存储管理**

姓 名：陈立宇

学 号: 2017310210305

专业班级：信息1703班

课程名：操作系统实验

指导教师：任继平

完成日期：2019.6.18

**一、实验目的**

文件系统是操作系统的一个重要组成部分，也是与用户关系极为密切的部分。学生应独立的用高级语言编写和调试一个简单的文件系统，模拟文件管理的工作过程，从而对各种文件操作命令的实质内容和执行过程有比较深入的了解，掌握它们的实施方法，加深对教材中有关内容的理解。

**二、实验内容**

1. 本次实验设计一个共有10个用户的文件系统，每个用户最多可保存10个文件，一次运行过程中用户可同时打开5个文件。
2. 程序采用二级文件目录，即设置了主文件目录（MFD）和用户文件目录（UFD）。前者应包含文件主（即用户）及他们的目录区指针；后者应给出每个文件主占有的文件目录，即文件名、保护码、文件长度以及它们存放的位置等。另外为打开文件设置了运行文件目录（AFD），在文件打开时应填入打开文件号，本次打开保护码和读写指针等。
3. 为了便于实现，对文件的读写作了简化，在执行读写命令时，只修改读写指针，并不进行实际文件的读写操作。

**三、实验设计**

**任务分析：**

1. 因系统小，文件目录的检索使用了简单的线性搜索，而没有采用Hash等有效算法。
2. 文件保护简单使用了三位保护码，对应于允许读，允许写和允许执行，如下所示：

1 1 1

允许读 允许写 允许执行

如对应位为0，则不允许。

1. 程序中使用的主要数据结构如下：
   1. 主文件目录（MFD）和用户文件目录（UFD）

MFD UFD

用户名1

文件目录指针1

用户名2

文件目录指针2

用户名3

文件目录指针3

┋

┋

文件名

┋

文件长度

保护码

文件名

┋

文件长度

保护码

主文件目录（MFD）和用户文件目录（UFD）

② 打开文件目录（AFD）

打开文件号

打开保护码

读写指针

打开文件目录（AFD）

**概要设计：**

定义文件数据结构

typedef struct file//定义文件数据结构

{

char file\_name[20] ;

bool file\_protect[3] ;

bool open\_file\_protect[3] ; //仅在文件打开时有效

int read , write ; //定义为读写指针

int file\_length ;

struct file \*next ;

} File ;

定义用户与文件的映射

typedef struct x\_map//用户与文件的映射

{

char userName[20] ;

File \*file ;

struct x\_map \*next ;

} Map ;

定义主文件目录

typedef struct mfd//定义主文件目录

{

Map \*head , \*tail ;

} MFD ;

定义打开文件目录

typedef struct afd//打开文件目录

{

File \*head , \*tail ;

int max\_open ;

int current\_open ;

} AFD ;

定义几个功能函数模板：

void initUser(MFD \*mfd) ;//进行用户的初始化

void displayUser(MFD \*mfd) ;//进行系统用户的输出

Map \* queryUser(char userName[] , MFD \*mfd) ;//进行用户的查找，找到则返回用户映射指针

bool createFile(Map \*user , char file\_name[] , bool file\_protect[3] , int file\_length) ;//进行文件的创建操作,成功则返回true ， 否则返回false

bool deleteFile(Map \*user , char file\_name[] , AFD \*afd) ;//进行文件删除操作

bool openFile(Map \*user , char file\_name[] , AFD \*afd , bool open\_file\_protect[]) ;//进行文件打开操作

bool readFile(AFD \*afd , char file\_name[]) ;//进行文件的读操作

bool writeFile(AFD \*afd , char file\_name[]) ;//进行文件的写操作

bool closeFile(AFD \*afd , char file\_name[]) ;//关闭文件

void displayUserFile(Map \*user) ;//进行用户文件的显示

void displayOpenFile(AFD \*afd , Map \*user) ;//显示打开的文件

主函数流程：

建立一个主文件目录，初始化目录，并打印用户名；

循环

{输入登录的用户名，如果用户存在，为用户初始化打开文件目录，并输入想要进行的操作（create/delete/open/close/read/write/exit）}

程序框图：

输入用户名

开始

在MFD中找到该用户？

报告无此用户文件

“try again”

否

是

结束

显示该用户目录表UFD中所有文件

初始化运行

文件表AFD

保存当前文件目录

Say good bye

打印当前文件目录

输入操作命令

是什么命令？

建立文件处理程序

删除文件处理程序

打开文件处理程序

建立文件处理程序

关闭文件处理程序

建立文件处理程序

create delete open bye close read write

文件系统各个函数算法实现流程图

**源代码：**

#include<iostream>

#include<malloc.h>

#include<string.h>

using namespace std ;

typedef struct file//定义文件数据结构

{

char file\_name[20] ;

bool file\_protect[3] ;

bool open\_file\_protect[3] ; //仅在文件打开时有效

int read , write ; //定义为读写指针

int file\_length ;

struct file \*next ;

} File ;

typedef struct x\_map//用户与文件的映射

{

char userName[20] ;

File \*file ;

struct x\_map \*next ;

} Map ;

typedef struct mfd//定义主文件目录

{

Map \*head , \*tail ;

} MFD ;

typedef struct afd//打开文件目录

{

File \*head , \*tail ;

int max\_open ;

int current\_open ;

} AFD ;

void initUser(MFD \*mfd) ;//进行用户的初始化

void displayUser(MFD \*mfd) ;//进行系统用户的输出

Map \* queryUser(char userName[] , MFD \*mfd) ;//进行用户的查找，找到则返回用户映射指针

bool createFile(Map \*user , char file\_name[] , bool file\_protect[3] , int file\_length) ;//进行文件的创建操作,成功则返回true ， 否则返回false

bool deleteFile(Map \*user , char file\_name[] , AFD \*afd) ;//进行文件删除操作

bool openFile(Map \*user , char file\_name[] , AFD \*afd , bool open\_file\_protect[]) ;//进行文件打开操作

bool readFile(AFD \*afd , char file\_name[]) ;//进行文件的读操作

bool writeFile(AFD \*afd , char file\_name[]) ;//进行文件的写操作

bool closeFile(AFD \*afd , char file\_name[]) ;//关闭文件

void displayUserFile(Map \*user) ;//进行用户文件的显示

void displayOpenFile(AFD \*afd , Map \*user) ;//显示打开的文件

int main()

{

MFD \*mfd ;

mfd = (MFD\*)malloc(sizeof(MFD)) ;

if(mfd == NULL)

{

exit(0) ;

}

mfd->head = mfd->tail = NULL ;

initUser(mfd) ;

displayUser(mfd) ;

char userName[20] ;

while(true)

{

cout<<"请输入登录的用户名: " ;

cin>>userName ;

Map \*user ;

user = queryUser(userName , mfd) ;

if(user == NULL)

{

cout<<"对应用户不存在"<<endl;

}

else

{

//为用户初始化打开文件目录

AFD \*afd ;

afd = (AFD\*)malloc(sizeof(AFD)) ;

if(afd == NULL)

{

cout<<"内存空间不足！"<<endl ;

exit(0) ;

}

afd->head = afd->tail = NULL ;

afd->max\_open = 5 ;

afd->current\_open = 0 ;

char command[20] ;

char file\_name[20] ;

bool file\_protect[3] ;

bool open\_file\_protect[3] ;

int file\_length ;

while(true)

{

cout<<userName<<">>请输入你想要进行的操作（create/delete/open/close/read/write/exit）：";

cin>>command ;

//输入命令进行操作

if(strcmp(command , "create") == 0)

{

cout<<"请输入文件名、文件保护码、文件长度：" ;

cin>>file\_name>>file\_protect[0]>>file\_protect[1]>>file\_protect[2]>>file\_length ;

createFile(user , file\_name , file\_protect , file\_length) ;

displayUserFile(user) ;

}

else if(strcmp(command , "delete") == 0)

{

cout<<"请输入你想要删除的文件名：" ;

cin>>file\_name ;

deleteFile(user , file\_name , afd) ;

displayUserFile(user) ;

}

else if(strcmp(command , "open") == 0)

{

cout<<"请输入你想要打开的文件名和对应保护码：" ;

cin>>file\_name>>open\_file\_protect[0]>>open\_file\_protect[1]>>open\_file\_protect[2] ;

openFile(user , file\_name , afd , open\_file\_protect) ;

displayOpenFile(afd , user) ;

}

else if(strcmp(command , "close") == 0)

{

cout<<"请输入你想要关闭的文件名：" ;

cin>>file\_name ;

closeFile(afd , file\_name) ;

displayOpenFile(afd , user) ;

}

else if(strcmp(command , "read") == 0)

{

cout<<"请输入你想要读的文件名：" ;

cin>>file\_name ;

readFile(afd , file\_name) ;

displayOpenFile(afd , user) ;

}

else if(strcmp(command , "write") == 0)

{

cout<<"请输入你想要写的文件名：" ;

cin>>file\_name ;

writeFile(afd , file\_name) ;

displayOpenFile(afd , user) ;

}

else if(strcmp(command , "exit") == 0)

{

break ;

}

else

{

cout<<"No such command \""<< command <<"\""<<endl ;

}

}

}

}

return 0 ;

}

void initUser(MFD \*mfd)

{

//初始化十个不同用户

for(int i = 1 ; i <= 10 ; i++)

{

Map \*m ;

m = (Map\*)malloc(sizeof(Map)) ;

if(m == NULL)

{

exit(0) ;

}

cout<<"请输入用户名 : " ;

cin>>m->userName ;

m->file = NULL ;

m->next = NULL ;

if(mfd->head == NULL)

{

mfd->head = mfd->tail = m ;

}

else

{

mfd->tail->next = m ;

mfd->tail = m ;

}

}

}

void displayUser(MFD \*mfd)

{

Map \*m = NULL ;

m = mfd->head;

cout<<"用户名 : " ;

while(m)

{

cout<<m->userName<<" " ;

m = m->next ;

}

cout<<endl ;

}

Map \* queryUser(char userName[] , MFD \*mfd)

{

Map \*m = NULL ;

m = mfd->head ;

while(m)

{

if(strcmp(userName , m->userName) == 0)

{

return m ;

}

m = m->next ;

}

return NULL ;

}

bool createFile(Map \*user , char file\_name[] , bool file\_protect[3] , int file\_length)

{

File \*file ;

file = (File\*)malloc(sizeof(File)) ;

if(file == NULL)

{

return false ;

}

//进行文件的初始化

strcpy(file->file\_name , file\_name) ;

file->file\_protect[0] = file\_protect[0] ;

file->file\_protect[1] = file\_protect[1] ;

file->file\_protect[2] = file\_protect[2] ;

file->file\_length = file\_length ;

file->read = file->write = 0 ;

file->next = NULL ;

if(user->file == NULL)

{

user->file = file ;

}

else

{

File \*op , \*preOp = NULL ;

op = user->file ;

//查找是否存在同名文件

while(op)

{

if(strcmp(op->file\_name , file->file\_name) == 0)

{

cout<<"文件名 "<<file->file\_name<<" 已经存在了 ! "<<endl ;

return false ;

}

preOp = op ;

op = op->next ;

}

preOp->next = file ;

}

}

void displayUserFile(Map \*user)

{

cout<<"用户 "<<endl<<user->userName<<" 对应的文件有:"<<endl ;

File \*file = NULL ;

file = user->file ;

while(file)

{

cout<<"文件名："<<file->file\_name<<" 文件保护码："<<file->file\_protect[0]<<" "<<file->file\_protect[1]<<" "<<file->file\_protect[2]<<" 文件长度："<<file->file\_length<<endl ;

file = file->next ;

}

}

bool deleteFile(Map \*user , char file\_name[] , AFD \*afd)

{

File \*file = NULL , \*prefile = NULL , \*temp ;

file = afd->head ;

//在打开文件中查找

while(file)

{

if(strcmp(file\_name , file->file\_name) == 0)

{

cout<<"文件\""<<file\_name<<"\"已打开,请先关闭它！\n" ;

return false ;

}

file = file->next ;

}

file = user->file ;

//在文件中进行查找

while(file)

{

if(strcmp(file\_name , file->file\_name) == 0)

{

if(file == user->file)

{

temp = file ;

user->file = file->next ;

}

else

{

temp = file ;

prefile->next = file->next ;

}

delete temp ;

return true ;

}

prefile = file ;

file = file->next ;

}

if(prefile->next == NULL)

{

cout<<"用户 "<<user->userName<<" 没有文件 \""<<file\_name<<"\""<<endl;

}

return false ;

}

bool openFile(Map \*user , char file\_name[] , AFD \*afd , bool open\_file\_protect[])

{

File \*file = NULL ;

file = user->file ;

while(file)

{

if(strcmp(file->file\_name , file\_name) == 0)

{

break ;

}

file = file->next ;

}

if(file)

{

File \*xfile ;

xfile = (File\*)malloc(sizeof(File)) ;

if(xfile == NULL)

{

return false ;

}

\*xfile = \*file ;

//根据文件的权限进行打开权限的赋值

if(xfile->file\_protect[0] >= open\_file\_protect[0])

{

xfile->open\_file\_protect[0] = open\_file\_protect[0] ;

}

else

{

cout<<"没有读的权力！"<<endl;

return false ;

}

if(xfile->file\_protect[1] >= open\_file\_protect[1])

{

xfile->open\_file\_protect[1] = open\_file\_protect[1] ;

}

else

{

cout<<"没有写的权力！"<<endl;

return false ;

}

if(xfile->file\_protect[2] >= open\_file\_protect[2])

{

xfile->open\_file\_protect[2] = open\_file\_protect[2] ;

}

else

{

cout<<"没有执行的权力！"<<endl;

return false ;

}

xfile->next = NULL ;

if(afd->head == NULL)

{

afd->head = afd->tail = xfile ;

afd->current\_open += 1 ;

}

else if(afd->current\_open < afd->max\_open)

{

afd->tail->next = xfile ;

afd->tail = xfile ;

afd->current\_open += 1 ;

}

else

{

cout<<"打开的文件太多了，无法再打开！" <<endl ;

return false ;

}

}

else

{

cout<<"对应文件/"<<file\_name<<"/不存在！"<<endl ;

return false ;

}

}

bool closeFile(AFD \*afd , char file\_name[])

{

File \*file = NULL , \*preFile = NULL , \*temp = NULL ;

//在打开文件链表中进行查找

file = afd->head ;

while(file)

{

if(strcmp(file->file\_name , file\_name) == 0)

{

if(file == afd->head)

{

if(file == afd->tail)

{

temp = file ;

afd->head = afd->tail = NULL ;

}

else

{

temp = file ;

afd->head = file->next ;

}

}

else if(file == afd->tail)

{

temp = file ;

preFile->next = NULL ;

afd->tail = preFile ;

}

else

{

temp =file ;

preFile->next = file->next ;

}

delete temp ;

return true ;

}

preFile = file ;

file = file->next ;

}

cout<<"对应文件不存在！"<<endl ;

return false ;

}

bool readFile(AFD \*afd , char file\_name[])

{

File \*file = NULL ;

file = afd->head ;

while(file)

{

if(strcmp(file->file\_name , file\_name) == 0)

{

if(file->open\_file\_protect[0])

{

file->read++ ;

return true ;

}

else

{

cout<<"没有读的权力！"<<endl ;

return false ;

}

}

file = file->next ;

}

cout<<"对应文件不存在！"<<endl ;

return false ;

}

bool writeFile(AFD \*afd , char file\_name[])

{

File \*file = NULL ;

file = afd->head ;

while(file)

{

if(strcmp(file->file\_name , file\_name) == 0)

{

if(file->open\_file\_protect[1])

{

file->write++ ;

return true ;

}

else

{

cout<<"没有写的权力！"<<endl ;

return false ;

}

}

file = file->next ;

}

cout<<"对应文件不存在！"<<endl ;

return false ;

}

void displayOpenFile(AFD \*afd , Map \*user)

{

cout<<"用户 "<<endl<<user->userName<<" 打开的文件有: "<<endl ;

File \*file ;

file = afd->head ;

while(file)

{

cout<<"文件名："<<file->file\_name<<" 文件保护码："<<file->open\_file\_protect[0]<<" "<<file->open\_file\_protect[1]<<" "<<file->open\_file\_protect[2]<<" 文件长度："<<file->file\_length;

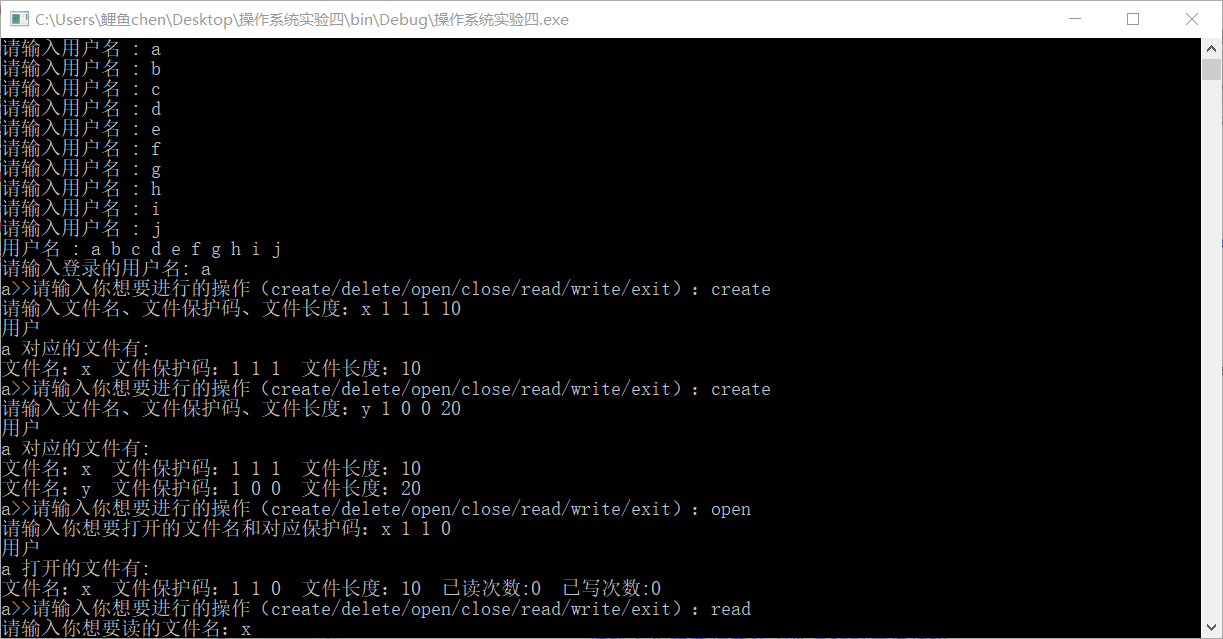
cout<<" 已读次数:"<<file->read<<" 已写次数:"<<file->write<<endl ;

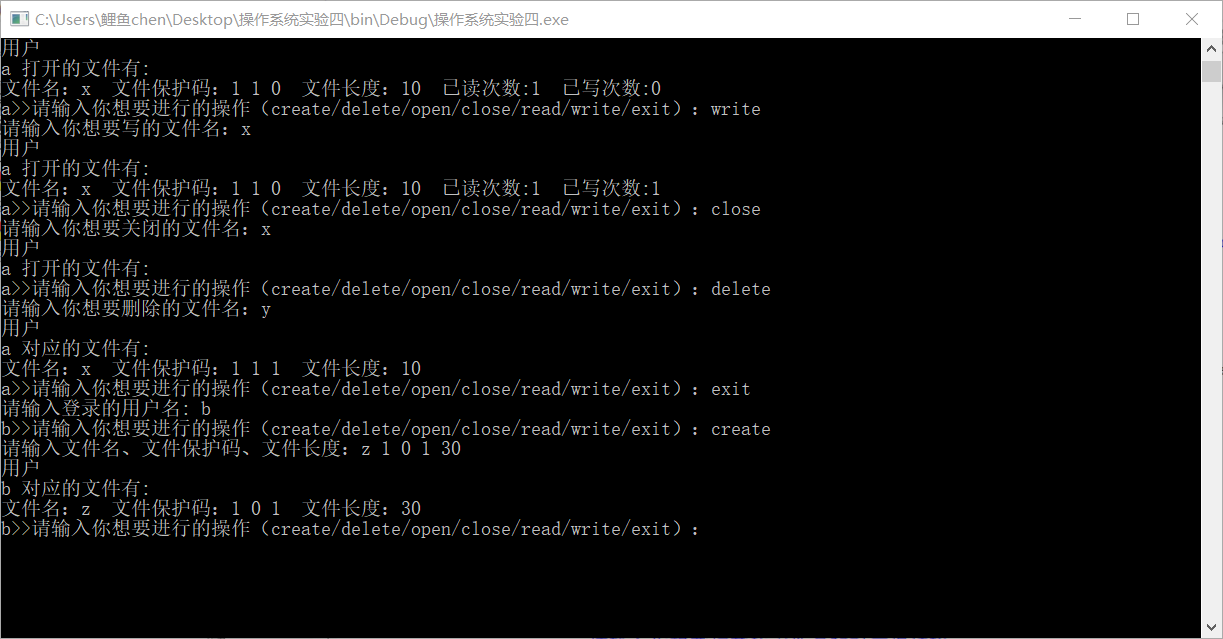
file = file->next ;

}

}

**输入输出示例：**





**四、实验心得**

在这次设计中，我对于文件属性，文件命名，文件操作，文件存取方式有了更加深刻的理解，其中令我印象最为深刻的是文件目录的管理。其中的用户跟文件的映射索起到了非常重要的作用。文件控制块中包含了有关文件存取控制的信息，文件结构的信息，文件使用的信息，文件管理的信息。查找文件时，根据用户名和文件名找到文件控制块，再通过控制块获得相应的信息。为了减少检索文件访问的物理速度，文件系统把文件目录项中的文件名和其他管理信息分开。后者单独组成定长的一个数据结构，称为索引节点。把文件控制块的内容与索引节点分开，不仅加快了目录检索速度，而且，便于实现文件共享，有利于系统的控制和管理。