**北京邮电大学计算机学院**

**2022-2023 学年第 2 学期项目总结报告**

**（每个项目小组一份）**

**课程名称：  *操作系统原理***

**项目名称： 实验六 Linux文件系统调用**

**项目完成人：**

**姓名：\_万志恒 学号： 2021212413**

**姓名： 李伟泽 学号： 2021211462**

**姓名： 唐子潇 学号： 2021211460**

**指导教师： 赵方**

**日 期： 2023 年 6 月 1 日**

**一 . 项目目的和要求**

1.掌握 linux 提供的文件系统调用的使用方法；

2.熟悉文件系统的系统调用用户接口；

3.了解操作系统文件系统的工作原理和工作方式。

**二 .项目开发环境**

硬件：PC微型计算机、8核、16.0G内存、467G硬盘

软件：Linux操作系统、Clion、g++

**三 .项目内容**

编写一个文件工具 filetools，使其具有以下功能：

0.退出

1.创建新文件

2.写文件

3.读文件

4.修改文件权限

5.查看当前文件权限并退出

提示用户输入功能号，并根据用户输入的功能选择相应的功能。

**四 .项目结果及分析**

源代码：

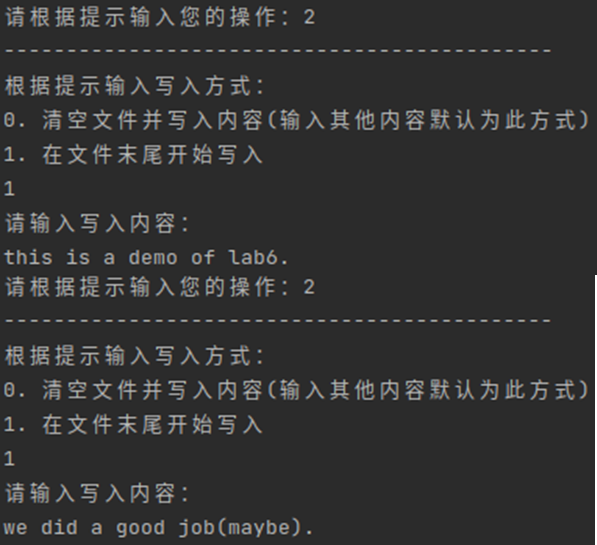
#include <iostream>  
#include <unistd.h>  
#include <sys/stat.h>  
#include <sys/file.h>  
#include <fcntl.h>  
#include <cstring>  
  
using namespace std;  
  
struct my\_file{  
 int fd=-1;  
 int open\_mode=0, write\_mode=0;  
 char path\_name[256]="";  
}cur\_file;  
  
static struct flock lock\_it; /\* 用于加锁的flock对象 \*/  
static struct flock unlock\_it; /\* 用于解锁的flock对象 \*/  
static int lock\_fd = -1;  
  
void my\_lock\_init(char \*path\_name, short int lk\_type);  
void my\_lock\_wait();  
void my\_lock\_release();  
  
void flock\_wait(int opr){  
 /\*if(flock(cur\_file.fd, opr)<0){  
 cout<<"当前文件正被其他进程占用，请稍候..."<<endl;  
 }\*///为什么没用呢。。。  
 while(flock(cur\_file.fd, opr)<0);  
 cout<<"已获取锁"<<endl;  
}  
  
void flock\_rls(){  
 flock(cur\_file.fd, LOCK\_UN);  
 cout<<"已释放锁"<<endl;  
}  
  
int opt();  
void get\_route();  
void open\_file();  
void close\_file();  
void new\_file();  
void write\_file();  
void read\_file();  
void alter\_file\_mode();  
void view\_file\_mode();  
  
int main() {  
 int usr\_choice;  
 do{  
 usr\_choice=opt();  
 }while(usr\_choice>=0&&usr\_choice<=5);  
 return 0;  
}  
  
void my\_lock\_init(char \*path\_name, short int lk\_type) {  
 char lock\_file[1024]; /\* 存放文件名的临时缓存 \*/  
  
 /\* 由于mkstemp会修改参数的最后6个字符，为避免pathname是字符串常量，  
 \* 需要将字符串从pathname copy到临时缓存lock\_file中，然后作为创建唯一临时文件的参数 \*/  
 strncpy(lock\_file, path\_name, sizeof(lock\_file));  
 lock\_fd = mkstemp(lock\_file); /\* 创建唯一的临时文件并打开 \*/  
  
 unlink(lock\_file); /\* 删除指定文件名的文件，如果有进程打开了该文件，  
 \* 会在进程结束后所有指向该文件的描述符都关闭后删除文件。这样可以确保即使程序崩溃，临时文件也会完全消失 \*/  
  
 /\*计算文件长度\*//\*  
 long file\_size = -1;  
 struct stat stat\_buff;  
 if(stat(lock\_file, &stat\_buff)>=0){  
 file\_size=stat\_buff.st\_size;  
 };\*/  
  
 /\* 设置flock对象的写锁属性 \*/  
 lock\_it.l\_type = lk\_type;  
 lock\_it.l\_whence = SEEK\_SET;  
 lock\_it.l\_start = 0;  
 lock\_it.l\_len = 0;  
  
 /\* 设置flock对象的解锁属性 \*/  
 unlock\_it.l\_type = F\_UNLCK;  
 unlock\_it.l\_whence = SEEK\_SET;  
 unlock\_it.l\_start = 0;  
 unlock\_it.l\_len = 0;  
}  
  
void my\_lock\_wait() {  
 int rc;  
 while((rc=fcntl(lock\_fd, F\_SETLKW, &lock\_it))<0){  
 cout<<rc<<endl;  
 };  
}  
  
void my\_lock\_release() {  
 fcntl(lock\_fd, F\_SETLKW, &unlock\_it);  
}  
  
int opt(){  
 char choice;  
 cout << "--------------------------------------------" << endl;  
 cout << "file\_tools" << endl;  
 cout << "0. 关闭当前文件" << endl;  
 cout << "1. 创建新文件（这将关闭现有文件）" << endl;  
 cout << "2. 写文件" << endl;  
 cout << "3. 读文件" << endl;  
 cout << "4. 修改文件权限" << endl;  
 cout << "5. 查看当前文件权限并退出" << endl;  
 cout << "6. 退出file\_tools（其它输入默认为该操作）" << endl;  
 cout << "请根据提示输入您的操作：";  
 cin>>choice;  
 if(choice>'6'||choice<'0') choice='6';  
 cout<<"--------------------------------------------"<<endl;  
/\* while(cur\_file.fd<0&&(choice>='2'&&choice<='5')){  
 open\_file();  
 }  
不能这样写，不然的话如果打开错误就一直循环直到成功打开文件为止，不能直接退出了QAQ\*/  
  
 if(cur\_file.fd<0&&(choice>='2'&&choice<='5')){  
 cout<<"当前未打开任何文件，先打开一个文件吧！"<<endl;  
 open\_file();  
 }  
 if(cur\_file.fd>=0||choice=='1'||choice=='0'||choice=='6'){  
 switch (choice) {  
 case '1':  
 new\_file();  
 break;  
 case '2':  
 write\_file();  
 break;  
 case '3':  
 read\_file();  
 break;  
 case '4':  
 alter\_file\_mode();  
 break;  
 case '5':  
 view\_file\_mode();  
 case '0':  
 close\_file();  
 break;  
 default:  
 cout<<"n\_n bye-bye! n\_n"<<endl;  
 close(cur\_file.fd);  
 cout<<"--------------------------------------------"<<endl;  
 break;  
 }  
 }  
 return choice-'0';  
}  
  
void get\_route() {  
 cout<<"输入文件的路径名：";  
 cin>>cur\_file.path\_name;  
}  
  
void open\_file(){  
 //获取新建文件路径名  
 get\_route();  
  
 //获取打开模式  
 cout<<"根据提示选择打开模式："<<endl;  
 cout<<"0. 只读（输入其他默认为该模式）"<<endl;  
 cout<<"1. 只写"<<endl;  
 cout<<"2. 可读可写"<<endl;  
 cin>>cur\_file.open\_mode;  
 if(cur\_file.open\_mode<0||cur\_file.open\_mode>2) cur\_file.open\_mode=0;  
 cur\_file.write\_mode=0;  
  
 //打开文件  
 cur\_file.fd=open(cur\_file.path\_name, cur\_file.open\_mode/\*|cur\_file.write\_mode\*/);  
 if(cur\_file.fd<0) cout<<"warning：文件打开失败！"<<endl;  
}  
  
void close\_file(){  
 if(cur\_file.fd<0){  
 cout<<"当前未打开任何文件！"<<endl;  
 return;  
 }else{  
 close(cur\_file.fd);  
  
 //对cur\_file全局变量重新初始化  
 cur\_file.fd=-1;  
 cur\_file.open\_mode=0;  
 cur\_file.write\_mode=0;  
 cout<<"已关闭"<<cur\_file.path\_name<<endl;  
 strcpy(cur\_file.path\_name, "");  
 }  
}  
  
void new\_file() {  
 //关闭已打开的文件  
 close(cur\_file.fd);  
 //获取新建文件路径名  
 get\_route();  
 //获取创建模式  
 cout<<"根据提示选择创建模式（输入其他默认为退出）："<<endl;  
 cout<<"0. 只读创建"<<endl;  
 cout<<"1. 只写创建"<<endl;  
 cout<<"2. 可读可写创建"<<endl;  
 cin>>cur\_file.open\_mode;  
 if(cur\_file.open\_mode<0||cur\_file.open\_mode>2) return;  
 //获取文件访问权限  
 mode\_t file\_umask;  
 cout<<"请根据以下规则输入umask值设置文件访问权限："<<endl;  
 cout<<"umask值有三位，第一位代表所有者权限，第二位代表群组权限，第三位代表其他用户权限；"<<endl;  
 cout<<"每一位对应加权值意义如下：4表示读权限，2表示写权限，1表示执行权限，通过对所有权限的相应权值相加得到该位的值。"<<endl;  
 cout<<"请输入新建文件的umask值：";  
 cin>>oct>>file\_umask;  
 //创建新文件  
 cur\_file.fd=open(cur\_file.path\_name, cur\_file.open\_mode|O\_CREAT|O\_EXCL, file\_umask);  
 if(cur\_file.fd<0) cout<<"warning：文件创建错误！"<<endl;  
 else cout<<"文件创建成功！"<<endl;  
}  
  
void write\_file() {  
 //首先判断权限  
 if(cur\_file.open\_mode==O\_RDONLY){  
 cout<<"warning: 文件以只读方式打开，无法写文件。"<<endl;  
 return;  
 }  
  
 //根据写入方式重新打开文件  
 close(cur\_file.fd);  
  
 /\*创建写锁\*//\*  
 my\_lock\_init(cur\_file.path\_name, F\_WRLCK);  
 \*//\*申请锁\*//\*  
 my\_lock\_wait();\*/  
  
 cout << "根据提示输入写入方式（输入其他默认退出操作）：" << endl;  
 cout << "0. 清空文件并写入内容" << endl;  
 cout << "1. 在文件末尾开始写入" << endl;  
 scanf("%d", &cur\_file.write\_mode);  
 if (cur\_file.write\_mode == 1) cur\_file.write\_mode = O\_APPEND;  
 else if(cur\_file.write\_mode == 0) cur\_file.write\_mode = O\_TRUNC;  
 else return;  
 cur\_file.fd=open(cur\_file.path\_name, cur\_file.open\_mode|cur\_file.write\_mode);  
  
 flock\_wait(LOCK\_EX);  
  
 //写入文件  
 char write\_buf[1024] = "";  
 cout << "请输入写入内容：" << endl;  
 getchar();  
 cin.getline(write\_buf, 1024, '\n');  
 write\_buf[strlen(write\_buf)] = '\n';  
 ssize\_t is\_write=write(cur\_file.fd, write\_buf, strlen(write\_buf));  
 if(is\_write<0) cout<<"warning: 写文件错误！"<<endl;  
  
 /\*释放锁\*//\*  
 my\_lock\_release();\*/  
  
 flock\_rls();  
}  
  
void read\_file() {  
 //首先判断权限  
 if(cur\_file.open\_mode==O\_WRONLY){  
 cout<<"warning: 文件以只写方式打开，无法读文件。"<<endl;  
 return;  
 }  
  
 //read函数读完文件后需要重新打开文件才能重新从头开始读  
 close(cur\_file.fd);  
  
 /\*创建读锁\*//\*  
 my\_lock\_init(cur\_file.path\_name, F\_RDLCK);  
 \*//\*申请锁\*//\*  
 my\_lock\_wait();\*/  
  
 open(cur\_file.path\_name, cur\_file.open\_mode);  
  
 flock\_wait(LOCK\_SH);  
  
 char read\_buf;  
 cout<<cur\_file.path\_name<<"文件的内容如下："<<endl;  
 ssize\_t is\_read=read(cur\_file.fd, &read\_buf, 1);  
 while(is\_read==1){//read函数读入EOF返回0  
 cout<<read\_buf;  
 is\_read=read(cur\_file.fd, &read\_buf, 1);  
 }  
 if(is\_read<0) cout<<"warning: 读文件错误！"<<endl;  
  
 cout<<"按下换行键以继续";  
 getchar(); getchar();  
  
 /\*释放锁\*//\*  
 my\_lock\_release();\*/  
  
 flock\_rls();  
}  
  
void alter\_file\_mode() {  
 flock\_wait(LOCK\_EX);  
  
 //获取新的文件访问权限  
 mode\_t file\_umask;  
 cout<<"请根据以下规则输入umask值修改文件访问权限："<<endl;  
 cout<<"umask值有三位，第一位代表所有者权限，第二位代表群组权限，第三位代表其他用户权限；"<<endl;  
 cout<<"每一位对应加权值意义如下：4表示读权限，2表示写权限，1表示执行权限，通过对所有权限的相应权值相加得到该位的值。"<<endl;  
 cout<<"请输入修改后的umask值：";  
 cin>>oct>>file\_umask;  
  
 /\*创建写锁\*//\*  
 my\_lock\_init(cur\_file.path\_name, F\_WRLCK);  
 \*//\*申请锁\*//\*  
 my\_lock\_wait();\*/  
  
  
 //修改文件访问权限  
 int is\_altered=fchmod(cur\_file.fd, file\_umask);  
 if(is\_altered==0) cout<<"文件:"<<cur\_file.path\_name<<"的权限修改成功"<<endl;  
 else if(is\_altered==-1) cout<<"文件:"<<cur\_file.path\_name<<"的权限修改失败"<<endl;  
  
 /\*释放锁\*//\*  
 my\_lock\_release();\*/  
  
 flock\_rls();  
}  
  
void view\_file\_mode() {  
 //获取当前文件状态  
 unsigned int mask=0700, N\_BITS=3;  
 struct stat buff{};  
 const static char \*perm[]={"---","--x","-w-","-wx","r--","r-x","rw-","rwx"};  
 if(stat(cur\_file.path\_name, &buff)!=-1){  
 printf("%s 的权限为：", cur\_file.path\_name);  
 //将st\_mode的每一位（对应二进制的三位）与0b111按位与，三位中各位若为1则表示对应用户拥有该位对应权限）  
 for(int i=3; i; i--){  
 printf("%3s", perm[(buff.st\_mode&mask)>>(i-1)\*N\_BITS]);  
 mask>>=N\_BITS;  
 }  
 cout<<endl;  
 }else printf("warning: %s的权限修改失败\n", cur\_file.path\_name);  
}

结果：

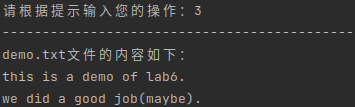
新建文件：



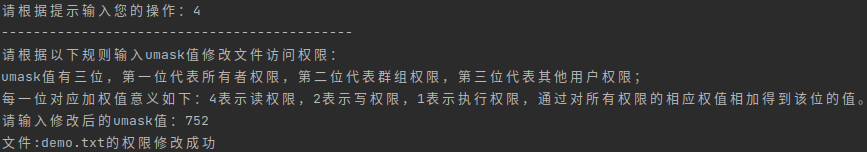
写入文件：

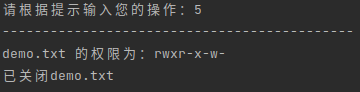


读文件：

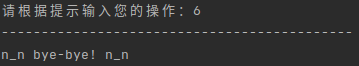


修改文件权限：

查看文件权限并退出：

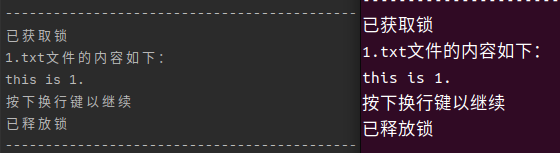


退出文件管理器：

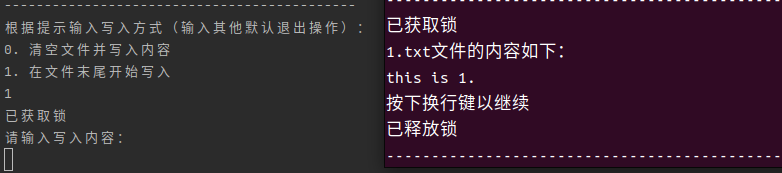
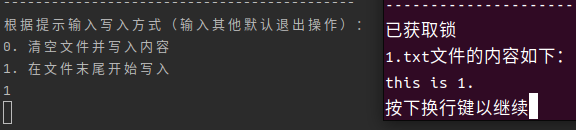


读写锁的实现：

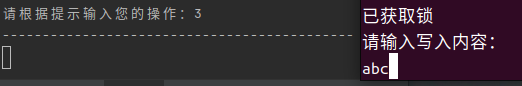
读，随后另一进程读：

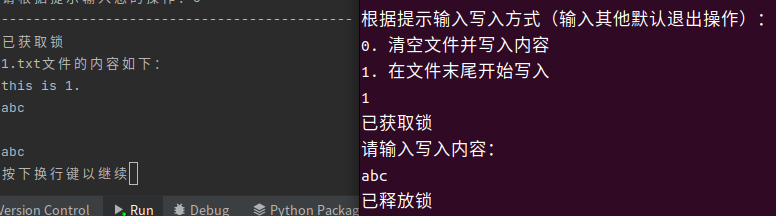


读，随后另一进程写

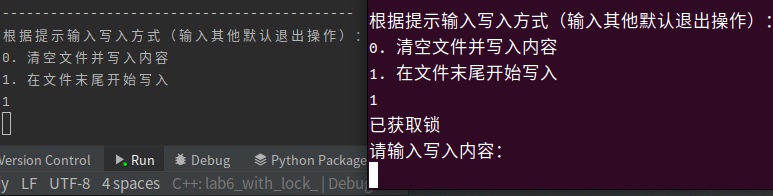


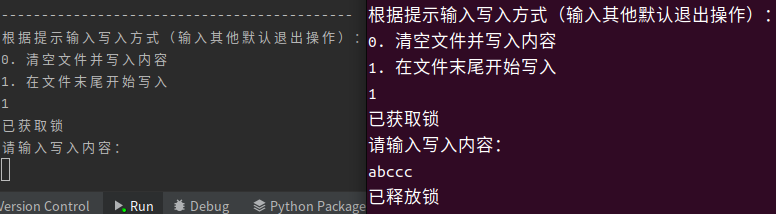
写，随后另一进程读





写，随后另一进程写





结果分析：程序的实现逻辑是通过my\_file()类型对象存储当前打开文件的信息，并根据用户输入对当前文件执行相应操作。其中涉及到如open()、close()、write()、read()、fchmod()等一系列Linux提供的系统调用。此外，我们还通过flock()实现了对文件加读写锁，在对文件执行写操作及修改权限操作时加写锁，读文件时加读锁。

问题分析：通过查阅资料我们了解到Linux系统提供了flock()函数来实现多进程操作文件的读写锁。但在编写了相应程序之后我们无法成功对文件进行加锁。经过调试后我们发现，需要先打开文件再对文件加锁，否则会出错，最终实现了对文件添加读写锁。

**五 .项目人员、进度安排及完成过程**

项目人员：万志恒、李伟泽、唐子潇

万志恒负责opt()、write\_file()、view\_file\_mode()函数代码的编写、对文件加读写锁的实现以及代码整合，李伟泽负责get\_route()、open\_file()、new\_file()、close\_file()函数的编写，唐子潇负责main()、read\_file()、alter\_file\_mode()函数的编写以及程序所涉及数据结构的实现。

**六 . 项目心得及体会**

万志恒：本次实验我通过C++以及其中包含的Linux系统调用函数和同学一起实现了一个简单的文件管理器。我通过write()系统调用实现了写入文件操作，同时深入认识了不同open()函数的参数对应的不同操作；此外在编写查看用户权限的过程中我深入认识了通过stat()函数获取当前文件信息的操作，并学习了文件的stat信息中mask量所代表的权限信息。此次实验锻炼了我的c++编程能力，同时使我更加熟悉Linux中open()、write()、stat()等系统调用。此外我还了解了通过flock()、fcntl()等系统调用实现对文件加读写锁的方式，加深了我对锁机制的理解。

李伟泽：熟悉了linux下文件创建、打开和关闭调用操作；清楚了umask值以及对应的文件权限。

唐子潇：通过本次实验，我了解了操作系统文件系统的工作原理和工作方式，掌握了linux 提供的文件系统调用的使用方法，熟悉了文件系统的系统调用用户接口，尤其对于读取用户文件有了较为深刻的认识。

**七 . 附录（附上项目其他文档，具体内容根据项目要求确定**