|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： 操作系统原理  院系：计算机学院（国家示范性软件工程学院）  班级： 2021211318  姓名： 万志恒  学号： 2021212413  教师： 赵方  成绩：  2023年 春季学期 |
| 实验目的 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。 实验环境 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++   1. **实验任务及内容**   实验内容：  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。  源代码：  本人负责opt()、write\_file()、view\_file\_mode()函数代码的编写、对文件加读写锁的实现及代码整合  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 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 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      实验总结：opt()函数根据用户输入选择操作类型，其逻辑如下：对于读、写、修改权限、查看权限操作，若当前未打开文件则首先进行打开文件操作。根据用户输入调用不同的函数以进行不同的文件操作。只有当输入非操作字符时才会跳出循环结束操作，否则执行完操作后再次读取用户输入并重复执行上述操作。  write()函数会对当前文件打开方式进行判断，若文件以只读方式打开则无法写入；此外open()函数的的第二个参数可以通过设置为O\_APPEND或O\_TRUNC对文件继续写入或清空后写入。  view\_file\_mode()操作的逻辑如下：Linux中用一个三位八进制数来表示一个文件的权限，其中各位分别代表所有者、组、其他的权限，而每一位对应一个三位的二进制数，相应二进制数由高位到低位分别表示写权限、读权限和执行权限。则通过截取八进制的每一个位，并将这些位与代表权限的字符串rwx、rw-、---等进行对应，即可描述文件的权限。  通过flock()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。 实验心得及体会 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。 |