长安大学

《操作系统》

课程设计报告

设计题目：文件系统设计

学　　院：信息工程学院

专　　业：计算机科学与技术（交通信息工程）

姓　　名：瞿强鑫

学　　号：2016902094

2019 年 6 月 6 日

目　录

一、 小组成员及分工

成员：瞿强鑫

分工：软件设计总过程

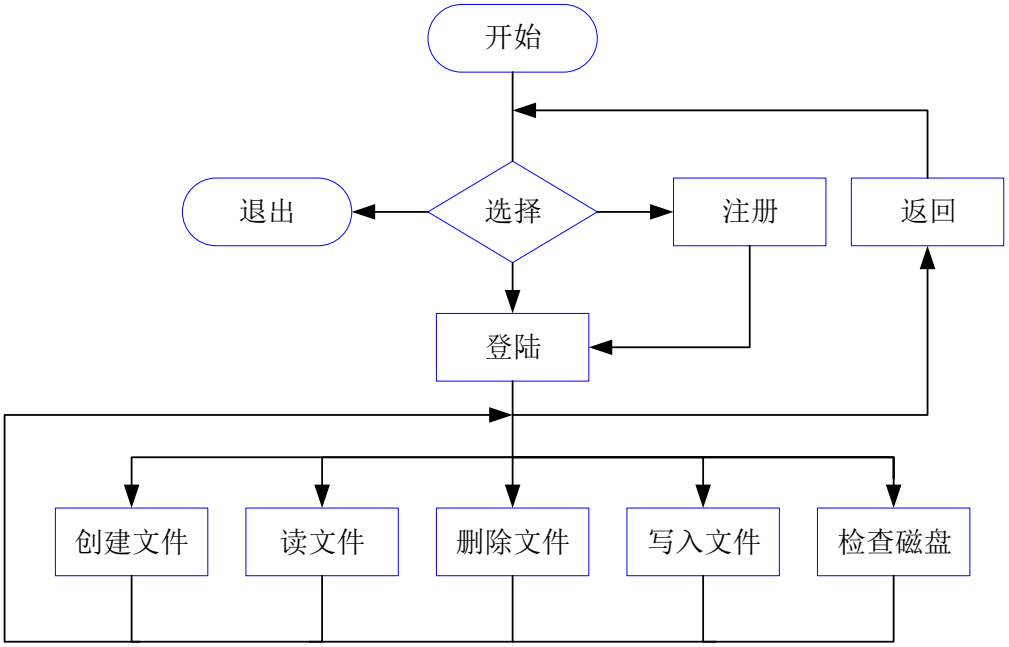
二、 系统采用的开发环境

硬件设备：ASUS 个人电脑

操作系统：Windows 10

编译软件：VC++ 6.0

三、 程序的流程图

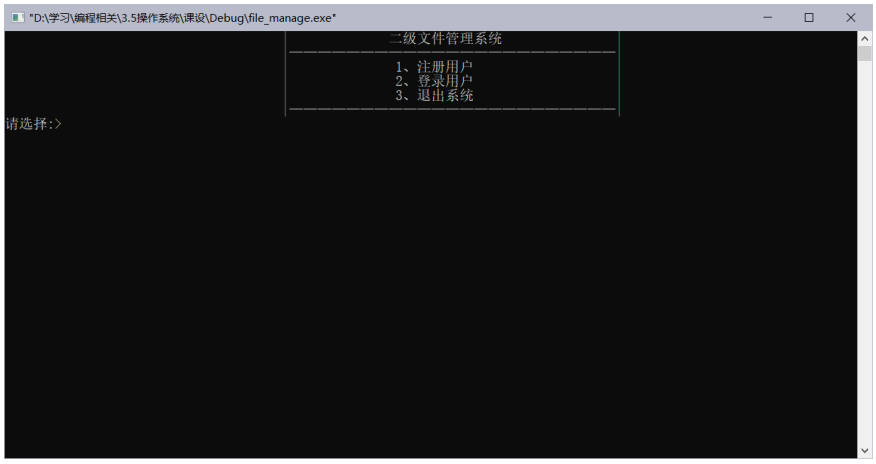


四、 程序功能段的说明及代码解释

一级目录界面代码段：

```
cout << "|                    二级文件管理系统                    |\n";
cout << "|————————————————————————————————————————————————|\n";
cout << "|                      1、注册用户                       |\n";
cout << "|                      2、登录用户                       |\n";
cout << "|                      3、退出系统                       |\n";
cout << "|————————————————————————————————————————————————|\n";
```

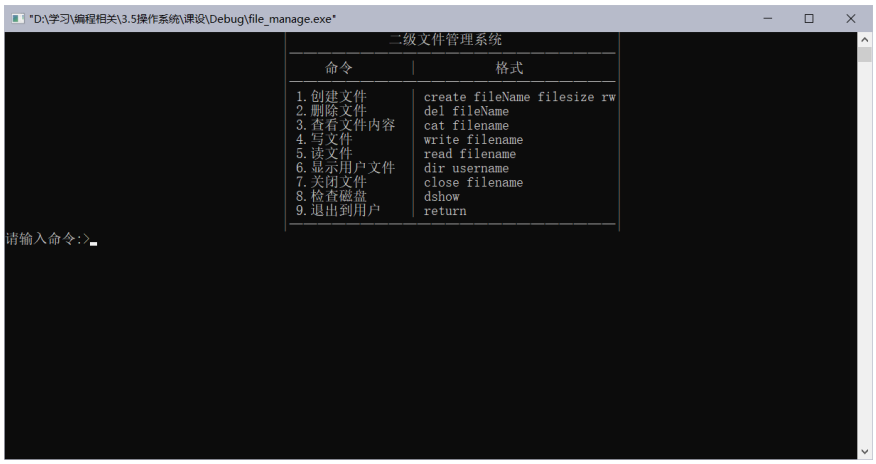
附演示截图：



二级目录界面代码段：

```
cout<<"|                    二级文件管理系统                    |\n";
cout<<"|————————————————————————————————————————|\n";
cout<<"|     命令          |          格式                    |\n";
cout<<"|————————————————————————————————————————|\n";
cout<<"| 1.创建文件        | create fileName filesize rw      |\n";
cout<<"| 2.删除文件        | del fileName                     |\n";
cout<<"| 3.查看文件内容    | cat filename                     |\n";
cout<<"| 4.写文件          | write filename                   |\n";
cout<<"| 5.读文件          | read filename                    |\n";
cout<<"| 6.显示用户文件    | dir username                     |\n";
cout<<"| 7.关闭文件        | close filename                   |\n";
cout<<"| 8.检查磁盘        | dshow                            |\n";
cout<<"| 9.退出到用户      | return                           |\n";
cout<<"|————————————————————————————————————————|\n";
```

附演示截图：



二级目录交互界面皆是通过 switch（）实现功能的选择，具体实现由数组存储命令的字符串如 strcpy(order[0],"create");在 switch 循环里进行

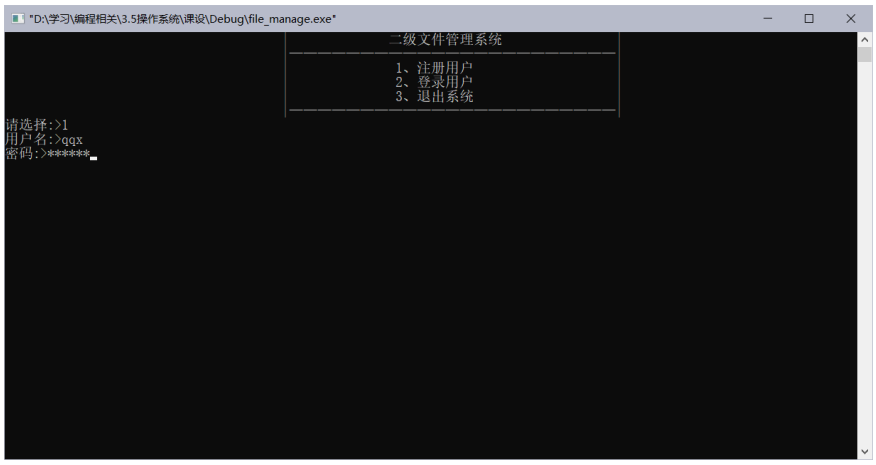
字符串匹配，用 if(!strcmp(command_str1,order[i])){select=i;break;} 将对应命令的序号存入 select 变量，作为 switch 的参数进行循环判断。

用户注册代码段：

```
void userCreate()
{
  char c;
  char userName[10];
  int i;
  if (used < MaxUser)
  {
      cout << "用户名:>";
      for (i = 0; c = getch(); i++)
      {
          if (c == 13) break;
          else
              userName[i] = c;
          printf("%c", c);
      }
      userName[i] = '\0';
      for (i = 0; i < used; i++)
      {
          if (!strcmp(userTable[i].userName, userName))
          {
              cout << "\n";
              cout << "该用户名已存在,创建用户失败\n";
              system("pause");
              return;
          }
      }
      strcpy(userTable[used].userName, userName);
      cout << "\n";
      cout << "密码:>";
      for (i = 0; c = getch(); i++)
      {
          if (c == 13) break;
          else
              userTable[used].password[i] = c;
          printf("*");
      }
      userTable[userID].password[i] = '\0';
      cout << "\n";
      cout << "用户创建成功\n";
```

```
        used++;
        system("pause");
    }
    else
    {
        cout << "创建用户失败，用户已达到上限\n";
        system("pause");
    }
    fflush(stdin);
}
```

附演示截图：



对于注册输入的用户名会遍历存放用户信息的结构体链表，如果该用户名已注册则提示输入用户名已存在，返回重新输入；密码存入结构体链表对应属性。

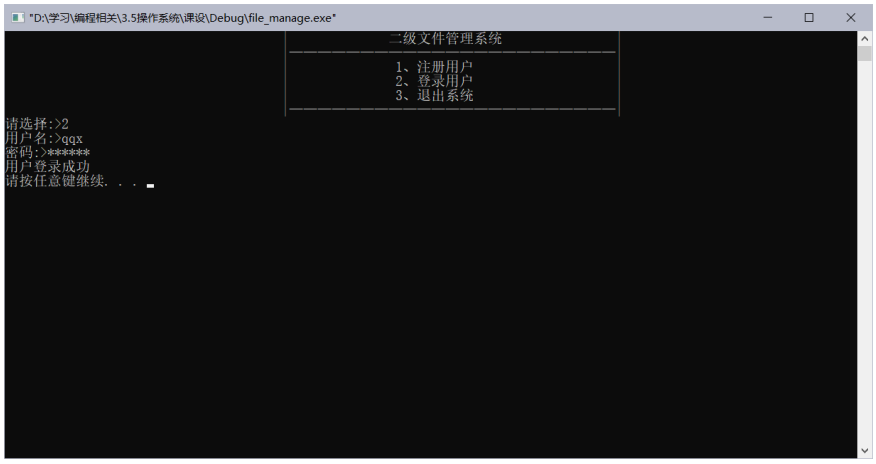
用户登陆代码段：

```
int login()
{
    char name[10], psw[10];
    char c;
    int i, times, user_psw_i;
    cout << "用户名:>";
    for (i = 0; c = getch(); i++)
    {
        if (c == 13) break;
        else
            name[i] = c;
        printf("%c", c);
    }
    name[i] = '\0';
    for (i = 0; i < used; i++)
    {
```

```
        if (!strcmp(userTable[i].userName, name))
        {
            user_psw_i = i;
            break;
        }

    }
    if (i == used)
    {
        cout << "\n 您输入的用户名不存在\n";
        system("pause");
        return -1;
    }
    for (times = 0; times < 3; times++)
    {
        memset(psw, '\0', sizeof(psw));
        cout << "\n 密码:>";
        for (i = 0; c = getch(); i++)
        {
            if (c == 13) break;
            else
                psw[i] = c;
            cout << "*";
        }
        printf("\n");
        if (!strcmp(psw, userTable[user_psw_i].password))
        {
            printf("用户登录成功\n");
            system("pause");
            break;
        }
        else
        {
            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);
        }
    }
    fflush(stdin);
    return user_psw_i;
}
```

附演示截图：



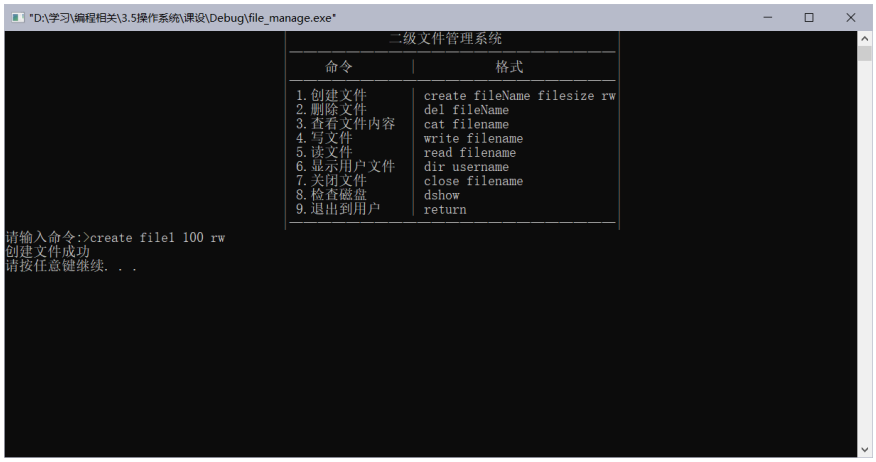
对于用户登陆输入的用户名，遍历结构体链表找到用户名属性，匹配字符串如果存在则把输入密码的字符串与该结构体实例对应密码属性进行字符串匹配，如果通过则进入文件管理系统二级界面。

创建文件代码段：

```
void createFile(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");
          system("pause");
          return;
      }
  }
  if (requestDist(startPos, length))   //磁盘分配查询
  {
      fileNode = (UFD *)malloc(sizeof(UFD));
      fileNode->file = (fileTable *)malloc(sizeof(fileTable));  //这一步必不可少，因为 fileNode 里面的指针也需要申请地址，否则 fileNode->file 指向会出错
      strcpy(fileNode->file->fileName, fileName);
      strcpy(fileNode->file->fileKind, fileKind);
      fileNode->file->maxlength = length;
      fileNode->file->strat = startPos;
      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");
            system("pause");
        }
        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");
            system("pause");
        }
    }
```

附演示截图：



设定输入命令存储方式：for(i=k+1,k=0;command[i]!=' ';i++,k++) command_str3[k]=command[i];command_str3[k]='\0';将其存放于字符串数组，其中 create 命令设置了四位，分别存放命令名、文件名、文件大小（字节为单位）、权限（r 读 w 写）。命令格式输入正确以后，在模拟磁盘（动态内存分配的 512KB 大小的模拟内存）中申请相应文件大小的空间，具体由链表的 start 属性和 length 属性标记磁盘使用情况。

读文件代码段：

```
void fileCat(char fileName[])
{
  int startPos, length;
  int 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 (int i = startPos; k < length; i++, k++)
            {
                if (i % 50 == 0) printf("\n");  //一行大于 50 个字符换行
                printf("%c", disk[i]);
            }
            printf("\n\n");
            printf("-------------------------------------\n");
            printf("%s 已被进程打开,请用 close 命令将其关闭\n",
p->file->fileName);
            system("pause");
        }
        else
        {
            printf("没有找到该文件,请检查输入的文件名是否正确\n");
            system("pause");
        }
    }
```

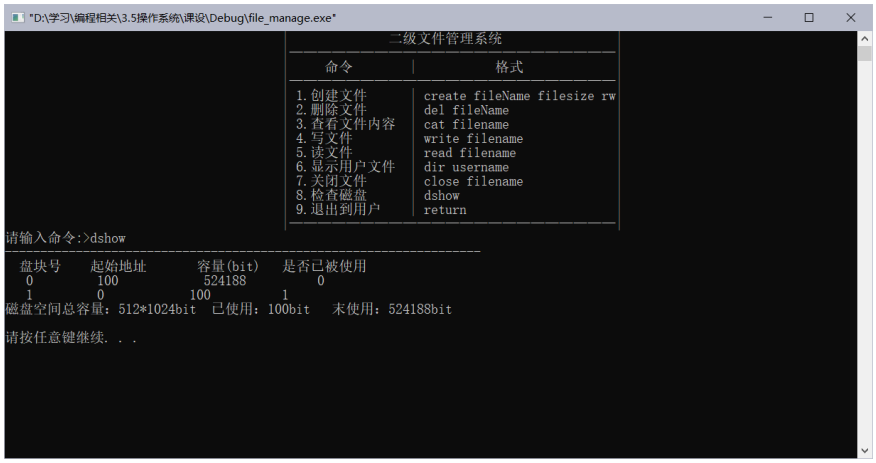
附演示截图：

cat 设置了两位，分别是命令名、文件名。 遍历结构体链表查找到与输入文件名相同的结构体实例并输出对应存储的内容，如果是非可读文件则提示权限不够无法操作。

磁盘检查代码段：

```
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      \n", i,
p->start, p->maxlength, p->useFlag);
  }
 printf("磁盘空间总容量:512*1024bit  已使用:%dbit   未使用:%dbit\n\n",
MaxDisk - unusedDisk,
      unusedDisk);
  system("pause");
}
```

附演示截图：



dshow 命令设置了两位，分别是命令名、文件名。命令格式输入正确以后，在模拟磁盘（动态内存分配的 512KB 大小的模拟内存）中查询文件已经使用的大小的空间，具体由链表的 start 属性和 length 属性标记磁盘使用情况。

五、 课程设计体会与软件评价

心得体会：

由于课设周安排了三门考试，本次课程设计时间非常紧张，在这压力之下可能做出来的课设质量没有我自己期望的那么高。选题之前我其实蛮纠结的，两个题目我倾向于第一个进程管理，但因为之前做过了相关实验，就学习新知识而言我还是选择了做第二个文件目录管理，从老师提前一周发布了指导书到课设周前，我陆续阅读了 CSDN 和博客园很多关于文件管理系统设计的博客，发现有一种模拟 Linux 命令行文件的操作的趋势，可能确实是大势所趋现在 Linux 更加普及应用范围更广了。在选数据结构的时候我最后决定用结构体链表，是因为他的扩展性比普通的数组要高得多我可以做更多级的目录而不局限于一个二维数组存储的二级目录。（用树也是可行的，甚至在层次上更加分明）。

记录遇到的一些问题：

在设计输入用户名时使用如下代码：

```
for (i = 0; c = getch(); i++)
```

此时我用的编译器是 Visual Studio2017，对于 getch（）函数出现了一些问题，getch（）与 getchar（）函数一样用于从缓冲区获取字符，但前者不是显式的，这正是我设计输入密码时用到的，由于不是显式的我可以屏幕输出*代替输入的字符以达到密码输入的保密性和软件的专业性。但是 Visual Studio 编译运行产生的的命令行黑框不支持该函数的多字符即字符串输入，仅支持单字符的缓冲区获取。这个问题起初让我怀疑函数本身设计的问题，在一篇很少人关注的 blog 上我发现一位大神有发现这个问题并解释道，编译器支持函数的方式不同，用 VC++编译执行可解决，虽然他没说的很彻底但我至少知道了不同编译器对相同代码编译产生的效果是不一定完全相同的。就是底层汇编的规则不一样。（我自己的编译原理还有待学习）；

再有收获的代码：

```
fileNode->file = (fileTable *)malloc(sizeof(fileTable));
```

此处 malloc（）函数用于动态内存分配，这句代码我也是从初次见到用了很多次才能掌握，sizeof（）返回 fileTable 的大小，这里 fileTable 是结构体数组，为 fileNode->file 结构体链表申请相当大小的内存空间，因为 fileNode 里面的指针也需要申请地址,否则 fileNode->file 指向会出错。

总的来说，本次课设虽然短暂但是还是从中得到了很多 C 编程的知识，字符串的精确匹配（如输入命令：create file1 100 rw），我用空格再次细分每个字符串，’\0’和’ ’。此外，对二级文件目录的理解更深刻，理

解了以前我们在 Linux 命令行环境下进行的文件操作，不再仅只是书上的片面知识。

软甲评价：

软件设计存在一些不足：

未设计全面：文件权限更改，文件重命名等功能；

过于片面：我理解老师的模拟磁盘偏了，以为仍然是将目录文件存在模拟磁盘里，但现在想起来如果存在现实磁盘里更能实现我设计这个软件系统的目的。希望这一点小小的不足对课设影响不大吧，下次一定尽力。

最后感谢老师的理解与支持，之前与老师的沟通让我做这个系统有了自己的方向和思路。