

# 操作系统之 文件管理

张仕奇 1152671

语言：C

运行环境：vs2008/2010/2012

## 一. 实验目的

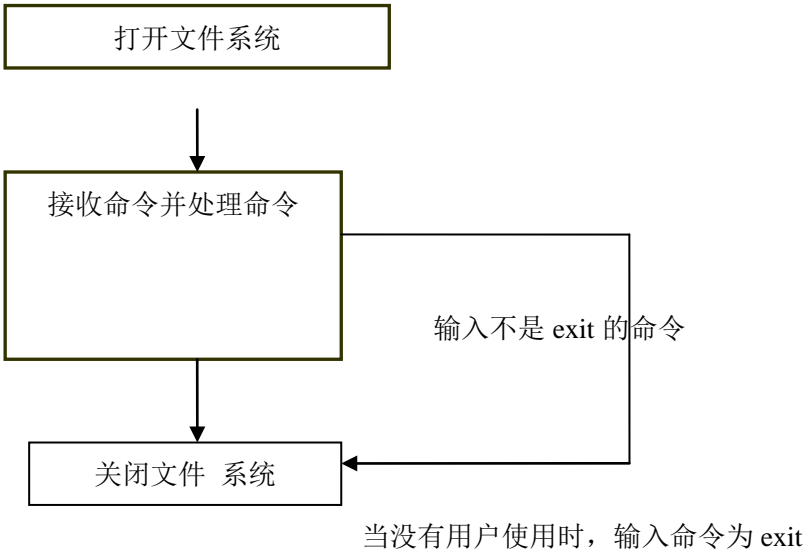
熟悉文件存储空间的管理；  
熟悉文件的物理结构、目录结构和文件操作；  
熟悉文件系统管理实现；  
加深对文件系统内部功能和实现过程的理解

## 二. 功能介绍

以 1M 的存储器空间作为文件空间，空间“分块”。超级块，在最前面，共占3.2k个字节。其中0.2K个字节存放目录节点的位示图，1K个字节存放文件节点的位示图，2k字节存放盘块节点的位示图。用位状态='0'表示空闲，状态='1'表示已分配。后半部用于存放目录接点和文件接点。超级块不参与文件空间的动态分配。其他块用于存贮目录接点和文件接点的信息。该模拟文件系统包括目录文件（简称目录）、普通文件（简称文件），并实现下面一些基本功能：

1. 改变目录：CD 〈目录名〉，工作目录转到指定的目录下。
2. 创建文件：CREATE 〈文件名〉，创建一个指定名字的新文件，即在目录中增加一项，不考虑文件的内容。
3. 删除文件：DELF 〈文件名〉，删除指定的文件。
4. 显示目录：DIR [〈目录名〉]，显示目录下全部文件和第一级子目录，如果没有指定路径名，则显示当前目录下的内容。
5. 创建目录：MD 〈目录名〉，再指定路径下创建指定的目录，或者在没有指定路径时，在当前目录下创建子目录。
6. 删除目录：DELD 〈目录名〉，删除指定的目录。
7. 保存文件 SAVEF 〈目录名〉 保存指定文件
8. 保存目录 SAVED 〈文件名〉，保存指定目录
9. 获得命令帮助 HELP
10. 退出文件系统 EXIT

## 2. 2 功能及模块设计



## 三、主要数据结构及函数设计

### 3. 1 数据结构设计

<1>目录文件结点信息存储结构:

```
struct dir_node                                     //目录节点
{
    SYSTEMTIME ctime;                               //创建时间
    char dir_name[32];                               //目录名
    int child_dir[8];                                //子目录索引
    int dir_count;                                   //当前子目录数
    int child_file[16];                              //子文件索引
    int file_count;                                  //当前子文件数
    int parent;                                       //父目录索引
}
```

<2>文件目录存储位置存储结构:

```
struct file_node                                   //文件节点
{
    SYSTEMTIME ctime;                               //创建时间
    char file_name[32];                             //文件名
    int block[4];                                    //该文件占有的磁盘块索引
    int block_count;                                //该文件当前占有的磁盘块数
    int file_length;                                //文件长度
    int parent;                                       //父目录索引
};
```

### <3>文件、目录、盘块的占用的标志位

```
int dir_flag[DIR_NUM];           //各目录节点占用标志, 0 表示空闲, 1 表示被占用
int file_flag[FILE_NUM];         //各文件节点的占用标志
int block_flag[BLOCK_NUM];       //磁盘块的占用标志
```

### <4>常驻内存的标志文件（目录）修改的标志位和标志文件（目录）修改标志位的标志位

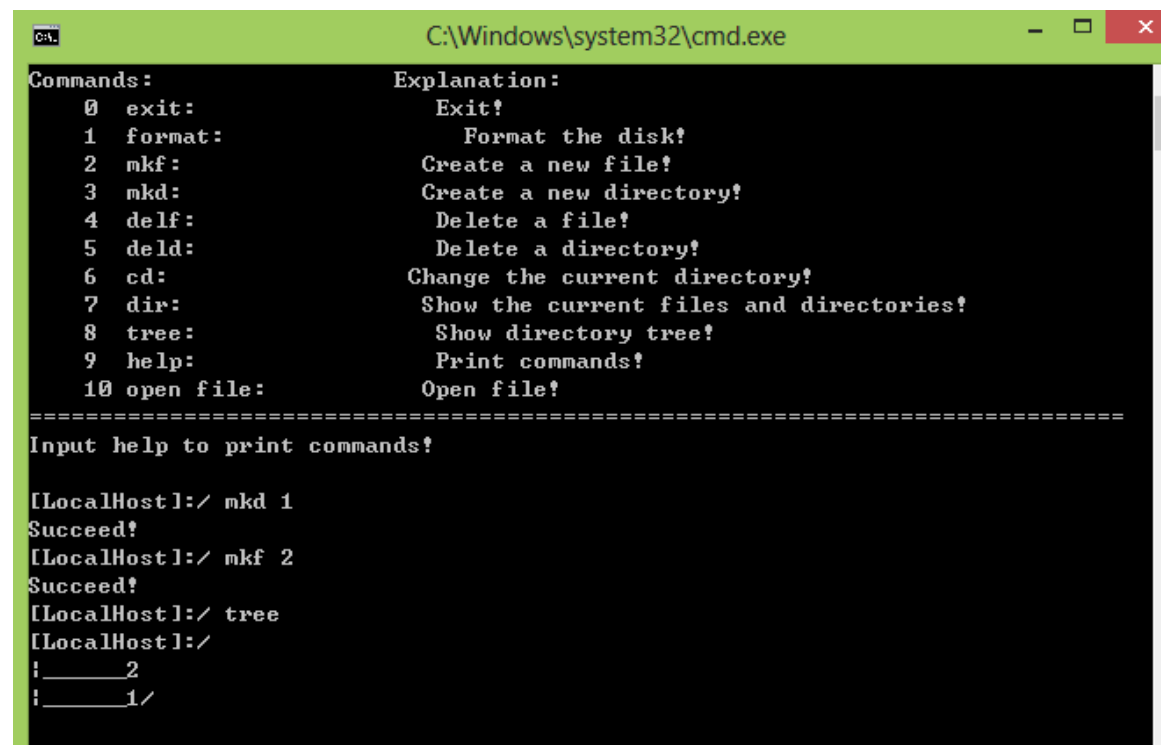
```
int dir_change_flag[DIR_NUM];    //用于标志目录节点的修改
int file_change_flag[FILE_NUM];  //用于标志文件节点的修改

int dirflag_flag[DIR_NUM];       //用于标志 dir_flag[DIR_NUM]的修改
int fileflag_flag[FILE_NUM];     //用于标志 file_flag[FILE_NUM]的修改
int blockflag_flag[BLOCK_NUM];   //用于标志 block_flag[BLOCK_NUM]的修改
```

### <5>记录当前路径和当前目录的全局变量

```
int curr;                        //当前目录索引
dir_node *curr_dir;              //当前目录节点指针
char curr_path[512];             //当前路径
```

## 四、测试数据及运行结果



```
C:\Windows\system32\cmd.exe

Commands:           Explanation:
0  exit:            Exit!
1  format:          Format the disk!
2  mkf:             Create a new file!
3  mkd:             Create a new directory!
4  delf:            Delete a file!
5  deld:            Delete a directory!
6  cd:             Change the current directory!
7  dir:            Show the current files and directories!
8  tree:           Show directory tree!
9  help:           Print commands!
10 open file:      Open file!

=====
Input help to print commands!

[LocalHost]:/ mkd 1
Succeed!
[LocalHost]:/ mkf 2
Succeed!
[LocalHost]:/ tree
[LocalHost]:/
!_____2
!_____1/
```

## 五、实验总结

本系统基本上实现文件系统的部分命令。如 cd、dir、create、md、delf、deld 等。而且通过采用树型数据结构，可以最多显示 64 级目录，建立 512 个文件节点。但仍存在很大不足：

- (1)没有按照面向对象思想设计程序结构，仍以函数为主，而不是对象为主；
- (2)使用太多全局变量，浪费太多有用空间，而且存在安全隐患；
- (3)系统函数多，并且没有完全按照匈牙利命名法命名函数名和变量名，程序的可读性比较差，使得整个程序显得清晰度不够。
- (4)文件系统的测试用例还不够多，大量测试时还存在几个 bug,特别是删除的文件和目录时有时候难以将控制信息和标志信息写入硬盘。

在开发系统的过程中，我对操作系统的功能实现，特别是文件管理这一块的内容有了较深的理解，这个理解程度是看书所无法达到的。另外，这次开发要使用到文件存取的操作来实现

数据块资料的存取，从而使我对文件存取的操作更熟悉了。

当然在写 c 时我也参考了其他人的一些代码和想法