## 实验五报告：简单文件系统的实现

### 一、实验目的

通过具体的文件存储空间的管理、文件物理结构、目录结构和文件操作的实现，加深对文件系统内部数据结构、功能以及实现过程的理解。

### 二、实验内容

* 在内存中开辟一个虚拟磁盘空间作为文件存储器，在其上实现一个多用户多目录的文件系统。在退出该文件系统的同时，应将虚拟磁盘上的内容以一个文件的方式保存到磁盘上，以便下次可以再将它恢复到内存的虚拟磁盘中。
* 文件物理结构可采用显式链接或其他方法。
* 空闲磁盘空间的管理可选择FAT图、位示图或其他方法。
* 文件目录结构采用多级目录结构。为简单起见，目录组织方式可以不使用索引结点，每个目录项应包含文件名、物理地址、长度等信息，还可以通过目录项实现对文件的读写保护。
* 要求提供以下操作命令：
  + my\_format：对文件存储器进行格式化，即按照文件系统的结构对虚拟磁盘空间进行布局，并在其上创建根目录以及用于管理文件存储空间等的数据结构。
  + my\_mkdir：用于创建子目录
  + my\_rmdir：用于删除子目录
  + my\_ls：用于显示目录中的内容
  + my\_cd：用于更改当前目录
  + my\_create：用于创建文件
  + my\_open：用于打开文件
  + my\_close：用于关闭文件
  + my\_write：用于写文件
  + my\_read：用于读文件
  + my\_rm：用于删除文件
  + my\_exitsys：用于退出文件系统

### 三、设计方案

* 在该虚拟文件系统启动时，申请一块内存作为磁盘空间
* 将这块内存空间格式化，仿照FAT16文件系统

|  |  |  |  |
| --- | --- | --- | --- |
| * 1块 | * 2块 | * 2块 | * 995块 |
| * 引导块 | * FAT1 | * FAT2 | * 数据区 |

* **关键数据结构设计**
  + 引导块sysblock
  + 在引导块中主要存放逻辑磁盘的相关描述信息，磁盘块大小、数量、文件分配表、根目录区、数据区在磁盘中的起始位置等。本实例是在内存的虚拟磁盘中创建的一个文件系统，因此包含的内容较少，只有磁盘块大小，数量，数据区开始位置，根目录开始位置等
  + typedef struct BLOCK {  
     char magic\_number[8]; //魔数  
     char information[200]; //文件系统信息  
     unsigned short root; //根目录起始盘块号  
     unsigned char \*startblock; //虚拟磁盘数据区开始位置  
    } sysblock;
  + 文件控制块fcb
  + typedef struct FCB {  
     char filename[8];  
     char exname[3];  
     unsigned char attribute; //0目录文件，1数据文件  
     unsigned short time; //文件创建时间  
     unsigned short date; //文件创建日期  
     unsigned short first; //文件起始盘块号  
     unsigned long length; //文件大小  
     char free; //0：空fcb  
    } fcb;
  + 文件分配表fat
  + typedef struct FAT {  
     unsigned short id;  
    } fat;
  + 用户打开文件表useropen
  + 当打开一个文件时，必须将文件的目录项中的所有内容全部复制到内存中，同时还要记录文件操作的动态信息，如读写指针的值等。本实例中实现的是一个用于单用户单人舞系统的文件系统，为简单起见，我们把用户文件描述符表和内存fcb表合在一起，称为用户打开文件表，表项数目为10，即一个用户最多可同时打开10个文件，用一个数组来描述，即数组下表即某个打开文件的描述符
  + typedef struct USEROPEN {  
     char filename[8];  
     char exname[3];  
     unsigned char attribute; //0目录文件，1数据文件  
     unsigned short time; //文件创建时间  
     unsigned short date; //文件创建日期  
     unsigned short first; //文件起始盘块号  
     unsigned long length; //文件大小  
     char free; //0：空fcb  
      
     int dirno; //在父目录文件中的盘块号  
     int diroff; //fcb在父目录中的逻辑序号  
     char dir[80]; //全路径信息  
     int pos; //读写指针在文件中的位置  
     char fcbstate; //是否修改  
     char topenfile; //打开的表项是否为空  
    } useropen;
  + 全局变量定义
  + unsigned char \*myhard; //指向虚拟磁盘内存的起始地址  
    unsigned char buffer[SIZE]; //缓冲区  
    useropen filelist[MAXOPENFILE]; //用户打开文件表数组  
    int currfd; //用户当前目录位置  
    unsigned char \*startb; //记录虚拟磁盘数据区开始位置
  + 虚拟磁盘空间布局
  + 由于真正的磁盘操作需要设计设备的驱动程序，所以本实例是在内存中申请一块空间作为虚拟磁盘使用，我们的文件系统建立在这个虚拟磁盘上。虚拟磁盘一共划分为1000个磁盘块，每个块占1024字节，其布局格式是模仿FAT文件系统设计的

### 四、实验过程

* **initSys**
  + 功能：文件系统启动函数，读取文件系统内容，初始化全局变量并打开根目录
  + 流程：
    - 如果已经存在文件系统（即SYSNAME文件且开头魔数相同）则将root目录载入，否则调用my\_format函数创建文件系统再载入
    - 初始化根目录并进入
    - 初始化文件打开表filelist，将当前目录写入第一个
* **my\_format**
  + 功能：格式化磁盘，初始化前6个磁盘块
  + 流程：
    - 分配虚拟磁盘内存myhard
    - 初始化引导块，即第0块
    - 初始化两个FAT表，即第1-4块，每个FAT表占2个块
    - 初始化根目录，即第5块，创建.和..两个特殊目录
    - 写入磁盘空间，即SYSNAME文件
* **my\_exitsys**
  + 功能：将文件系统数据写入磁盘保存，退出系统
  + 流程：
    - 依次关闭已打开文件
    - 将虚拟磁盘信息myhard写入SYSNAME文件
* **my\_mkdir(char \*filename)**
  + 功能：在当前目录创建新目录
  + 流程：
    - 调用do\_read读入当前目录文件到内存，检查新建文件目录是否重名
    - 分配一个空间的打开文件表项
    - 分配一个空闲盘块
    - 更新FAT表
    - 在当前目录中为新建目录寻找一个空闲的目录项
    - 设置fcb
    - 创建特殊的两个目录项.和..
* **my\_cd(char \*filename)**
  + 功能：在当前目录进入filename目录
  + 流程
    - 调用do\_read读当前目录文件表项到内存
    - 遍历查找filename是否存在
    - 如果是.则直接返回；如果是..则直接关闭当前目录
    - 如果是其他目录，申请一个打开文件表项，把找到的目录信息写入打开文件表，设置当前目录为该目录
* **my\_rmdir(char \*filename)**
  + 功能：删除空目录
  + 流程：
    - 调用do\_read读当前目录文件表项到内存
    - 遍历查找要删除的文件目录是否存在
    - 检查该目录是否为空
    - 检查是否已经打开，若打开则调用close关闭
    - 回收目录文件的磁盘块
    - 清空该目录文件的目录项
    - 修改用户打开表项的长度信息
* **my\_ls**
  + 功能：列出当前目录中的所有目录和文件项
  + 流程：
    - 调用do\_read读出当前目录文件表项到内存
    - 将目录文件信息打印到屏幕上
* **my\_create(char \*filename)**
  + 功能：新建文件
  + 流程：
    - 检查文件名是否合法
    - 检查是否在目录中新建文件，不能再文件中新建文件
    - 读当前目录文件表项到内存
    - 遍历检查是否有重名文件
    - 申请fcb和空闲盘块，更新FAT表
    - 更新当前目录的打开目录文件表项
    - 修改父目录的fcb
* **my\_rm(char \*filename)**
  + 功能：删除文件
  + 流程：
    - 调用do\_read读当前目录文件表项到内存
    - 遍历查找filename文件
    - 更新FAT表
    - 清空该文件的fcb
    - 修改父目录文件的fcb
* **my\_open(char \*filename)**
  + 功能：打开文件
  + 流程：
    - 调用do\_read读当前目录文件表项到内存
    - 遍历查找filename文件
    - 申请空闲的打开文件表表项
    - 写入打开文件表
* **my\_close(int fd)**
  + 功能：关闭文件
  + 流程：
    - 检查fd的有效性
    - 检查用户打开表表项的fcbstate字段，即是否修改
    - 回收该文件占据的打开文件表表项内存
* **my\_write(int fd)**
  + 功能：在文件中写入数据
  + 流程：
    - 检查fd的有效性
    - 提示用户输入写模式
    - 提示用户输入内容
    - 调用do\_write()将键入的内容写入到文件中
    - 返回写入的字节数
* **my\_read(fd)**
  + 功能：读取文件内容并输出
  + 流程：
    - 检查fd的有效性
    - 调用do\_read()读取内容到内存
    - 输出
* **do\_write(int fd, char \*text, int len, char wstyle)**
  + 功能：写入
  + 参数：
    - fd：打开文件表的下标
    - text：写入的内容
    - len：写入长度
    - wstyle：写入的模式
  + 流程：
    - 根据不同的写入模式移动写指针，截断写移到开头，追加写移到末尾
    - 定位磁盘块和块内偏移量
    - 循环写入
    - 如果写入长度比磁盘块大小大，则写满后申请一块新的磁盘块，循环写入；否则一次性写入
    - 删除多余的磁盘块，因为截断写可能导致新写入的数据占据磁盘块数量小于原占据的磁盘块数量，需要释放
* **do\_read(int fd, int len, char \*text)**
  + 功能：读出
  + 参数：
    - fd：打开文件表的下标
    - text：写入的内容
    - len：写入长度
  + 流程：
    - 根据读取指针位置pos，首磁盘块first，fat表定位读取目标磁盘块和块内地址
    - 循环读取到剩余待读取长度小于等于0
    - 每次读取判断剩余要读取的长度是否超过当前磁盘块，不超出就读完，超出就读到这个磁盘块结束，等下一个循环再读

### 实验遇到的问题及解决方式

1.追加写时，写入的内容没有保存到文件里

找此文件跨越的最后一个磁盘块时出错。文件开始指针应为数据区开始地址+文件打开表当前表项的 first\*BLOCKSIZE。

2.删除文件时问题

把当前目录第一个盘块号赋值给 fat 表的 id,通过 fat 表依次查找文件的下一个盘块，把 fat 的 id 设置为 free。若只将 fat 表中文件的表项释放，并没有将实际文件中的内容删除，防止数据被恢复，保障用户的数据安全。

根据当前目录的第一个盘块号的 id,磁盘块大小 BLOCKSIZE,得到id\*BLOCKSIZE 为文件的第一个磁盘块地址，通过循环找到文件的每个磁盘块并把所占的磁盘空间清空。

程序实现创新性代码量少，代码可读性高。可以打印fat表直观

一些程序目前存在的缺陷：

(1) 删除目录文件时，若欲删除目录文件为空，正常删除该目录文件；不为空则直接返回，并显示错误信息。可以改进为若不为空，提示用户是否将目录内文件同时删除，若用户同意，通过 FCB 表找到该目录下的子文件，先将子文件删除，再删除整个父目录文件。

(2) 文件系统内文件的名称不支持中文以及其他 unicode 码，不支持 emoji 表情作为文件名。

(3) 程序崩溃时数据无法保存。

3.程序考虑的改进思路

(1) 增加函数模块，增加调用命令。

(2) 增加多种编码格式。

(3) 增加一个账号登录，提高安全保障。