模拟文件系统

1、题目概述

用一个大文件模拟磁盘，实现一个简化的 ext 文件系统，及简化的类 Linux 命令行界面。

要求两人一组协作完成，每组提交一份程序源代码文件及说明文档。说明文档中应包含功能描述（特别是规定以外的功能）及小组内的分工说明，同小组的两名同学可能由于分工不同得到不同的分数。

2、题目要求

程序首次运行时（即磁盘文件不存在时），应进行“格式化”操作；程序再次运行时（即磁盘文件已存在时），应读取文件系统状态。

进行必要初始化后，应设置当前路径为根目录，并显示命令行界面“>>”，等待用户输入命令。根据用户输入的命令实现文件的创建、读取、删除；目录创建、切换、查看、删除等基本操作。

程序退出时应将文件系统状态写入模拟磁盘文件中。

至少支持以下命令：

命令格式	功能说明
pwd	查看当前目录
cd $path	将当前目录切换到$path 目录
mkdir $path	创建$path 目录
ls $path	查看$path 目录或文件
rmdir $path	删除$path 目录，递归操作，先删除目录树的叶子节点
echo $str $pat	将$str 写入$path 文件。如果文件已存在则覆盖原文件,，否则创建文件。
cat $path	读取$path 文件内容
rm $path	删除$path 文件

其中，$path 表示文件路径名，可以是绝对路径（以“/”开头），也可以是相对路径。规定用“/”符号分隔目录名称。功能说明中，创建文件或目录时，规定上级目录必须已经存在。功能说明中要求文件或目录存在，但实际上不存在

的，给出错误提示“$path No such file or directory”。功能说明中要求$path为目录，但实际上为文件的，给出错误提示“$path is not a directory.”；功能说明中要求$path为文件，但实际上为目录的，给出错误提示“$path is not a file.”

3、模拟磁盘文件格式

简化起见，在本次作业中，使用的模拟磁盘文件大小固定为16520KB。该文件以4KB为单位,可分成4130个块。最开始的2个块称为超级块(super block)，之后的32个块存储索引结点（inode block)，再之后的4096个块存储文件数据（data block)，如下图所示。

区域	超级块		索引结点			数据块		
功能	inode bitmap	block bitmap	inode_0	……	inode_n	block_0	……	block_n
大小	8KB		32B * 4096 = 128KB			4KB * 4096 = 16MB		

3.1 超级块区

超级块区中存储两个位图（bitmap)，分别用来表示索引结点和数据块的使用情况。索引结点位图中的每个bit表示对应的索引结点是否占用，数据块位图中的每个bit表示对应的数据块是否占用。这样,恰好每个位图占用1个4KB块。

3.2 索引结点区

索引结点区中连续存放着4096个索引结点，每个索引结点占用32字节，共计128KB。

每个索引结点中存储着一个文件或目录的描述信息，包括：索引结点编号、该结点对应的是文件还是目录、内容长度、数据块编号等。

0号索引结点中存储的是根目录的信息，应在格式化时设置。

简化起见，规定每个文件的内容均不会超过4KB。

3.3 数据块区

数据块区中连续存放着4096个数据块，每个数据块4KB，共计16MB。

文件对应的数据块中存储的是正常的文件内容。

目录对应的数据块中存储的是目录项（directory entry)。这里规定每个目

录项占用 256 字节，这样一个数据块中最多可以存储 16 个目录项，即每个目录下最多可以有 16 个文件或子目录。

每个目录下均有一个名称为“.”的目录，即为本身；除根目录外，每个目录下均有一个名称为“..”的目录，即为上级目录。

3.4 目录项

目录项中存储着项目的名称和对应的索引结点编号。

简化起见，规定项目名称最长 252 字节，如果不足 252 字节，通过补“\0”的方式凑足 252 字节。规定项目名称由字母（区分大小写）、数字、小数点“.”和下划线“_”组成，不得包含空格和其他符号，也不得包含中文字符。项目名称全为“\0”的目录项是未使用的目录项（可能由于删除文件导致）。

索引结点编号为 4 字节整数，正常的取值范围在[0-4095]。

3.5 数据结构

前述过程中涉及的主要数据结构定义如下：

```
// super_block区域
// 使用位图来记录inode和block的情况
// 如果inode_x已经被使用，则inode_bitmap[x] = 1
// super_block共占用8k
struct super_block {
    bool inode_bitmap[4096];   // inode_bitmap占用1个block
    bool block_bitmap[4096];   // block_bitmap占用1个block
};

// inode区域
// 每个inode记录一个文件的描述信息
// 每个inode大小为32 Byte，因此1个block有128个inode
// 共4096个inode，共4096*32B/1024=128KB用于存放inode
struct inode {
    int i_id;                 // inode号码
    int i_mode;               // 该inode是文件还是目录
    int i_file_size;          // 文件大小,单位是字节（Byte）
    ...                       // 其他字段，如有必要可以自由添加
    int i_blocks[1];          // 该文件占用blocks,每个文件最多25KB
    char i_place_holder[x]; //占位符，补全32 Byte，x=32-4*4=16
};

```

```
// data区域
// 每个data_block记录文件的具体内容
// data_block分为file_block和dir_block
// 每个file_block大小为4kB，因此1个block有1个file_block
// 共4096个，共4096*4K=16M
struct file_block {
    char data[4096];     //每个data_block的内容是4KB
};

// dir_entry是目录项
// 每个dir_entry大小为256B，因此16个block有4个目录项
struct dir_entry {
    char name[252];      //目录名,所以目录和文件名长度有限制为252B
    int inode_id;        //对应的inode
};

// 每个data_block中有16个目录项，所以每个目录下面最多有16个子目录或者文件
struct dir_block {
    dir_entry dirs[16]; //256*16=4096
};
```

4、操作举例

例 1：>> mkdir /home

1、由于“/home”是绝对路径。因此，首先读取根目录的索引结点（即 0 号索引结点）。

2、根据 0 号索引结点里面的数据 i_mode，发现这是目录。

3、读取 0 号索引结点里面 i_blocks 的数据，找到目录文件所在的数据块号。

4、读取该数据块中的数据，在 dir_block 结构的 dirs 数组中查找是否存在名字为“home”的目录项。

5、若已存在名为“home”的目录项，提示“/home already exists.”。回到命令行提示符“>>”，等待下一条命令。

6、若不存在，找一个未使用的目录项、未使用的索引结点和未使用的数据块。这里：

6.1、未使用的目录项可通过查找名称为空的目录项获得；

6.2、未使用的索引结点可通过超级块中的索引结点位图，找到其中为 0 的位，获得相应的索引结点；

6.3、未使用的数据块可通过超级块中的数据块位图，找到其中为 0 的位，获得相应的数据块。

7、修改未使用的目录项，名称为“home”，索引结点号为新分配的索引结点的编号；修改索引结点位图，将对应位改为 1；修改索引结点，记录新分配的数据块编号；修改数据块位图，将对应项改为 1；修改数据块内容，加入“.”和“..”目录的信息。

8、（可选）将上述修改写入模拟磁盘文件。

例 2：>> echo "abc" ../a.txt

1、“../a.txt”为相对路径。因此，从当前目录的索引结点中找到对应的数据块。

2、读取该数据块中的数据，在 dir_block 结构的 dirs 数组中查找名字为“..”的目录项。

3、从该目录项中得到对应的索引结点编号。

4、从该索引结点中确认这是一个目录。

5、从该索引结点中找到对应的数据块编号。

6、读取对应的数据块中的数据，在 dir_block 结构的 dirs 数组中查找名字为“a.txt”的目录项。

6.1、若 a.txt 已存在，找到对应的索引结点，并确认 a.txt 是文件还是目录。

6.1.1、若 a.txt 是目录，提示错误；

6.1.2、若 a.txt 是文件，找到对应的数据块，覆盖原内容，写入双引号中的文件内容“abc”，同时修改索引结点中记录的文件长度。

6.2、若 a.txt 不存在，创建 a.txt 文件，写入文件内容“abc”，同时修改索引结点中记录的文件长度。

6.2.1、创建 a.txt 文件的过程，应包括：找到未使用的目录项、找到未使用的索引结点、找到未使用的数据块，并参考例 1 创建目录过程完成配套操作。

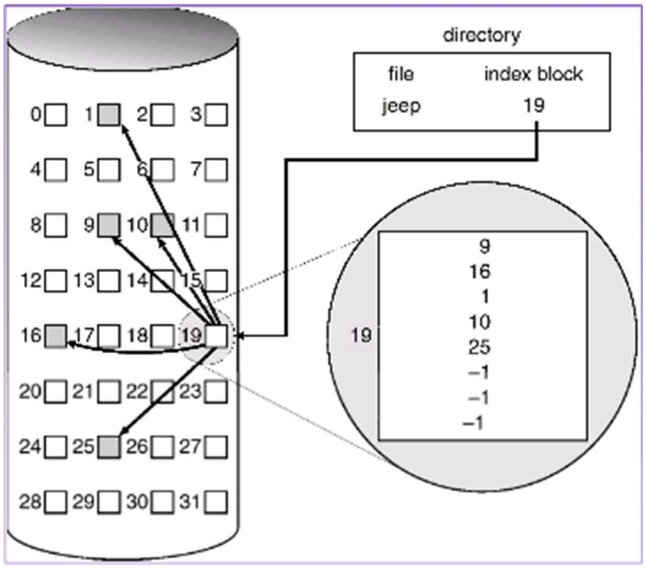
6.2.2、（可选）如果文件内容为空，即如“echo "" ../a.txt”这样的命令，可以不占用数据块。

5、拓展阅读

在操作系统中，文件是指具有文件名的一组相关元素的有序序列，是一段程序或数据的集合，而文件系统是操作系统的一部分，主要功能是管理存储在磁盘等物理介质中的文件的存储、检索、更新，提供安全可靠的共享和保护手段，并且方便用户使用。

文件如何存储在磁盘上呢？磁盘可以看成一个个磁盘块（扇区）组成，每个磁盘块的大小固定（512B-4K），因此存储文件的一个关键就是记录文件在哪些磁盘块上。通常有三种方法来分配和记录文件的磁盘块，连续分配、链表分配和索引分配。

Linux 文件系统是采用索引分配的，简单的索引分配如下图所示：



从图中可以看出，文件有索引节点描述，可以读 index block 中的数据，可以得到 jeep 文件所在的块分别是：9，16，1，10，25，将他们连起来就是 jeep 文件的全部内容。

典型的 linux 文件系统中磁盘主要包括四个区域：启动区、超级块（superblock）区、索引节点（inode）区、和数据（data block）区。

超级块区：使用位图来管理空闲的 inode 和 data block。

索引节点区：每个文件和目录都有一个 inode 数据结构，包括文件系统中文件的基本属性：文件大小、inode 号、存放的 block 数目和具体 block 编号等相关信息。

数据区：存放文件的内容，若文件过大，会占用多个 block。

就像一本书有封面、目录和正文一样。在文件系统中，超级块就相当于封面，从封面可以得知这本书的基本信息；inode 块相当于目录，从目录可以得知各章节内容的位置；而数据块则相当于书的正文，记录着具体内容。