## 一、题目介绍

1. 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。

1. 内容要求

(1)在内存中开辟一个虚拟磁盘空间作为文件存储分区，在其上实现一个简单的基于多级目录的单用户单任务系统中的文件系统。在退出该文件系统的使用时，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回复到内存的虚拟磁盘中。

(2)文件物理结构可采用显式链接或其他结构。

(3)空闲磁盘空间的管理可选择FAT表、位示图或其它办法。

(4)文件目录结构采用多级目录结构。为简单起见，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。

(5)要求提供以下操作命令：

1. 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。
2. my\_mkdir：用于创建子目录。
3. my\_rmdir：用于删除子目录。
4. my\_ls：用于显示目录中的内容。
5. my\_cd：用于更改当前目录。
6. my\_creat：用于创建文件。
7. my\_open：用于打开文件。
8. my\_close：用于关闭文件。
9. my\_write：用于写文件。
10. my\_read：用于读文件。
11. my\_rm：用于删除文件。
12. my\_exitsys：用于退出文件系统。

## 二、具体实验

1. 实验思路
2. 整个文件系统结构的设计

对于系统的存储逻辑结构，我们采用了无结构式存储，所有的数据都以char字符流来保存，根据针对不同的使用情况再解析成不同的数据结构。

再者对于系统的物理结构，我们采用单级索引和直接块的混合，因为如果数据小于一块，索引块会造成浪费一块磁盘块的情况，因此我们对数据小于一块的情况和数据大于一块的情况做了直接块和单级索引的转换。单级索引相比于链接形式，能支持随机读写，查询速度更快。

对于FCB的结构，我们采用了索引节点也就是i节点法，把文件名和拓展名和i节点号作为fcb，其余信息都存放于i节点中，减少了fcb的大小，进一步的提高检索的速度。

对于目录结构，我们使用了多级目录，目的是为了提高检索的效率。

再对于磁盘块和i节点表的管理，我们都使用了位示图，占用的空间极少。

对于磁盘块，我们一共分了1000块每块大小为1KB。

首块为引导块，存放根目录块的块号和位示图，目的是初始化整个磁盘和记录i节点和磁盘块的使用情况。

第2块到11块为i节点表，所有的i节点放到此处统一管理，并用一张位示图来统计其使用情况。

第12块为根目录的数据块，存放根目录信息。

下图为磁盘块分配使用图。

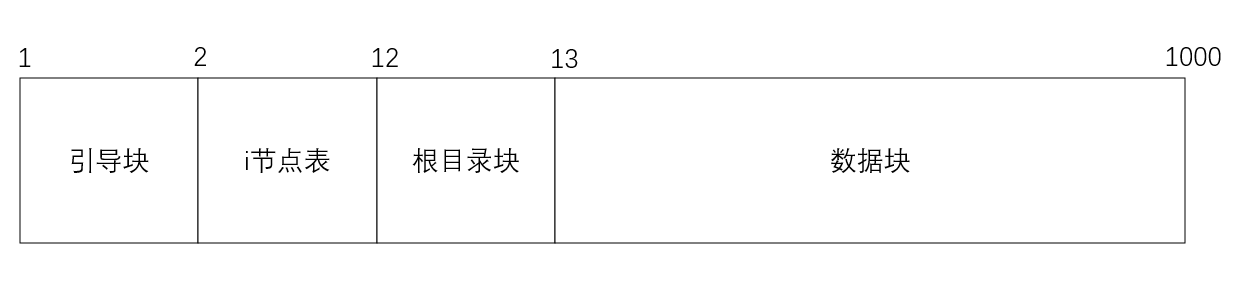


图1磁盘块使用图

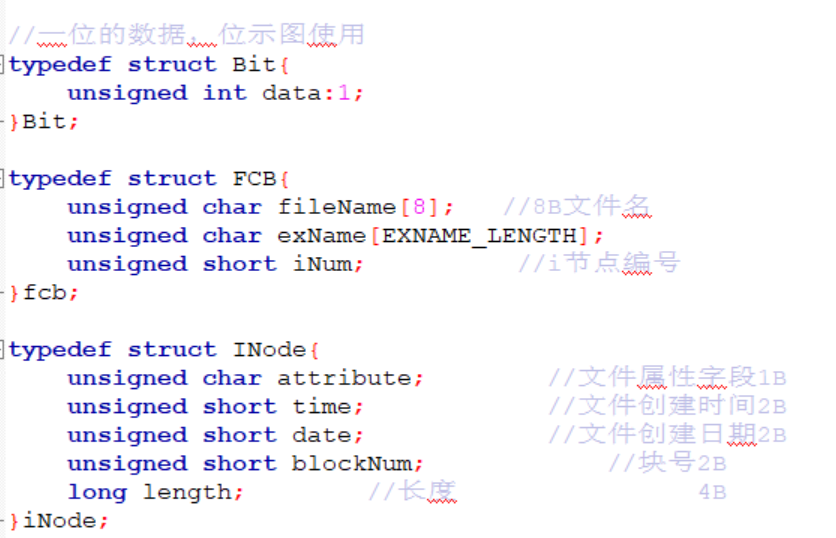


图2 Fcb和i节点信息

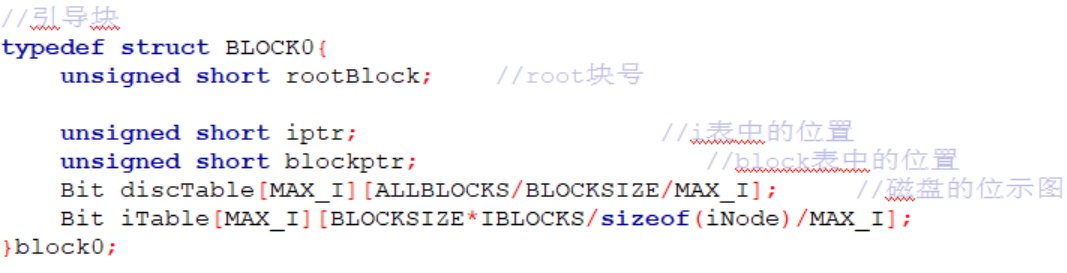


图3引导块信息

1. 各项功能的具体实现

1.start()

整个系统的启动函数，需要把整个文件的信息加载到虚拟磁盘中，并且初始化打开引导块、i节点表、根目录，把他们的内容放到内存中，方便之后的使用。

2.format()

格式化整个文件系统，首先需要初始化引导块、i节点表、根目录，对于引导块，我们需要初始化里面存放的位示图，根目录块号和位示图指针，对于i节点表，我们需要初始化其中根目录的i节点，对于根目录我们需要添加.和..的fcb信息。

3.ls()

此函数作用为展示当前目录下所有文件的部分fcb信息。

为了实现该功能，我们首先需要找到这个目录的数据块，再把数据块中的信息给一一遍历出来展示，为了提高这一查找速度，我们把目录文件的数据放到了内存的打开文件信息中的contents字段上，在内存中直接进行查找，提高了检索效率，但是这样做有一个缺点便是对于目录文件的数据块不能进行扩充，因此一个目录下面只能存放一定数量的文件。

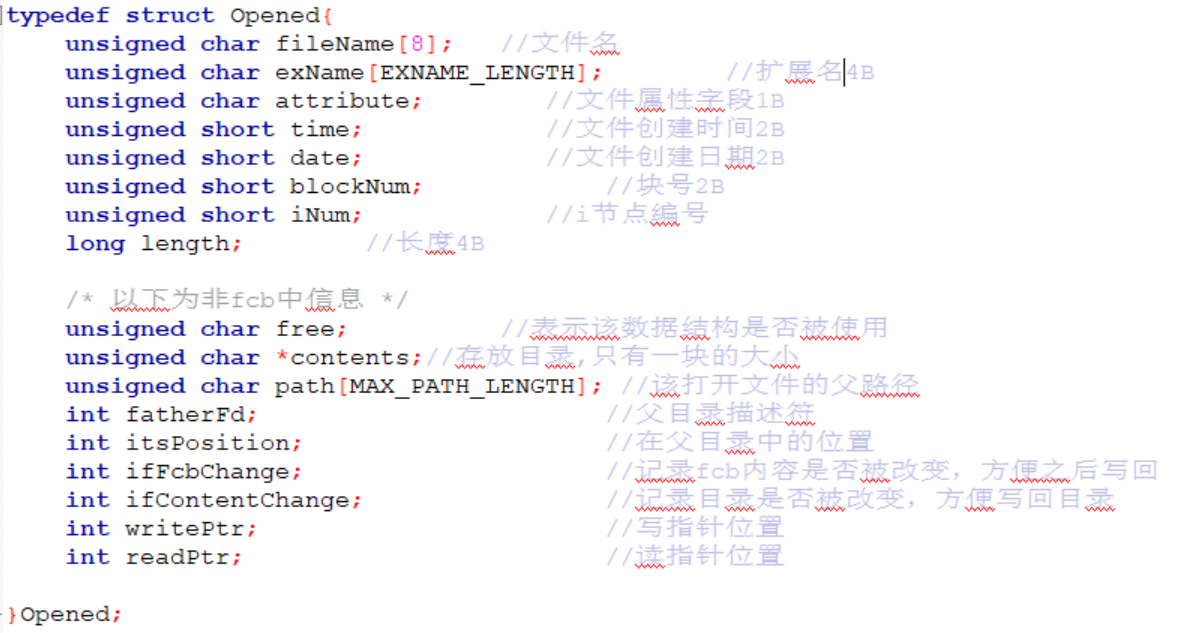


图4 打开文件信息

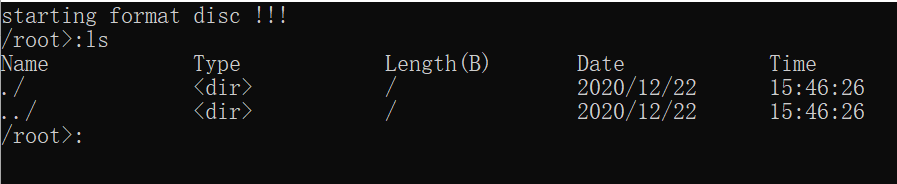


图5 ls操作

3.open(unsigned char \*path,int tempFd)与cd()

对于打开文件操作，实际目的便是把文件的fcb信息放入到内存当中。

对于我设计的文件系统，我们把打开的文件存放在内存的一个openFileList中，其数据结构如图4所示。

Open操作需要打开指定路径的文件，首先我们需要从指定的文件描述符找到相应的目录文件，再从其contents数据结构中获取到当前目录的数据，然后再遍历当前的目录通过文件名来查询我们需要打开的文件的fcb，当查找到相应的fcb之后，把响应的文件信息添加到openFileList中，便完成了打开操作。

对于open操作，我也实现了打开绝对路径的操作，具体实现是通过递归操作依次打开当前目录下的文件，直到打开到最后一层或者是找不到响应的文件。

对于cd操作，实际上就是open操作，只是cd操作打开的是目录文件，open操作打开的是普通文件，但是其实现的方法是一致的。

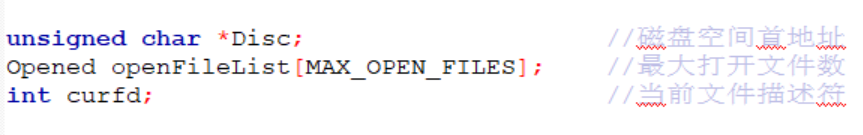


图6 内存中信息



图7 打开文件



图8 cd操作

4.create(unsigned char \*name)与mkdir(unsigned char \*name)

create操作和mkdir操作分别为创建文件和创建目录，实际上也是相同的操作。

因为fcb是文件存在的唯一标识，因此创建文件可以说来就是创建文件的fcb。

为了创建文件的fcb，我们首先需要在当前目录中找到一块空位置，该fcb位置未被使用的标志是fileName为\0，找到位置之后，又需要从i节点位示图中查找空闲i节点，最后便是查找空闲块，但是对于普通文件来讲，因为未向其中写入数据，因此可以暂时不给他分配磁盘块，只包留一个fcb信息，对于目录文件，我们需要分配一块的磁盘块来存放目录，初始化的时候把.和..的信息放入其中。在创建好文件之后，因为修改了目录文件的contents字段和length字段，并且其字段信息是存在内存中的，并没有写到磁盘块中，因此我们需要把ifContentChange和ifFcbChange字段置为1，方便之后关闭文件回写文件的信息。

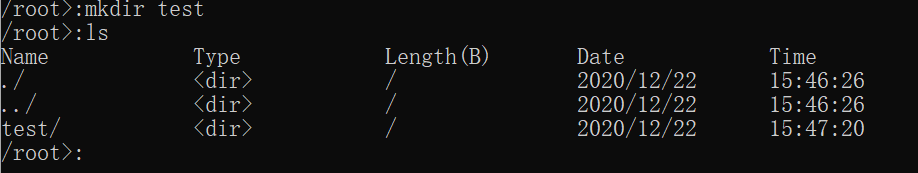


图9 创建目录

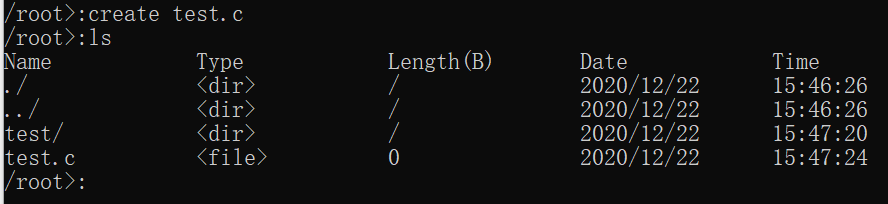


图 10 创建普通文件

5.close(int fd)

close操作为关闭指定文件描述符的文件。

关闭操作实际上只是把打开文件中的free字段置为0，标志该区域未被使用。除此之外，我们还需要根据打开文件的ifContentChange和ifFcbChange字段来判断当前文件是否被改变过，如果其被改变过，还需要回写我们所改变的信息，把信息放回到磁盘中，保持数据的一致性。然后把当前的文件描述符改变为其父目录的文件描述符。

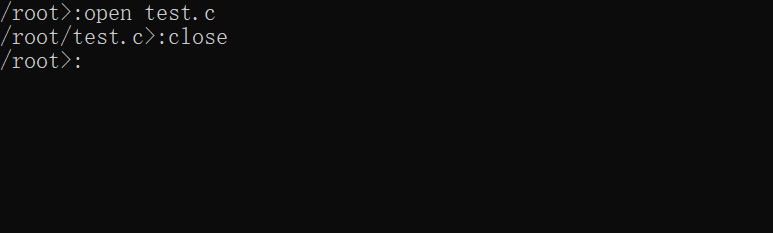


图11 关闭操作

6.removeFile (unsigned char \*path)

remove操作为删除当前目录下指定名称的文件。

对于删除文件，我们首先需要判断该文件是否被打开，如果其是被打开的状态，那么其不能被删除，需要先关闭文件才能对其进行删除。对于删除文件如果是普通文件那么便直接寻找到其所占用的物理块然后把位示图中信息置为0，然后再把i节点表中的信息置为0便可，但是对于目录文件，我们需要进行递归删除，需要从目录的最里层进行删除，最后才删除该目录文件。

删除完成之后，我们也需要把当前目录的ifContentChange和ifFcbChange置为1，因为我们对其contents字段和length字段进行了改变。

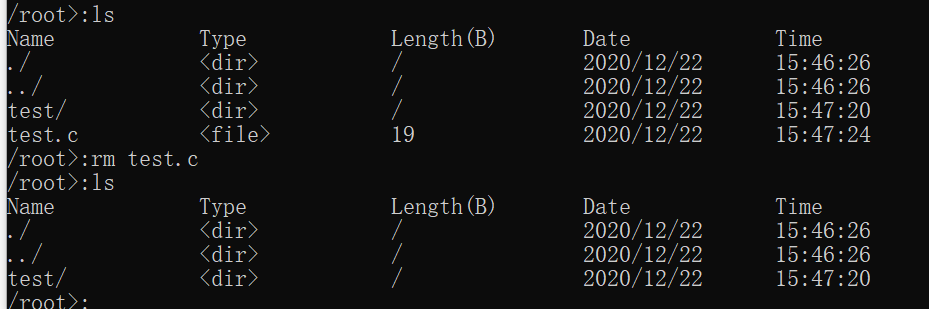


图12 删除操作

7.read()

read操作是读取当前文件数据块中的信息，并且可以指定指针来进行随机读取。

读操作只需要根据文件的fcb信息查找到文件的磁盘块，然后再根据设置的指针来读取打印出磁盘块中的内容便可，其读取的结束标志是文件的length，不是c语言中的结束符\0，因为在磁盘中，我们不是删除文件的时候并未真正的把数据给移除掉，而是把存在的标志符置为0，因此信息还存在磁盘当中，如果以\0为读取结束标志，很肯读取到之前的脏数据。

8.write()

对于写操作，我们需要向磁盘块中填充数据，写操作分为三种，截断写，为删除之前文件信息，重新开始写，追加写，直接从文件尾部开始写，覆盖写，指定一指针，从指针位置开始覆盖文件。

同样写操作需要先找到磁盘块，再根据文件的length判断其是直接文件还是单级索引文件，然后再找到磁盘块进行写操作，对于我设计的文件系统，需要注意的是直接块和单级索引直接的转换，当数据的总长度大于一个磁盘块时，需要把直接块转换为单级索引，此时需要分配一个磁盘块来充当索引块，还有一个磁盘块来装多出来的数据，而当修改后的数据是小于一个磁盘块的时候，我们便需要删除索引块，把i节点中块号信息直接指向直接块从而实现转换。

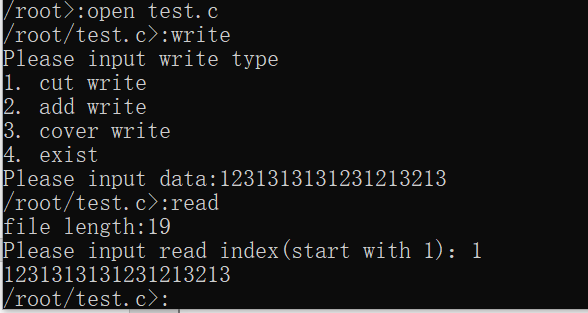


图13 读写操作

## 三、实验总结

1. 系统优缺点总结

优点：

1.采用单级索引结构，支持文件的随机读写。

2.不对文件目录进行扩充，从而把目录存入内存中，加快文件的检索。

3.存放文件数据时涉及了直接块和单级索引的转换，不会因为当写入数据小于一块时而浪费一块磁盘块。

缺点：

1.因为采用的为单级索引结构，因此只能存放小文件。

2.未对目录文件进行扩充，因此一个目录下所能创建的文件数量一定。

1. 个人总结

经过这次文件系统设计，我对于文件系统的整个架构的认识更为清晰，对于整个文件系统的设计实际上就是对于逻辑结构、物理结构、目录结构、FCB结构等等相关数据结构的设计，只要数据结构设计完整了，其文件系统的大题框架便出来了，因此对于文件系统的优化需要从数据结构上入手。再者对于c语言的使用更加娴熟了，对于各种数据类型之间的转换、链表指针的使用更加的熟练了。