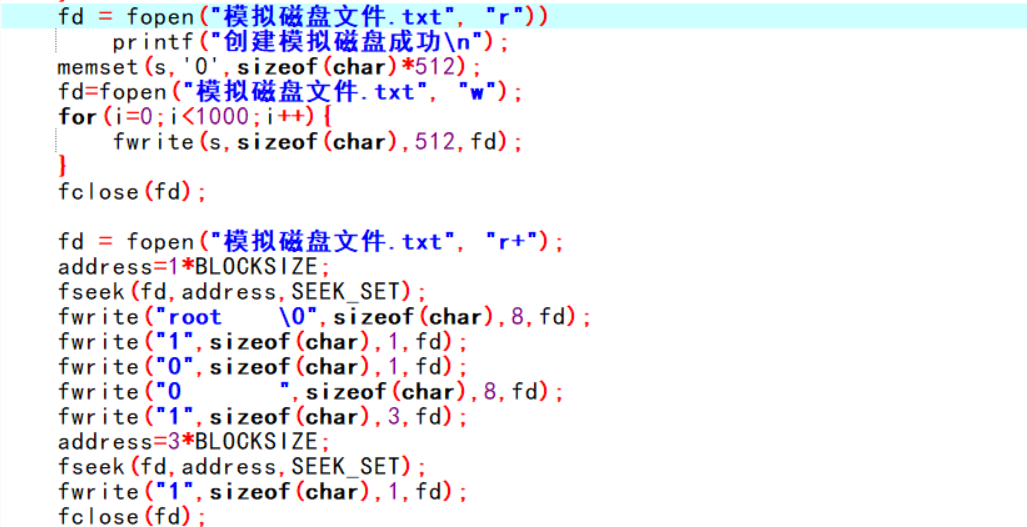
* **测试中出现的问题：**

1. **模拟磁盘文件出现BUG**

**问题1：**当开始运行代码的时候，结束后；重新打开文件的代码并进行运行，发现文件进行了重新的初始化，即没有保存之前的内容。这会产生一个问题：这里假设我们编写好了写入文件的代码，但是文件系统设计初衷是当文件关闭后，此时写入的内容依然存在。

**问题2：**同时忽略了一个重要的问题就是：u\_ofile表、file表要再文件系统重新进入的时候都要初始化；而不是仅在创建模拟磁盘文件的时候才创建。因为u\_ofile表和file表是记录打开文件的信息fcb；所以当文件系统关闭后，此时打开文件的内容必须全部关闭，即对用表项赋值为-1。

**第一次代码：**



**更改的代码**



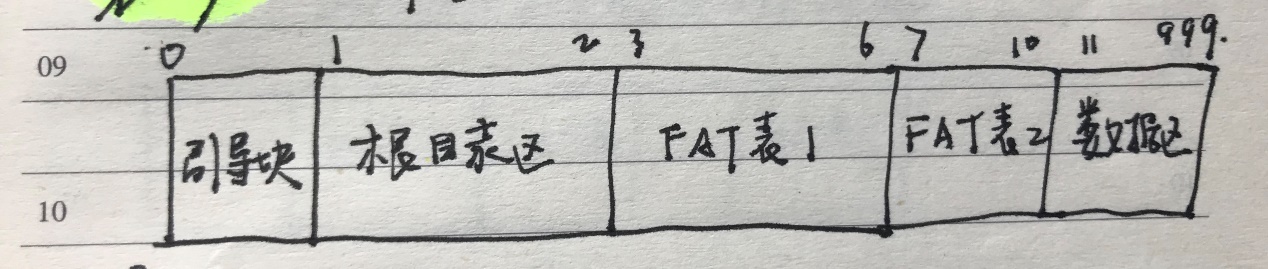
**更改说明：**

在初始化函数中增加判断条件，即**if((fd = fopen("模拟磁盘文件.txt", "r")) == NULL)，**当且仅当模拟磁盘文件不存在的时候，才会进行模拟磁盘的创建及初始化。同时初始化要进行文件的赋0；及ROOT根目录区的重新赋值。

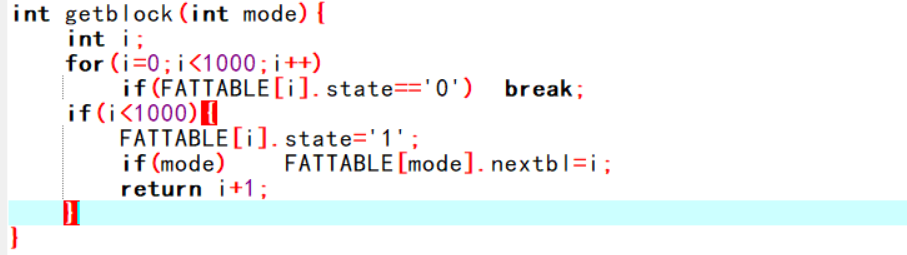
1. **为文件/目录增加物理块代码**

**问题说明：**这次的代码错误不是明显，但是当FAT表（即所有的物理块都被是用完毕后），会产生巨大的问题。代码是根据文件卷的设计，但是数据区域的物理块的为**11-999**（即999-11=989个物理块）；但是代码中确实设计了1000个物理块，即FAT表的容量为1000。所以当文件写的内容超过了989个物理块会出现问题。

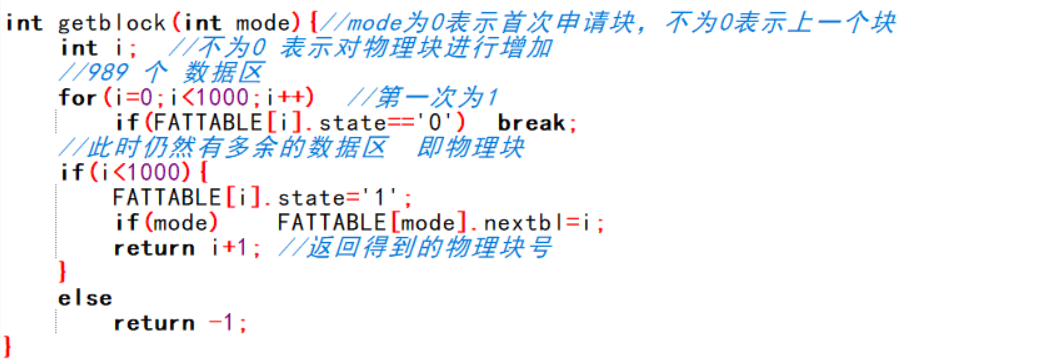
同时，因为物理块设计的非常大，总共989个物理块。所以完全可以满足创建文件/写入文件内容，所以getblock没有设返回值。但是实际上要设的，因为当物理块被使用完后，会产生不可增加物理块的问题，此时应该返回报错。

****

**第一次代码：**



**更改后的代码：**



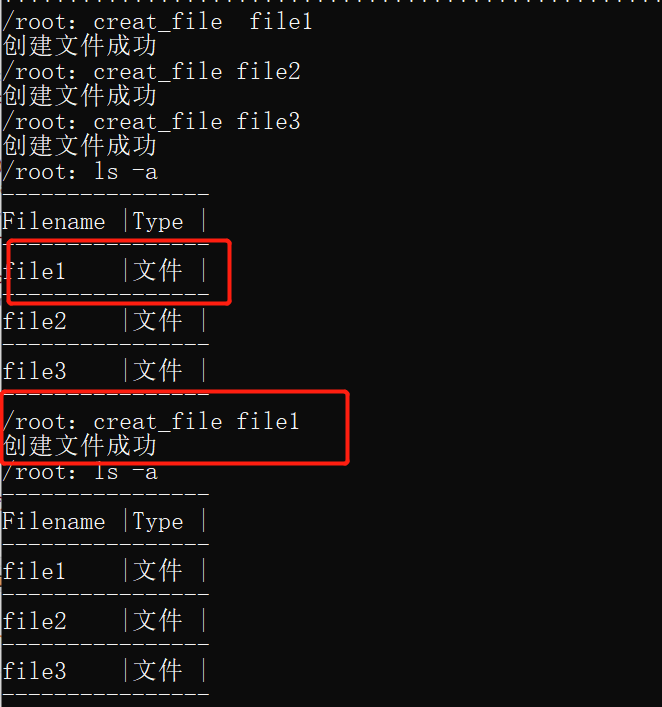
**更改说明：**

在第一的代码基础上，物理块的遍历范围从1000块，更改为文件卷所设计的989块的范围；同时为了完善文件系统，增加了返回报错的情况，即此时文件系统无可用物理块。

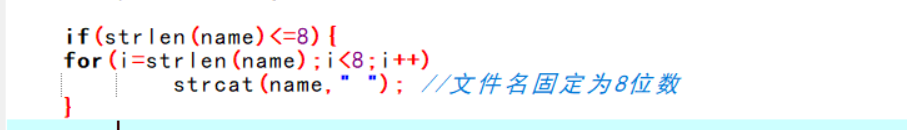
1. **创建文件时文件和文件重名的问题**

**问题说明：**在昨天对文件卷的初始化的代码进行详细读写以及发现代码BUG及更改后。对创建文件及目录的代码进行相信检查。

文件的创建首先要满足文件名在相同的目录下不可重复的问题，单数可以在不同的目录下创建相同名字的文件，基于这个原理我对creat\_s()函数进行代码的测试。但是我发现文件的创建确实不满足文件名不可重复的要求。如下图：可以创建两个重复名字的文件，如下图：在连续创建了三个文件file1、fiel2、file3后，使用”ls -a”命令查看全部文件，可以看见类型全部为文件类型；但是之后，再次创建名为file1的文件后，文件依然可以创建成功，并且覆盖了之前的file1文件，所以这里有了很大的BUG。



**代码更改：**

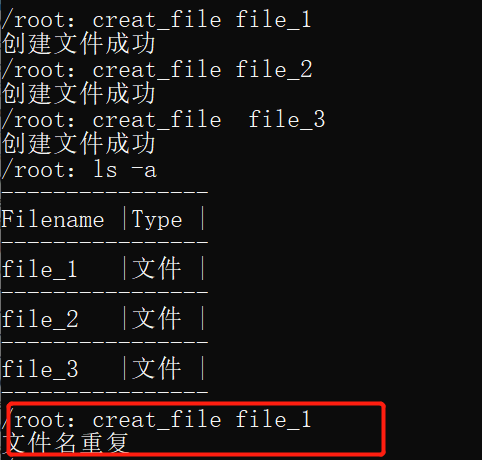


**更改说明：**

在发现了问题之后，我通读了creat\_s(filename)函数的代码。因为creat\_s()函数的逻辑是首先是调用Find\_name(filename)底层函数进行文件名重复问题的判断。通读了Find\_Name(filename)函数之后，并未找出代码的错误所在。

因为出错在文件名上，所以对此数据结构进行了查看，发现文件名的字节量为8bit。假设文件名为**file1**时候，在内存中存的结果为**file1000;**因为文件在初始化时候，将文件权保护初始化为0；但是它在数组中**file1xxx**未知**，**所以我的思路是将每个传进的文件名进行更改，将多余的字符填充为空格。

**更改后：**



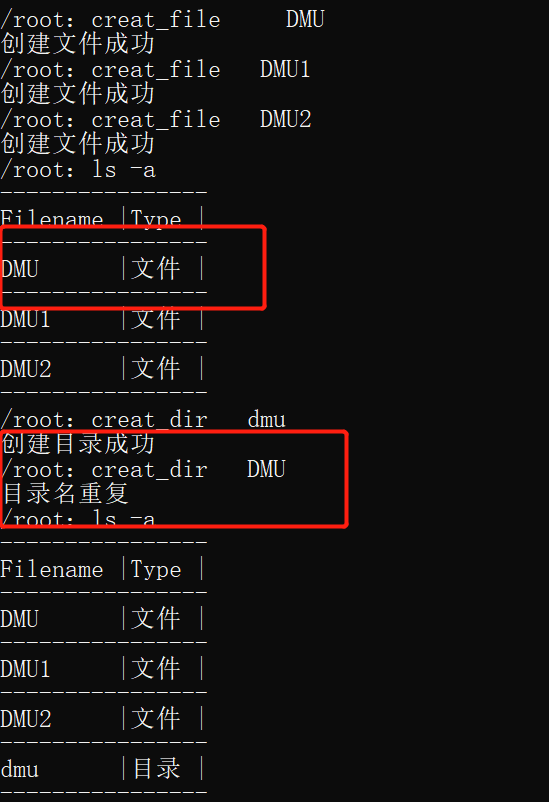
1. **关于目录名和文件名重复BUG**

**问题说明：**

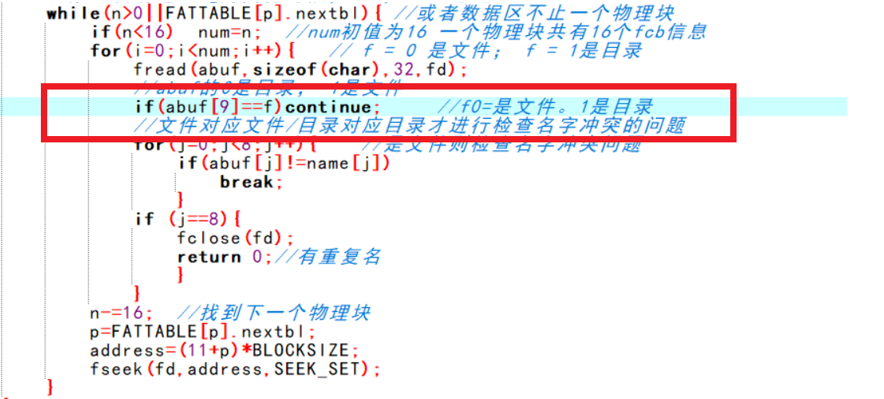
在发现了文件名可重复的问题后，我发现了**目录名也可以重复的问题**，但是创建文件和目录的函数都需要调用**Find\_name()函数**，所以目录名的重复的问题也解决了。接下来我继续对问题进行调试修改。

但是当我操作的时候，发现文件名和目录名不可重复。在客观上，文件名和目录名是可以重复的，所以经小组商议确定了这是个BUG问题后，我对此进行代码的DEBUG。

如下图：在连续创建了三个文件DMU、DMU1、DMU2后，使用”ls -a”命令查看全部文件，可以看见类型全部为文件类型；但是之后，在创建名为DMU的目录后，发现了目录名重复的问题。



**代码更改：**

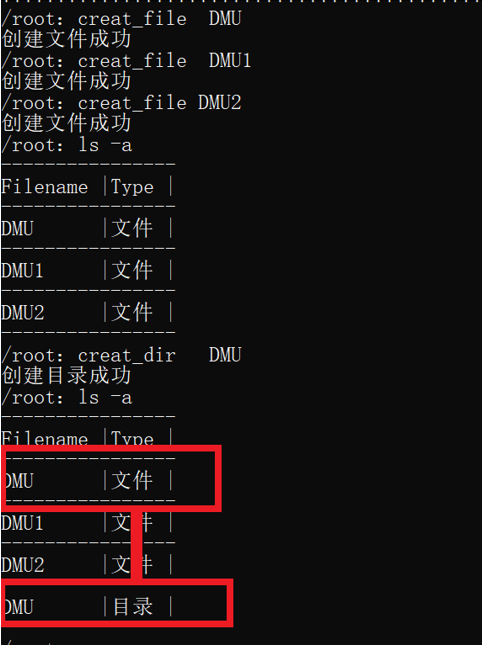


**更改说明：**

因为创建目录的函数依然首先调用Find\_Name()函数进行文件名是否重复的判断。所以我又仔细查看了此代码，但是没有发现任何问题。之后我对数据结构进行了查看，忽略了文**件/目录的类型**的判断。

在物理块中查找FCB的文件名/目录名时候，首先要进行判断是对哪种类型进行判断：文件/目录/文件和目录。所以，带着此问题，我发现了代码的错误问题：在内存找出子目录/子文件的fcb后，要找出此FCB的文件类型，因为当类型为文件时，就不需要进行下一步的判断。

**更改后：**



1. **关于对通过绝对路径对文件的操作。**

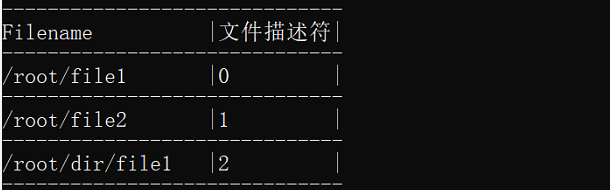
今天是测试的最后一天，也是最重要的一天。因为文件系统的测试关键在于打开文件、

读取文件、写文件等对文件功能方面的测试。

因为目前关于**open\_s()、creat\_s()、read\_s()**等具体的函数已经通过组员的分工进行实现；但是我们没有商量通过什么命令方式对文件进行操作，这里我们通过上学其的文件系统章节中**绝对路径**进行对文件操作。

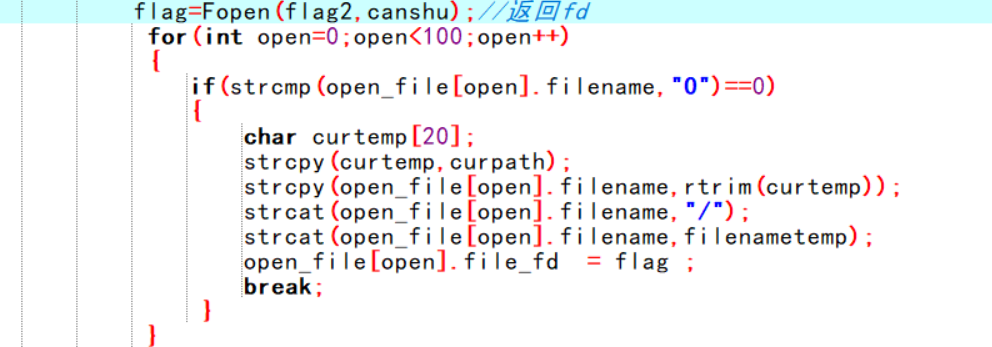
**（1） 打开文件表Open\_FIie的数据结构设计**

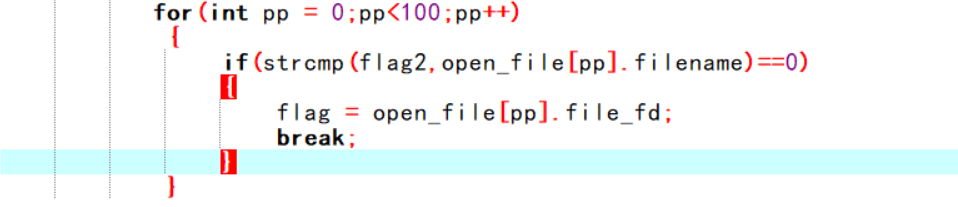
目前中Fiie表、u\_ofile表中存放了打开文件的各种描述：文件打开类型、文件名等；同时通过**文件描述符**进行对文件的操作。但是仅仅通过文件描述符对文件操作不能体现文件系统设计的初衷。所以这里采用的绝对路径的方式对文件进行读取。所以我们此时要增加一个数据结构：Open\_File表，其中记录了打开文件的绝对路径名、打卡文件的文件描述符。如图所示：



**（2） 如何判断绝对路径**

这里我们需要从新更改对open\_s()、creat\_s()、read\_s()等命令读取的操作。首先再对open\_s（）的命令输入时，要将返回的文件描述符对Open\_File表中设置，并增加文件的绝对路径、返回的文件描述符。同理，当室友creat\_s（）、read\_s（）函数对文件进行读取和写入工作时候，要循环遍历Open\_file表进行文件插曲获得文件描述符。





**（3）更改结果**

