OSlab5 实验报告

王亚楠 141220107

邮箱：[wyn1996@hotmail.com](mailto:wyn1996@hotmail.com)

# 实验进度

已完成实验的基本要求，包括文件的格式化、open、close、read、write、seek操作。拓展部分未进行深入探究。

# 实验心得

这次实验遇到了好多奇怪的错误，可能也不能算奇怪，就是平时不怎么会注意到的问题，这次实验中都暴露出来了。

这次实验的话我基本上是写死的，一开始可以选择格式化的文件以及数量，但是确定以后无法更改。

在格式化的时候采用了bitmap+dir+inode+data的形式。

文件结构类似PCB的结构



文件的访问通过数组下标实现。其中opened表示文件是否被使用，offset记录文件指针的偏移量，size记录文件的大小，block数组用来存放文件的inode块，这样方便文件数据的访问。

在打开文件的过程中，我就将文件的inode块放入到了fcb的block中去。

对于读写文件，我采用了一个扇区一个扇区的方式对文件的读写，也处理了跨边界的问题。

下面说说我遇到的问题。

1. 由于我的game.bin文件很大，有一百多兆，所以讲义上的inode数组开的过小，使得出现数组越界发生段错误。后来我将inode数组大小调大以后便可以正常进行格式化。但是，我采用了FCB数组来进行文件的操作，起初我将数组定义的太大，使得整个lab运行需要持续好久的时间，后来通过查看kernel.bin中的elf才发现了kernel中静态数据区的大小已经超过了100MB，最后，我将FCB数组大小变小。这个问题困扰了整整一天，也感谢欧先飞在这个问题上对我的帮助。
2. 起初我格式化文件的时候是采用了bitmap+dir+inode1+data1+inode2+data2+…的结构，后来在文件的读取上发现难度较为麻烦，因为每次寻找相对应的文件的数据块的偏移量都是需要重新计算的。后来我将采用了bitmap+dir+inode+data的格式来对文件进行格式化，这样一来就简化了后续对文件数据块的定位和访问。
3. 在从磁盘上将文件读入到dir中时，发现原来的readseg函数无法处理跨边界的情况，后来自己重新写了readseg，使得能够处理跨边界的情况。
4. 在将文件操作分装成系统调用的过程中，对于一些要取地址的变量，起初忘记加上段的偏移量，这个问题也困扰了我很久，后来才发现。