实验五

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。

实验内容

- 基本功能（格式化、目录文件的增、删、显示、改名等）
- 打印FAT数据（按16进制格式显示，每行16项）
- 实现顺序、随机读写
- 实现写入目录、文件多于1个扇区数据信息
- 扩展功能
    - 扇区大小可变 （扇区字节=1024～64）
    - 索引文件（基本功能）

实验方法

在 linux 下使用C语言实现FAT文件系统，大部分参照书本来实现，各种要求参照书本。

实验过程和结果

由于功能较多，此处仅展示部分功能，同时修正了验收时出现的问题。

物理块的大小为1024。

help命令

```
S1-pro:/$ help
format                  格式化文件系统
cd [dir]                进入目录
mkdir [dir] [num]       创建目录
rmdir [dir]             删除一个空目录
ls [-n|-l|-o]           列出文件
create [file]           创建文件
rm [file]               删除文件
open [dir|file]         打开目录或文件
close [dir|file|-a]     关闭目录或文件
write [-c|-w|-a]        写入文件
read [-n|-s]            读取文件
showfat [-l|-a]         显示 FAT 表
help                    显示命令提示
exit                    退出文件系统
S1-pro:/$
```

ls命令

```
ls -n  # 以平铺方式展示文件
ls -l  # 详细展示文件信息，包括大小，所在起始物理块，创建时间等
ls -o  # 新添加的功能，列出已打开的文件，即USEROPEN，方便调试
```

```
dobychao@S1-pro: ~/proj/ex5
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)
dobychao@S1-pro:~/proj/ex5$ ./main
S1-pro:/$ ls
/       a.txt
S1-pro:/$ ls -l
0           6    1024  2022-12-09      13:27:50        /
1           7       7  2022-12-09      14:09:36        a.txt
S1-pro:/$ open a.txt
S1-pro:/$ ls -o
0       /
1       /a.txt
S1-pro:/$
```

展示 FAT 表

```
showfat -l  # 仅展示占用的物理块以及其链接关系
showfat -a  # 展示整个 FAT 表
```

```
S1-pro:/$ showfat -l
 0000
 0001 -> 0002
 0003 -> 0004
 0005
 0006
 0007
S1-pro:/$ showfat -a
0000(ffff) 0001(0002) 0002(ffff) 0003(0004) 0004(ffff) 0005(ffff)
0006(ffff) 0007(ffff) 0008(0000) 0009(0000) 000a(0000) 000b(0000)
000c(0000) 000d(0000) 000e(0000) 000f(0000) 0010(0000) 0011(0000)
0012(0000) 0013(0000) 0014(0000) 0015(0000) 0016(0000) 0017(0000)
0018(0000) 0019(0000) 001a(0000) 001b(0000) 001c(0000) 001d(0000)
001e(0000) 001f(0000) 0020(0000) 0021(0000) 0022(0000) 0023(0000)
0024(0000) 0025(0000) 0026(0000) 0027(0000) 0028(0000) 0029(0000)
002a(0000) 002b(0000) 002c(0000) 002d(0000) 002e(0000) 002f(0000)
0030(0000) 0031(0000) 0032(0000) 0033(0000) 0034(0000) 0035(0000)
0036(0000) 0037(0000) 0038(0000) 0039(0000) 003a(0000) 003b(0000)
003c(0000) 003d(0000) 003e(0000) 003f(0000) 0040(0000) 0041(0000)
0042(0000) 0043(0000) 0044(0000) 0045(0000) 0046(0000) 0047(0000)
0048(0000) 0049(0000) 004a(0000) 004b(0000) 004c(0000) 004d(0000)
004e(0000) 004f(0000) 0050(0000) 0051(0000) 0052(0000) 0053(0000)
0054(0000) 0055(0000) 0056(0000) 0057(0000) 0058(0000) 0059(0000)
005a(0000) 005b(0000) 005c(0000) 005d(0000) 005e(0000) 005f(0000)
```

BITMAP

使用位释表，可以高效的寻找到空闲块的块号。

随机读写

关于write命令的解释：

- -c 截断写：放弃文件原有内容, 重新写文件
- -w 覆盖写：修改文件从当前读写指针所指的位置开始的部分
- -a 追加写：在原文件的最后添加新的内容
- 默认模式为截断写

这里以写入 a.txt 作为演示，首先写入内容"abcdefg"：

```
S1-pro:/$ write a.txt
please enter the file content, use "!q" in the end of the line to quit
abcdefg!q
S1-pro:/$
```

接着在第二个字符处写入"168"，读出文件：

```
S1-pro:/$ write a.txt -w
please input location: 1
please enter the file content, use "!q" in the end of the line to quit
168!q
S1-pro:/$ read a.txt
a168efgS1-pro:/$
```

然后在结尾处追加写入"fun"，读出文件：

```
S1-pro:/$ write a.txt -a
please enter the file content, use "!q" in the end of the line to quit
fun!q
S1-pro:/$ read a.txt
a168efgfunS1-pro:/$
```

使用`close a.txt`关闭并保存文件，然后用`ls -l`查看相关信息，可以注意到文件长度的变化：

```
S1-pro:/$ close a.txt
S1-pro:/$ ls -l
0            6    1024  2022-12-09      13:27:50        /
1            7      10  2022-12-09      14:09:36        a.txt
S1-pro:/$
```

使用`read a.txt -s`随机读命令，读取从第四个字符开始的三个字符（这里可以发现文件没有打开时就执行命令会有错误提示）：

```
S1-pro:/$ read a.txt -s
read: file is not open
S1-pro:/$ open a.txt
S1-pro:/$ read a.txt -s
please input location: 3
please input length: 3
8efS1-pro:/$
```

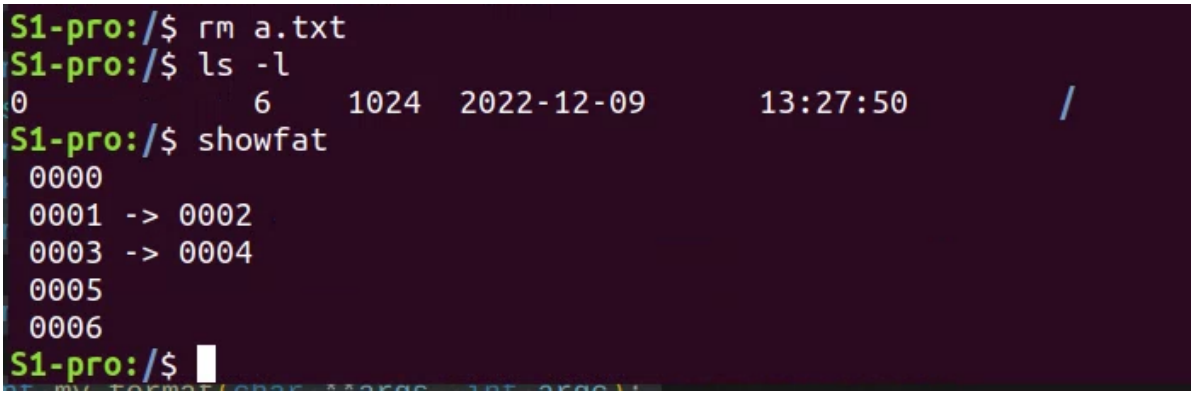
写入大量内容使得文件内容超出一个物理块（这里直接将 ex5/filesystem.h 的内容复制进去）：

```
dobychao@S1-pro: ~/proj/ex5
文件(F)  编辑(E)  查看(V)  搜索(S)  终端(T)  帮助(H)

unsigned short get_time(struct tm *timeinfo);
unsigned short get_date(struct tm *timeinfo);
char *trans_date(char *sdate, unsigned short date);
char *trans_time(char *stime, unsigned short time);

char *my_strtok(char *str, const char *ctr);
!q
S1-pro:/$ ls -l
0            6    1024  2022-12-09      13:27:50        /
1            7      10  2022-12-09      14:09:36        a.txt
S1-pro:/$ close a.txt
S1-pro:/$ ls -l
0            6    1024  2022-12-09      13:27:50        /
1            7    3218  2022-12-09      14:09:36        a.txt
S1-pro:/$ showfat -l
 0000
 0001 -> 0002
 0003 -> 0004
 0005
 0006
 0007 -> 0008 -> 0009 -> 000a
S1-pro:/$
```

验证得到跨物理块存储没有问题。

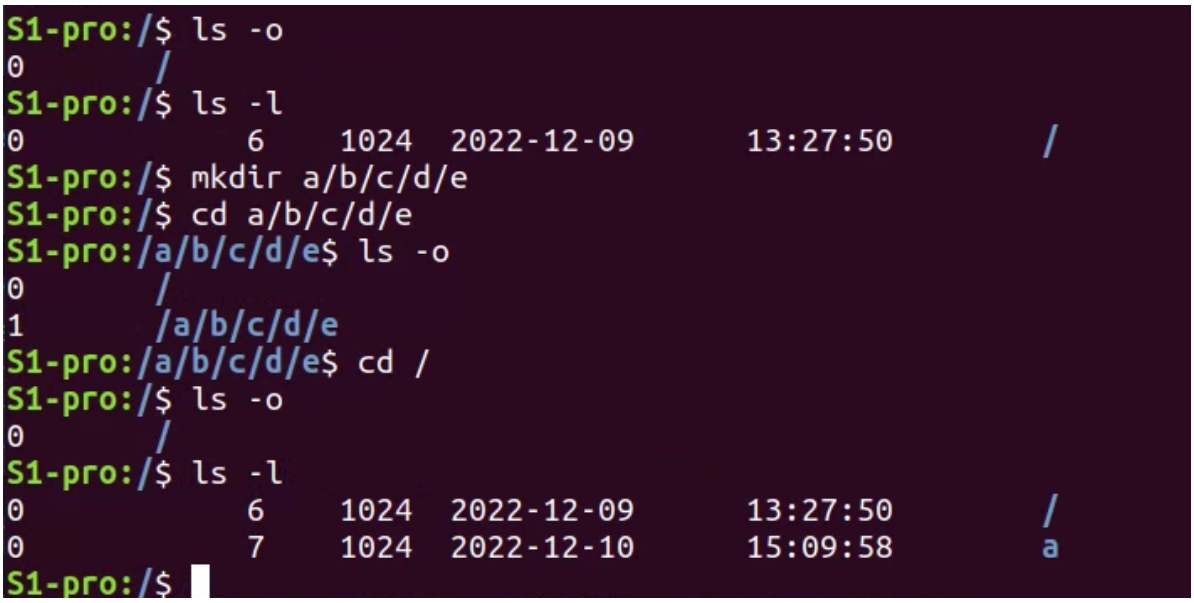
删除文件并释放物理块

使用命令`rm a.txt`删除刚刚的文件（在验收时未能释放全部占用的物理块，此处已修复）：



递归创建文件夹

使用命令`mkdir a/b/c/d/e`，并用`cd`命令进入，查看相应信息：



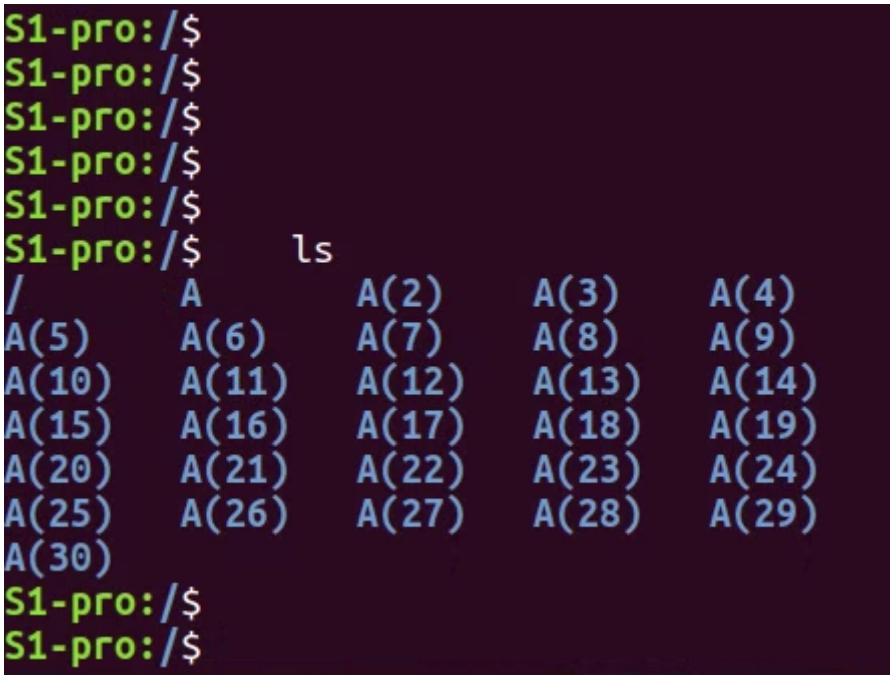
批量创建文件夹

试验功能，目的是测试跨物理块的 FCB 存储，执行命令`mkdir A 30`：

```
S1-pro:/$ format
S1-pro:/$ ls
/
S1-pro:/$ ls -l
0            6    1024  2022-12-10      15:24:30        /
S1-pro:/$ mkdir A 30
S1-pro:/$ ls -l
0            6    2048  2022-12-10      15:24:30        /
0            7    1024  2022-12-10      15:24:44        A
0            8    1024  2022-12-10      15:24:44        A(2)
0            9    1024  2022-12-10      15:24:44        A(3)
0           10    1024  2022-12-10      15:24:44        A(4)
0           11    1024  2022-12-10      15:24:44        A(5)
0           12    1024  2022-12-10      15:24:44        A(6)
0           13    1024  2022-12-10      15:24:44        A(7)
0           14    1024  2022-12-10      15:24:44        A(8)
0           15    1024  2022-12-10      15:24:44        A(9)
0           16    1024  2022-12-10      15:24:44        A(10)
0           17    1024  2022-12-10      15:24:44        A(11)
0           18    1024  2022-12-10      15:24:44        A(12)
0           19    1024  2022-12-10      15:24:44        A(13)
0           20    1024  2022-12-10      15:24:44        A(14)
0           21    1024  2022-12-10      15:24:44        A(15)
0           22    1024  2022-12-10      15:24:44        A(16)
0           23    1024  2022-12-10      15:24:44        A(17)
0           24    1024  2022-12-10      15:24:44        A(18)
0           25    1024  2022-12-10      15:24:44        A(19)
0           26    1024  2022-12-10      15:24:44        A(20)
0           27    1024  2022-12-10      15:24:44        A(21)
0           28    1024  2022-12-10      15:24:44        A(22)
0           29    1024  2022-12-10      15:24:44        A(23)
0           30    1024  2022-12-10      15:24:44        A(24)
0           32    1024  2022-12-10      15:24:44        A(25)
0           33    1024  2022-12-10      15:24:44        A(26)
0           34    1024  2022-12-10      15:24:44        A(27)
0           35    1024  2022-12-10      15:24:44        A(28)
0           36    1024  2022-12-10      15:24:44        A(29)
0           37    1024  2022-12-10      15:24:44        A(30)
S1-pro:/$
```

命令输入鲁棒性

在命令前面或后面有多余的空格不会影响命令本身：



实验体会

经过这次实验，通过具体的文件存储空间的管理、文件的物理结构、目录结构和文件操作的实现，我加深了对文件系统内部数据结构、功能以及实现过程的理解。

当然，实际上的文件系统要比这个实验所写的复杂得多，这个实验对我而言首先是编码能力的增强，这是一个纯 C 语言编写的程序，不同于一些应用层的程序，对算法和数据结构以及 C 语言基础的比较非常高，对我来说这既是一个复习的机会，也是一个学习的机会。就操作系统课程而言，这个实验对我最大的帮助就是加深了对理论知识的理解，对文件系统有了更为直观和细致的理解。

参考文献

https://github.com/leslievan/Operator_System/tree/8fe3990aef8d1c950c6169a85e8673d8c8840890/Operator_System_Lab5