操作系统课程设计实验报告

实验题目：实验五 简单文件系统的实现

姓　　名：张孜远

学　　号：20151521

组　　号：04

专　　业：卓越学院 智能计算与数据科学

班　　级：20186211

老师姓名：任彧老师

日　　期：2022 年 12 月 20 日

目　　录

一 题目介绍.....1
二 实验内容与思路.....1
三 遇到问题及解决方法.....2
四 核心代码及实验结果展示.....2
五 个人实验改进与总结.....8
5.1 个人实验改进.....8
5.2 个人实验总结.....8
六 参考文献.....9

一 题目介绍

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。

本 FAT 文件系统是仿照 FAT16 文件系统来设计实现的，但根目录没有采用 FAT16 的固定位的置、固定大小的根目录区，而是以根目录文件的形式来实现的，这也是目前主流文件系统对根目录的处理方式。

二 实验内容与思路

实验内容：

1、基本功能（格式化、目录文件的增、删、显示、改名等）；

2、打印 FAT 数据（按 16 进制格式显示，每行 16 项）；

3、实现顺序、随机读写；

4、实现写入目录、文件多于 1 个扇区数据信息；

5、实现 mv 指令；（自主创新内容）

扩展功能：

扇区大小可变 （扇区字节=1024～64）：通过修改 BLOCK 块的大小，即可实现扇区大小可变的功能。

实验思路：

首先我们需要编写一个 main 函数来模拟 Linux 系统的命令行窗口，即获取用户输入的指令；

其次，我们使用 C 语言根据 FAT 文件系统的基本原理，初步构建一个文件系统；

接着，我们根据 FAT 文件系统的基本操作，编写相应的函数；

最后，通过实验验证的方式，对自己编写的 FAT 文件系统进行正确性的证明。

详细实验思路：

(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恢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、位示图或其他办法；

(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；

①my_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；

②my_mkdir: 用于创建子目录；

③my_rmdir: 用于删除子目录；

④my_ls: 用于显示目录中的内容；

⑤my_cd: 用于更改当前目录；

⑥my_cd: 用于创建文件；

⑦my_open: 用于打开文件；

三 遇到问题及解决方法

实验方法：

在虚拟机中 Linux 环境下使用 C 语言进行编程，参照书本与课堂要求实现 FAT 文件系统。

四 核心代码及实验结果展示

实验过程和结果：

前提补充：

1、 FAT 文件系统拥有 1024 块物理块，每块物理块的大小是 1024 字节，即 1KB；

2、 FAT 文件系统大小 = 1024 * 1024 = 2^20 字节，即 1MB；

3、 main.c：模仿 Linux 系统的命令行窗口，获取用户相应输入命令；file_system.c：用于编写 FAT 文件系统中的函数；file_system.h：用于（全局）变量、结构体、函数等声明；

第 0 步：

使用 make 命令对 FAT 文件系统进行初始化：

```
zhangziyuan@ubuntu:~/exp5$ make
gcc -g main.c file_system.c -o main
file_system.c: In function ‘my_mv’:
file_system.c:2157:25: warning: too many arguments for format [-Wformat-extra-ar
gs]
         fprintf(stderr, "mv: The file does not exist.\n", args[1]);
                         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
file_system.c:2163:25: warning: too many arguments for format [-Wformat-extra-ar
gs]
         fprintf(stderr, "mv: It is not a file, it is a catalogue.\n", args[1]);
                         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
zhangziyuan@ubuntu:~/exp5$ ./main
System initializing ...
```

第 1 步：使用 help 命令查看 FAT 文件系统的功能：

```
zhangziyuan@ubuntu:~/exp5$ ./main
System initializing ...
ubuntu:/$ 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]              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
mv [-b|-f|-i]           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
```

使用 mkdir 指令在 FAT 文件系统中递归创建三个循环嵌套的文件夹 a/b/c，通过 cd 指令进入文件夹 c 中，在其中建立三个文件，a.txt、b.txt 和 c.html，便于后续对 FAT 文件系统的测试。

第 2 步：使用 ls 命令查看当前目录（文件夹 c）下的内容：

1、 使用 ls -n 指令（默认打开方式），以平铺的方式展示文件；

2、 使用 ls -l 指令，详细展示文件信息，包括文件编号、文件所在的起始物理块编号、文件大小（文件中字符的个数）、文件创建时间（年、月、日、时、分、秒）、文件具体名称；

```
ubuntu:/$ mkdir a/b/c
ubuntu:/$ cd a/b/c
ubuntu:/a/b/c$ create a.txt
ubuntu:/a/b/c$ create b.txt
ubuntu:/a/b/c$ create c.html
ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
ubuntu:/a/b/c$ ls -l
1           10        0  2022-12-18      17:09:34        a.txt
1           11        0  2022-12-18      17:09:38        b.txt
1           12        0  2022-12-18      17:09:40        c.html
```

第 3 步：使用 showfat 命令展示 FAT 文件系统的 FAT 表：

1、 showfat -l 指令（默认打开方式），仅展示占用的物理块以及其链接关系；

2、 showfat -a 指令，可以展示整个文件系统的 FAT 表；

```
ubuntu:/a/b/c$ showfat -l
 0000
 0001 -> 0002
 0003 -> 0004
 0005
 0006
 0007
 0008
 0009
 000a
 000b
 000c
```

```
ubuntu:/a/b/c$ showfat -a
0000(ffff) 0001(0002) 0002(ffff) 0003(0004) 0004(ffff) 0005(ffff)
0006(ffff) 0007(ffff) 0008(ffff) 0009(ffff) 000a(ffff) 000b(ffff)
000c(ffff) 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)
```

说明：

1、左图：

0000 物理块是引导块；

0001-0002 物理块和 0003-0004 物理块是文件系统的 FAT 表 0 和 FAT 表 1，其目的是在数组中存储了每个物理块的指针；由于在 FAT 表 0 和 FAT 表 1 中的内存区域存了 1024 个 short 类型的数据，所以每个 FAT 表占 2K，也就是 2 个物理块；

0001-0002 物理块和 0003-0004 物理块中的内容，也就是 FAT 表 0 和 FAT 表 1 完全相同，通过磁盘数据镜像实现数据冗余，在成对的独立磁盘上产生互为备份的数据。其采用了磁盘阵列技术的 Raid 1 模型，提高了文件系统的读取性能、高数据安全性和可用性；

0005 物理块是一个表，其存储了每个块是否被使用过的二进制标记，即 0 或者 1；

0006 物理块存储了根目录；

0007 物理块存储了文件夹 a；

0008 物理块存储了文件夹 b；

0009 物理块存储了文件夹 c；

000a 物理块存储了文件 a.txt；

000b 物理块存储了文件 b.txt；

000c 物理块存储了文件 c.html；

2、右图：

展示了文件系统的 BITMAP，其使用位释表，可以高效的寻找到空闲块的块号。

第 4 步：使用 create、open、write、read、close 指令对文件进行一系列操作：

前提：

对文件的 write 和 read 操作需要在文件的状态是“打开”的情况下进行，对文件操作结束后需要使用 close 命令关闭已打开的文件。

open 指令：打开指定文件

write 指令：

1、 write -c（默认写方式）：截断写，即清空文件内原有内容，重新写入；

2、 write -w：覆盖写，即从当前读写指针所指的位置开始修改文件内容；

3、 write -a：追加写，即从文件尾指针开始写入新增内容；

read 指令：

1、 read -n（默认读方式）：读命令，读取文件中的所有内容；

2、 read -s：随机读命令，读取文件中的部分内容；

将内容写入 a.txt 为例：

1、首先使用 open 指令打开 a.txt 文件；

2、使用 write -c 指令将“zzy is a handsome boy”内容写入 a.txt 中；

3、使用 write -w 指令将 zzy 替换为 Tom，此时 input location 是 0；

4、使用 write -a 指令在文件最后新增“It is a nice day.”内容。

在每次对 a.txt 文件操作结束后，使用 read 指令观察 a.txt 文件中内容的变化，并验证 read -s 随机读的效果正确性，如下图所示：

```
ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
ubuntu:/a/b/c$ open a.txt
ubuntu:/a/b/c$ read a.txt
ubuntu:/a/b/c$ write a.txt -c
please enter the file content, use "!q" in the end of the line to quit
zzy is a handsome boy.!q
ubuntu:/a/b/c$ read a.txt
zzy is a handsome boy.ubuntu:/a/b/c$ write a.txt -w
please input location: 0
please enter the file content, use "!q" in the end of the line to quit
Tom!q
ubuntu:/a/b/c$ read a.txt
Tom is a handsome boy.ubuntu:/a/b/c$ write a.txt -a
please enter the file content, use "!q" in the end of the line to quit
It is a nice day.!q
ubuntu:/a/b/c$ read a.txt
Tom is a handsome boy.It is a nice day.ubuntu:/a/b/c$ read a.txt -s
please input location: 0
please input length: 21
Tom is a handsome boyubuntu:/a/b/c$ close a.txt
ubuntu:/a/b/c$ ls -l
1           10      39  2022-12-18      18:23:40        a.txt
1           11       0  2022-12-18      17:09:38        b.txt
1           12       0  2022-12-18      17:09:40        c.html
```

第 5 步：使用 write 指令将超出一个物理块的内容写入文件 a.txt，测试跨物理课存储的正确性：

```
ubuntu:/a/b/c$ showfat
 0000
 0001 -> 0002
 0003 -> 0004
 0005
 0006
 0007
 0008
 0009
 000a -> 000d -> 000e -> 000f
 000b
 000c
```

第 6 步：使用 mv 指令对文件进行一系列操作：

0、mv：修改指定文件名称 或者 移动文件位置；

1、mv -b：如若修改的文件名已存在，则先将原来的文件备份后，再修改新文件；

2、mv -f：如若修改的文件名已存在，则不询问直接覆盖；

3、mv -i：如若修改的文件名已存在，则询问用户是否覆盖；

对文件夹 c 中的三个文件操作为例：

1、 使用 mv 指令将文件 a.txt 的名字修改为 c.txt；

2、 使用 mv -b 指令将文件 c.txt 的名字修改为 b.txt；

3、 使用 mv -f 指令将文件 b~.txt 的名字修改为 b.txt；

```
ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
ubuntu:/a/b/c$ mv a.txt c.txt
please print file&file operation or dic&file operation, namely using f and d in place of them.f
ubuntu:/a/b/c$ ubuntu:/a/b/c$ ls
c.txt   b.txt   c.html
ubuntu:/a/b/c$ mv c.txt b.txt -b
please print file&file operation or dic&file operation, namely using f and d in place of them.f
There already exists a file named b.txt
ubuntu:/a/b/c$ ubuntu:/a/b/c$ ls
b.txt   b~.txt  c.html
ubuntu:/a/b/c$ open b.txt
ubuntu:/a/b/c$ read b.txt
Tom is a handsome boy.It is a nice day.ubuntu:/a/b/c$ close b.txt
ubuntu:/a/b/c$ ls
b.txt   b~.txt  c.html
ubuntu:/a/b/c$ mv b~.txt b.txt -f
please print file&file operation or dic&file operation, namely using f and d in place of them.f
There already exists a file named b.txt
ubuntu:/a/b/c$ ubuntu:/a/b/c$ ls
b.txt   c.html
ubuntu:/a/b/c$ open b.txt
ubuntu:/a/b/c$ read b.txt
ubuntu:/a/b/c$ close b.txt
ubuntu:/a/b/c$ ls
b.txt   c.html
```

4、 使用 write 指令将字符串“aaaaaaaaaa”写入文件 b.txt；

5、 使用 create、write 指令将字符串“asdfghjkl”写入文件 a.txt；

```
ubuntu:/a/b/c$ open b.txt
ubuntu:/a/b/c$ write b.txt
please enter the file content, use "!q" in the end of the line to quit
aaaaaaaaaa!q
ubuntu:/a/b/c$ create a.txt
ubuntu:/a/b/c$ close b.txt
ubuntu:/a/b/c$ write a.txt
write: file is not open
ubuntu:/a/b/c$ open a.txt
ubuntu:/a/b/c$ write a.txt
please enter the file content, use "!q" in the end of the line to quit
asdfghjkl!q
ubuntu:/a/b/c$ close a.txt
ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
```

6、 使用 mv -i 指令将文件 a.txt 的名字修改为 b.txt；

7、 使用 mv 指令将文件 a.txt 的名字修改为 b.txt，并输出“文件名相同”警告；

```
ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
ubuntu:/a/b/c$ mv a.txt b.txt -i
There already exists a file named b.txt
mv: overwrite 'b.txt'?
please print 'y == yes, n == no' to continue
n
please enter the instrument again.ubuntu:/a/b/c$ ubuntu:/a/b/c$ ls
a.txt   b.txt   c.html
ubuntu:/a/b/c$ mv a.txt b.txt -i
There already exists a file named b.txt
mv: overwrite 'b.txt'?
please print 'y == yes, n == no' to continue
y
ubuntu:/a/b/c$ ubuntu:/a/b/c$ ls
b.txt   c.html
ubuntu:/a/b/c$ open b.txt
ubuntu:/a/b/c$ read b.txt
asdfghjklubuntu:/a/b/c$ close b.txt
ubuntu:/a/b/c$ ls
b.txt   c.html
ubuntu:/a/b/c$ create a.txt
ubuntu:/a/b/c$ ls
b.txt   a.txt   c.html
ubuntu:/a/b/c$ mv a.txt b.txt
There already exists a file named b.txt
please use 'mv XXX XXX -i' for more information.ubuntu:/a/b/c$ ls
b.txt   a.txt   c.html
```

8、使用 mv 文件名 地址名指令，将 a.txt 文件移动至 b 目录下；

```
ubuntu:/a/b/c$ mv a.txt /b
ubuntu:/a/b/c$ ls
b.txt   c.html
ubuntu:/a/b/c$ cd ..
ubuntu:/a/b$ ls
c       a.txt
ubuntu:/a/b$ open a.txt
ubuntu:/a/b$ read a.txt
helloubuntu:/a/b$ ls
c       a.txt
```

第 7 步：在根目录下使用 mkdir 指令（如：mkdir A 10）批量创建文件夹，测试跨物理块的 FCB 存储功能是否正确：

```
ubuntu:/$ mkdir A 10
ubuntu:/$ ls -l
0            6    1024  2022-12-18      19:06:12        /
0            7    1024  2022-12-18      19:06:16        A
0            8    1024  2022-12-18      19:06:16        A(2)
0            9    1024  2022-12-18      19:06:16        A(3)
0           10    1024  2022-12-18      19:06:16        A(4)
0           11    1024  2022-12-18      19:06:16        A(5)
0           12    1024  2022-12-18      19:06:16        A(6)
0           13    1024  2022-12-18      19:06:16        A(7)
0           14    1024  2022-12-18      19:06:16        A(8)
0           15    1024  2022-12-18      19:06:16        A(9)
0           16    1024  2022-12-18      19:06:16        A(10)
```

第 8 步：使用 rm、rmdir 指令对文件和文件夹进行删除操作，并释放对应的物理块：

```
ubuntu:/a/b/c$ ls
b.txt   c.html
ubuntu:/a/b/c$ rm b.txt
ubuntu:/a/b/c$ rm c.html
ubuntu:/a/b/c$ cd ..
ubuntu:/a/b$ rmdir c
ubuntu:/a/b$ ls
a.txt
ubuntu:/a/b$ rm a.txt
ubuntu:/a/b$ ls

ubuntu:/a/b$ cd ..
ubuntu:/a$ rmdir b
ubuntu:/a$ ls

ubuntu:/a$ cd ..
ubuntu:/$ rmdir a
ubuntu:/$ ls
/
ubuntu:/$
```

```
ubuntu:/$ showfat
 0000
 0001 -> 0002
 0003 -> 0004
 0005
 0006
```

五 个人实验改进与总结

5.1 个人实验改进

项目实现创新点说明	1、 修改原文件系统中关于 read 函数的错误： physic_block_num = (fat0 + physic_block_num)->id; 2、在各个地方设置了一定的缓存机制，如 cur_dir 的 FCB 等，可以有效加速文件系统的运行，减少磁盘 IO 操作； 3、**自主创新实现 mv 指令的所有内容**；

5.2 个人实验总结

实验体会：

刚开始，没有构思好数据结构，思路不顺畅，花了很久时间也没什么成果。后来我决定先在草稿纸上捋清思路，构思好数据结构及磁盘空间的总体设计方案。

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。同时本学期还在修读程序设计基础的我，对于源码的阅读也更容易写，对于参数的传递也更容易阅读。

当然，实际上的文件系统要比这个实验所写的复杂得多，我们只是通过这个实验达到：

1、 构建底层框架能力的提升；

2、 对于 C 语言基础能力的训练；

3、 对操作系统理论应用于实际的实战；

通过实践 FAT 文件系统，我对于文件系统有了更为直观和细致的理解。

六 参考文献

[1] 动手写一个简单的文件系统.Zhang Shurong.
https://zhangshurong.github.io/2019/01/02/%E5%8A%A8%E6%89%8B%E5%86%99%E4%B8%80%E4%B8%AA%E7%AE%80%E5%8D%95%E7%9A%84%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F
[2] Writing a File System in Linux Kernel.kmu1990.
https://kukuruku.co/post/writing-a-file-system-in-linux-kernel/
[3] File Systems: The Semester Project.sysplay.
https://sysplay.github.io/books/LinuxDrivers/book/Content/Part18.html
[4] 简单文件系统的实现范例：https://github.com/liu-jianhao/simpleFS
[5] Linux 文件系统详解：https://www.cnblogs.com/alantu2018/p/8461749.html
[6] Linux 文件系统的实现：https://www.cnblogs.com/pipci/p/10179502.html
[7] linux 文件系统管理的工作原理：https://www.cnblogs.com/alantu2018/p/8461680.html