

操作系统实验九/实验十报告

目录

操作系统实验九/实验十报告

目录

一、实验目的

二、特色简介

三、实验环境

四、实验方案

(1). FAT32文件系统

(2). 文件相关系统调用

(3). 基于文件系统的exec系统调用

(4). 为Shell添加dir、copy、move、del功能

(5). 实现文本编辑器

五、实验结果

1. 读取FAT32分区信息

2. 从磁盘读取并执行用户程序

3. 列出目录下的子目录和文件

4. 删除文件

5. 复制文件

6. 重命名（移动文件）

7. 测试文本编辑器

六、实验总结

七、参考文献

一、实验目的

原型保留原有特征的基础上，设计满足下列要求的新原型操作系统：

(1) 参考DOS的命令功能，实现文件管理控制命令cd、dir、del、copy等。

(2) 扩展MyOS的系统调用，在C语言中，程序可以调用文件创建、文件打开、文件读、文件写和文件关闭等系统调用。

(3) 编写一个测试的这些系统调用的C语言用户程序，编译产生可执行程序进行测试。

二、特色简介

1. 本次实验我实现了完整的FAT32文件系统功能，支持文件读写，支持多级目录，支持读取长文件名。

2. 实现了文件相关系统调用 (open、read、write、seek、close)
3. 修改exec系统调用以支持读取FAT32磁盘上的ELF格式用户程序
4. Shell程序中添加了dir、copy、move、del功能
5. 实现了一个简单的文本编辑器

三、实验环境

与之前实验大致相同，区别在于使用了hdiutil工具用于挂载磁盘镜像到物理机，并用newfs_msdos格式化磁盘镜像为FAT32格式

IDE：CLion 2018.2

C++编译器：g++ 7.3.0, Target: i386-elf

链接器：ld 2.30

二进制文件分析器：objdump 2.30

符号分析器：nm 2.3.0

EFL文件缩小：strip 2.3.0

ELF文件分析：readelf 2.3.0

主机操作系统：Mac OS 10.12

编辑器：Vim 8.0.1400、VS Code 1.21.0

汇编器：Nasm 2.13.02

虚拟机、调试器：Bochs 2.6.9

版本控制：Git 2.15.1

自动构建：GNU Make 3.8.1

四、实验方案

(1). FAT32文件系统

1. 所有FAT32文件系统操作基于的都是磁盘缓冲区SectorBuffer_t和FS_fetch函数。调用FS_fetch读取一个扇区时，如果缓冲区中已经存有这个扇区，那么并不会再次读取。如果缓冲区被修改了，那么会调用FS_fsync将缓冲区写回磁盘。
2. FAT32文件系统操作的第一步是读取BPB，获取文件系统起始扇区（在Unix上用bximage建立的磁盘镜像起始扇区一般是0，在Windows上用系统磁盘工具建立的VHD磁盘镜像起始扇区一般不为0）、数据区起始扇区、总扇区数，卷标等信息。
3. 打开文件实现为FS_fopen函数，它将文件路径进行拆分，从根目录（簇号为2）开始迭代地调用FS_find_file寻找下一层目录或最终的文件。FS_find_file打开当前目录的簇，遍历所有目录项，一个目录项通常对应一个文件或者子目录，但如果目录项属性为0x0f，那么该项是一个长文件名项，需要继续往下遍历，把多个长文件名项合成为一个。最后FS_find_file通过字符串匹

配确认是否有要寻找的下层目录或文件。

4. 读（FS_fread）写（FS_fwrite）文件都反复调用FS_fseek函数，FS_fseek将移动文件指针，如果移到了一个扇区的末尾，它将调用FS_get_FAT_entry查找FAT表，寻找文件的下一个簇号。读文件时，如果文件指针指向了文件末尾，那么继续读取会失败。写文件时，如果文件指针指向了文件末尾，那么文件大小会加一，并将新的字节写入文件。如果之前文件占用的簇用完了，那么FS_fseek会遍历FAT表找到一个空闲簇和对应的空闲FAT表项，将原来文件的最后一个FAT表项更新为新的簇号，再把空闲的FAT表项写入0x0FFFFFFF。
5. 新建文件（FS_touch）的原理是打开要新建的文件的父目录，然后调用FS_fwrite写入目录项。

(2). 文件相关系统调用

所有文件系统调用都是基于文件描述符（file descriptor）的。文件描述符是内核中的文件结构数组FileInfo_t fd[MAX_FD_NUM]的索引，通过文件描述符在内核和用户程序之间能够方便地指定一个文件。

目前我实现的文件相关系统调用包括：

```
1 //获取文件属性(包括文件大小、文件打开模式等)
2 int sys_fstat(int fildes, stat *buf);
3 //以mode模式打开路径为path的文件，如果mode为O_CREAT，那么如果文件不存在会新建文件
4 int sys_open(const char *path, uint32_t mode);
5 //从文件描述符fildes指定的文件中读取nbyte字节到缓冲区buf
6 int sys_read(int fildes, void *buf, size_t nbyte);
7 //从buf中写入nbyte字节到fildes指定的文件
8 int sys_write(int fildes, void *buf, size_t nbyte);
9 //关闭文件(将缓冲区写入磁盘)
10 int sys_close(int fd);
11 //移动文件指针offset个字节，起始位置由whence参数指定，包括从头开始(SEEK_SET, SEEK_CUR)
12 int sys_lseek(int fildes, int offset, int whence);
```

(3). 基于文件系统的exec系统调用

在实现文件系统后，exec系统调用接收一个字符串参数path，而像之前一样是起始扇区号。基于文件系统的exec函数更加安全，不会从任意磁盘区域加载程序了，同时调用sys_fstat获取文件大小，如果文件过大就不会加载。

(4).为Shell添加dir、copy、move、del功能

从实验七开始我的Shell就是一个普通的C++用户程序了，因此这些功能的实现都是基于文件相关系统调用在user space中完成的。

1. dir/ls：列出目录下所有文件

首先调用open打开目录，然后循环调用read，每次读取0x20字节的目录项。输出目录项中记录的文件名，文件大小和长文件名

2. copy/cp：复制文件

首先打开源文件，将源文件读入磁盘缓冲区，然后以O_CREAT模式调用open函数以创建目标文件，将磁盘缓冲区写入目标文件。最后要修改目录项中记录的目标文件的大小为源文件大小。

3. del/rm：删除文件

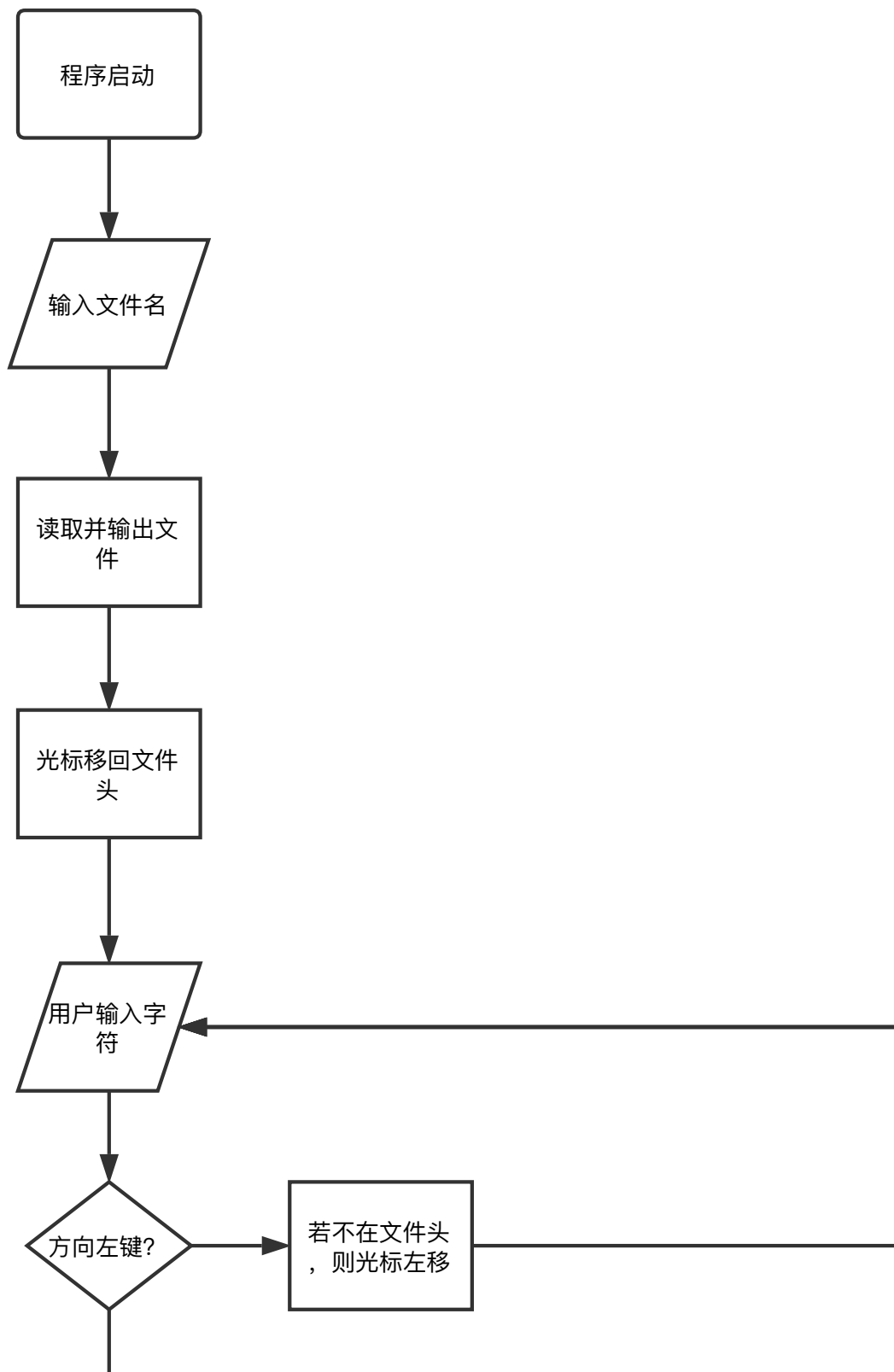
FAT32文件系统中删除一个文件仅仅是将目录项中记录的文件名首字符改为0xe5

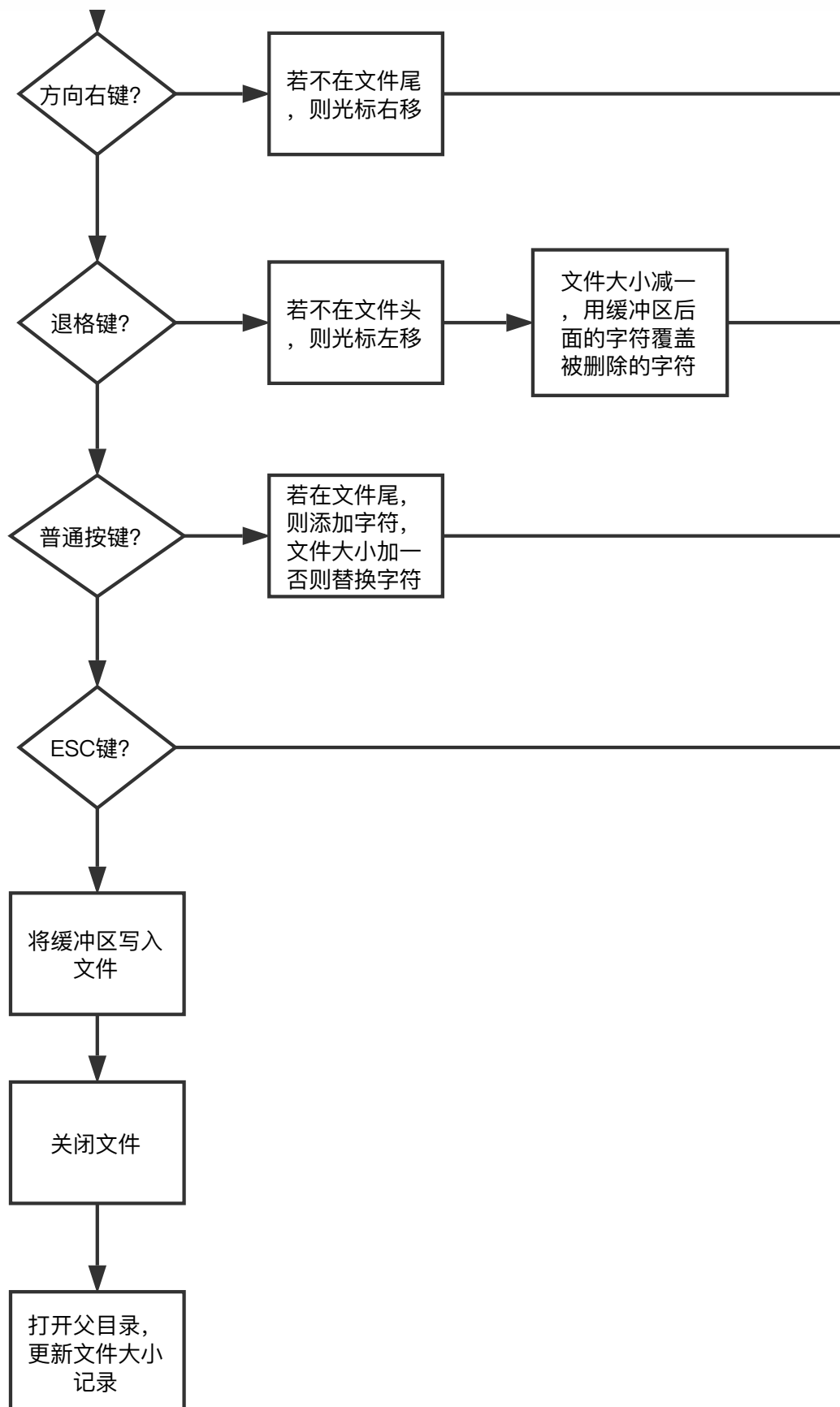
4. move/mv: 移动文件

修改目录项中记录的文件名，目前仅支持修改短文件名

(5). 实现文本编辑器

一个文本编辑器可以说是一个操作系统标配的基本功能了，也是一个很实用的功能。在实现文件系统后，我终于可以在我的操作系统中实现它了。程序流程图如下：





五、实验结果

1. 读取FAT32分区信息

```
-----FS INFO-----
This Volume Total have 81648 Sectors:
Cluster Size:512
NEW
Reserved Sectors: 32
Fat zone Total have 628 Sectors
:Data zone Start from Cluster: 1288
Data zone Total have 80360 Sectors:
Data zone Total have 80360 Clusters:
-----FS INFO END-----
```

2. 从磁盘读取并执行用户程序

如图，一开机就执行了Shell（SH.ELF）

```
file /SH.ELF opened
Parent Directory at Cluster: 2
File Start at Cluster: 58
File size (byte): 21204
```

3. 列出目录下的子目录和文件

```
HHOS />ls
Directory / opened
NEW 0 bytes
FSEVEN~1 0 bytes .fseventsd
BC.ELF 13496 bytes
EDIT.ELF 12808 bytes
SH.ELF 21204 bytes
TEST.ELF 12740 bytes
USR1.ELF 12740 bytes
USR2.ELF 12740 bytes
USR3.ELF 12740 bytes
USR4.ELF 16956 bytes
TEST.TXT 26 bytes
HHOS />
```

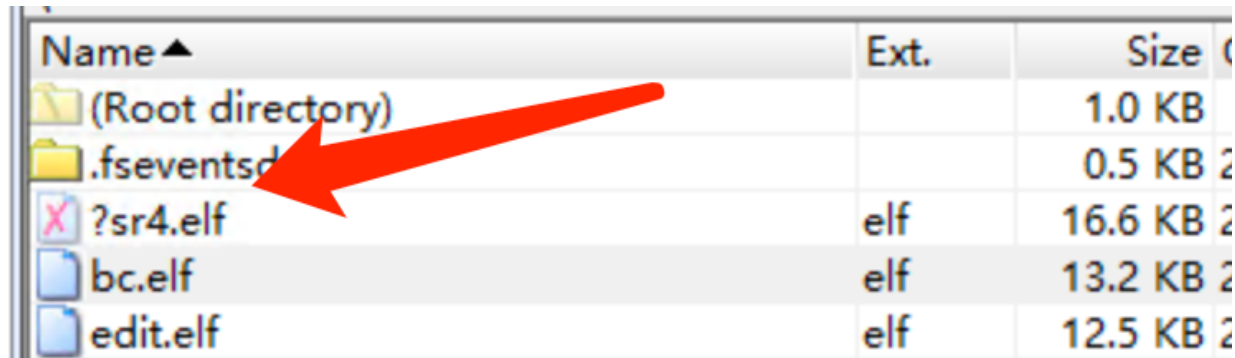
注意其中文件大小为0bytes的是子目录，它的名字超出了DOS8.3格式表示范围，在后面显示了它的长文件名


4. 删除文件

删除USR4.ELF

```
HHOS />rm USR4.ELF
Directory / opened
rm: succeeded
HHOS />
```

使用WinHex打开磁盘发现文件已被标记为删除



Name▲	Ext.	Size
(Root directory)		1.0 KB
.fsevents		0.5 KB
 ?sr4.elf	elf	16.6 KB
bc.elf	elf	13.2 KB
edit.elf	elf	12.5 KB

5. 复制文件

复制文件时必须输入绝对路径，下面的复制命令中，源文件为TEST.TXT，目标文件为T，源文件的内容为This is a fat32 test file.

```
HHOS />cp /TEST.TXT /T
file /TEST.TXT opened
Parent Directory at Cluster: 2
File Start at Cluster: 234
File size (byte): 26
FS_create: Filename: T
FS_create: Extension:
File /T created
Parent Directory at Cluster: 2
File Start at Cluster: 3
File size (byte): 0
cp: Ready to copy 26 bytes

Directory / opened
cp: succeeded
HHOS />
```

使用WinHex打开磁盘发现根目录下已近多出了文件T，大小为26B，文件大小也和原来的源文件相同

sh.elf	elf	20.7 KB	2018
T		26 B	2018
test.elf	elf	12.4 KB	2018
test.txt	txt	26 B	2018
usr1.elf	elf	12.4 KB	2018

文件内容也正确地复制了进去

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI	ASCII
00659968	54	68	69	73	20	69	73	20	61	20	66	61	74	33	32	20	This	
00659984	74	65	73	74	20	66	69	6C	65	2E	03	00	00	00	00	00	test	

6. 重命名（移动文件）

将USR3.ELF重命名为FAT.ELF，执行ls发现重命名成功

```
HHOS />mv USR3.ELF FAT.ELF
Directory / opened
mv: succeeded
HHOS />ls
Directory / opened
NEW 0 bytes
FSEVEN~1 0 bytes .fseventsd
BC.ELF 13496 bytes
EDIT.ELF 12808 bytes
SH.ELF 21204 bytes
TEST.ELF 12740 bytes
USR1.ELF 12740 bytes
USR2.ELF 12740 bytes
FAT.ELF 12740 bytes
USR4.ELF 16956 bytes
TEST.TXT 26 bytes
HHOS />
```

7.测试文本编辑器

我将把TEST.TXT的内容从This is a fat32 test file.改为 This is a very very fat file.

首先在Shell下输入文本编辑器程序的完整路径/EDIT.ELF以启动它，然后输入要编辑的文件的路径，就打开了TEST.TXT，将光标移动到字母a下开始修改文件，修改结束后按ESC结束保存文件。

操作过程我进行了录屏，请查看“测试文本编辑器.mp4”

运行截图：

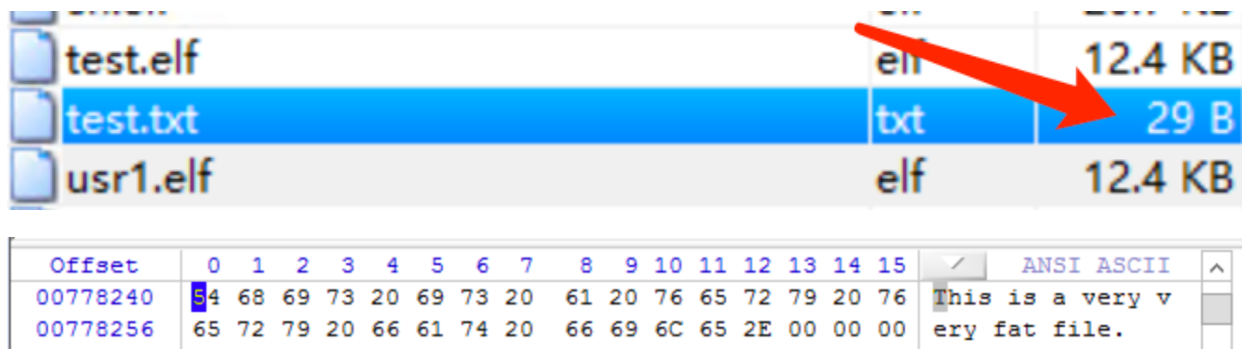

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
HHOS />/EDIT.ELF          OUCHS! OUCH!
-----
Shell forked for user program, pid = 3
-----
file /EDIT.ELF opened
Parent Directory at Cluster: 2
File Start at Cluster: 32
File size (byte): 12808
Please enter the absolute path of file: /TEST.TXT

file /TEST.TXT opened
Parent Directory at Cluster: 2
File Start at Cluster: 234
File size (byte): 26
-----

This is a very very fat file.
-----
Saving file...
Directory / opened
-----
user program exited
-----
HHOS />
```

IPS: 49.411M a: NUM CAPS SCRL HD: 0-M

用WinHex打开修改后的磁盘文件，发现文件内容修改成功，文件大小也正确地更新为了29Bytes



六、实验总结

因为时间关系，本次实验我并没有实现Unix中完整的六层文件系统，而是读取磁盘块后，直接实现FAT32文件操作，然后就实现了文件描述符层，并封装了文件系统系统调用。和真实的Unix系统相比少了事务日志层、文件层、目录层这几层抽象，导致不能把屏幕键盘等IO设备抽象为文件，在操作目录上也很麻烦。这是本次实验的遗憾之处。

FAT32的原理并不复杂，就是一个以根目录为起始的树状结构，使用FAT表记录文件在磁盘上占用的簇的情况。然而具体的实现细节却是十分复杂的，涉及到大量的二进制操作。为了完成本次实验。我读完了两三万单词的《Microsoft Extensible Firmware Initiative FAT32 File System Specification》，即便如此，还是有一些意料之外的要求，比如文件名和扩展名不足8个字符或3个字符，要用空格' '去补全，而不是用'\0'去补全，否则这个文件就不会被识别出来。因为Mac系统上没有Winhex，我得不断重复地把磁盘镜像文件压缩拷贝到我的Windows云服务器上进行调试，过程还是挺费事的。

本次实验应该是本学期我的最后一次操作系统实验了。一学期以来，操作系统可以说是完全融进了我的生活之中，几乎每周，每天都在思索如何改进和完成它。最后的成果，得到的是具有内存管理、进程管理、文件系统，有一个简单的C/C++库，组分基本完整的操作系统，代码量达到了9000行！如果回到学期开始时，一定很难想象自己可以完成这么大的项目。当然它还有很多不足，还有许许多多可以添加和完善的功能，然而这场冒险之旅已经为我拨开了操作系统上的重重迷雾，现在再看那些讲解内核实现的书籍，不再是一头雾水，甚至每个标题都觉得十分亲切。在之后学习Linux 2.6等更现代的内核实现以及Unix系统编程的过程中，这个我的操作系统也是一个绝佳的实验框架，可以把理论上的东西在里面实践起来。

github.com/AlDanial/cloc v 1.76 T=1.15 s (82.0 files/s, 10199.9 lines/s)

Language	files	blank	comment	code
C++	42	713	737	5738
C/C++ Header	36	342	250	1960
Assembly	13	154	124	978
C	1	19	212	185
make	2	76	27	173
SUM:	94	1304	1350	9034

七、参考文献

《Microsoft Extensible Firmware Initiative FAT32 File System Specification》

xv6中文文档-第六章 文件系统 <https://th0ar.gitbooks.io/xv6-chinese/content/content/chapter6.html>