实验四：

FAT文件系统的实现

实验目标

- 熟悉FAT16的存储结构，利用FUSE实现一个只读的FAT文件系统

实验环境

- OS：Ubuntu 14.04 （32位）

实验内容

一、熟悉使用FUSE

1、FUSE概述

- FUSE（Filesystem in Userspace，用户态文件系统）是一个实现在用户空间的文件系统框架，通过FUSE内核模块的支持，使用者只需要根据fuse提供的接口实现具体的文件操作就可以实现一个文件系统。
- FUSE主要由三部分组成：FUSE内核模块、用户空间库libfuse以及挂载工具fusermount：
    1. fuse内核模块：实现了和VFS的对接，实现了一个能被用户空间进程打开的设备。
    2. fuse库libfuse：负责和内核空间通信，接收来自/dev/fuse的请求，并将其转化为一系列的函数调用，将结果写回到/dev/fuse；提供的函数可以对fuse文件系统进行挂载卸载、从linux内核读取请求以及发送响应到内核。
    3. 挂载工具：实现对用户态文件系统的挂载。
- 更多详细内容可参考这个博客。

2、配置FUSE环境

- linux kernel在2.6.14后添加了FUSE模块，因此对于目前的大多数发行版来说只需要安装libfuse库即可。
- 在https://github.com/libfuse/libfuse里下载libfuse 2.9.5，然后编译安装：

```
wget -O libfuse-2.9.5.zip https://codeload.github.com/libfuse/libfuse/zip/fuse_2_9_5
unzip libfuse-2.9.5.zip
cd libfuse-fuse_2_9_5/
./makeconf.sh
./configure --prefix=/usr
make -j4
sudo make install
```

- **编译出错处理：**在执行./makeconf.sh可能会遇到“ libtoolize: not found ”或 “autoreconf: not found”，需要安装libtool和autoconf软件包，ubuntu下可以通过下面的命令安装：

```
sudo apt install libtool
sudo apt install autoconf
```

- **注意**：教学实验室的电脑已配置好环境，可以跳过这一步。

3、测试FUSE

- 通过libfuse-fuse_2_9_5/example下的fusexmp进行测试：

```
cd example
mkdir fuse_test_dir
./fusexmp -d fuse_test_dir
```

- 这时候在文件管理器中打开fuse_test_dir目录，可以看到当前Linux系统的"/"被挂载到这个目录下。
- 结束测试可以直接在当前终端中Ctrl + C结束程序，或者在新的终端中输入：

```
fusermount -u fuse_test_dir
```

- **提示**：当执行用户自己实现的fuse程序时，如果出现如下错误，可通过执行上面这条命令卸载对应的文件夹来解决。

```
emulation@emulation-VirtualBox:~/os-lab4-fs/lab4-code$ ./simple_fat16 -d fat_dir
/
fuse: bad mount point `fat_dir/': Transport endpoint is not connected
emulation@emulation-VirtualBox:~/os-lab4-fs/lab4-code$
```

二、FAT文件系统

1、FAT格式磁盘镜像的制作过程

- 分为三步：创建文件，格式化文件，挂载使用。

```
dd if=/dev/zero of=fat-disk.img bs=1M count=100

mkfs.vfat -F 16 fat-disk.img

mkdir mdir
mount fat-disk.img mdir
```

2、FAT16的存储结构

- FAT16文件系统的基本结构依次为：DBR扇区、FAT表1、FAT表2、根目录和数据区，FAT16格式的磁盘的组织方式如下图所示：

DBR扇区	保留扇区	FAT 1	FAT 2 （重复的）	根目录	数据区	剩余扇区
占1扇区	可能存在	大小取决于实际情况	与FAT 1相同	一般占32个扇区	2号簇开始编号	不足1簇

- **DBR扇区**：DBR是操作系统可以直接访问的第一个扇区,包括一个引导程序和一个称为BPB的本分区参数记录表。BPB参数块记录着本分区的启始扇区、结束扇区、文件存储格式、硬盘介质描述符、根目录大小、*FAT*个数、分配单元的大小等重要参数。下图是一个FAT16文件系统的DBR扇区：

modem | factory | asdf

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	20	01	00	ë< MSDOS5.0
00000010	02	00	02	00	00	F8	15	00	3F	00	FF	00	00	00	00	00	ø ? ÿ
00000020	00	A0	02	00	80	00	29	4E	61	BC	00	4E	4F	20	4E	41	€)Na¼ NO NA
00000030	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	C9	ME FAT16 3É
00000040	8E	D1	BC	F0	7B	8E	D9	B8	00	20	8E	C0	FC	BD	00	7C	ŽÑ¼ð{ŽÙ¸ ŽÀü½ |
00000050	38	4E	24	7D	24	8B	C1	99	E8	3C	01	72	1C	83	EB	3A	8N$}$<Á™è< r ƒë:
00000060	66	A1	1C	7C	26	66	3B	07	26	8A	57	FC	75	06	80	CA	f¡ |&f; &ŠWüu €Ê
00000070	02	88	56	02	80	C3	10	73	EB	33	C9	8A	46	10	98	F7	ˆV €Ã së3ÉŠF ˜÷
00000080	66	16	03	46	1C	13	56	1E	03	46	0E	13	D1	8B	76	11	f F V F Ñ‹v
00000090	60	89	46	FC	89	56	FE	B8	20	00	F7	E6	8B	5E	0B	03	`‰Fü‰Vþ¸ ÷æ‹^

DBR扇区每个字段代表的含义如下：

偏移地址	长度（字节）	含义	数值
0h	3	跳转命令	EB 3C 90
3h	8	OEM NAME	4D 53 44 4F 53 35 2E 30 = MSDOS5.0
Bh	2	每扇区字节数	00 02 = 0x200 = 512bytes
Dh	1	每簇扇区数	20 = 0x20 = 32 扇区
Eh	2	保留扇区数	01 00 = 0x1=1 扇区
10h	1	FAT数量	02 = 0x2 FAT
11h	2	根目录项数	00 02 =0x200=512
16h	2	每个FAT所占扇区数	15 00 = 0x15 = 21个扇区
18h	2	每个道所占扇区数	3F 00 = 0x3F = 63
1Ch	4	隐藏扇区数	0
36h	8	FAT NAME	FAT16
3Eh	448	引导程序执行代码	…
1FEh	2	扇区结束标志	AA 55

根据上面的DBR扇区，我们可以算出各FAT的偏移地址，根目录的偏移地址，数据区的偏移地址。

FAT1偏移地址：保留扇区（FAT1之前的扇区，包括DBR扇区）之后就是FAT1。因此可以得到，FAT1的偏移地址就是1个扇区的位置，也就是512。

FAT2偏移地址：FA1偏移地址+FAT1的大小，512+21*512 = 11264。

根目录偏移地址： FAT2偏移地址+FAT2的大小，11264+21*512= 22016。

数据区的偏移地址：根目录偏移地址+根目录大小，22016+32*512=38400。其中根目录大小是由根目录项数决定的，每项占32字节。

- **FAT表：**FAT是簇的链表，FAT2与FAT1的内容通常是即时同步的，可以认为两个FAT表完全相同。在根据目录项获取文件的首簇号后，在FAT中找到对应的簇，可以找到下一个簇，一直到文件结束。每个簇用2字节表示簇的状态，具体意义如下表所示：

取值	对应簇含义
0x0000	未使用的簇
0x0002-0xFFEF	已分配的簇
0xFFF0-0xFFF6	系统保留
0xFFF8-0xFFFF	文件结束

- **根目录**：FAT文件系统的一个重要思想是把目录当作一个特殊文件来处理，在FAT16中，虽然根目录地位并不等同于普通的目录，但其组织形式和普通的目录并没有不同。不管目录文件所占空间为多少簇，系统都会以32个字节为单位进行目录文件所占簇的分配。每个32字节目录项表示的具体含义如下：

偏移地址	长度	含义
0	8	文件名
8	3	扩展名
B	1	属性
C	10	保留
16	2	文件最近修改时间
18	2	文件最近修改日期
1A	2	文件的首簇号
1C	4	文件长度大小

系统将文件名分成两部分进行存储，即主文件名+扩展名。不记录主文件名与扩展名之间的"."，并且存储成**大写字母**。主文件名不足8个字符以空白符(20H)填充，扩展名不足3个字符同样以空白符(20H)填充。0x0偏移处的取值若为0x0，表明目录项为空；若为0xE5，表明目录项曾被使用，但对应的文件或文件夹已被删除。文件名中的第一个字符若为“.”或“..”表示记录的是一个子目录的目录项，“.”代表当前目录；“..”代表上级目录。

- **FAT表与文件定位**：FAT16文件系统从根目录所占的32个扇区之后的第一个扇区开始以簇为单位进行数据的处理，这之前仍以扇区为单位。对于根目录之后的第一个簇，系统并不编号为0号簇或1号簇，而是编号为2号簇，也就是说数据区顺序上的第1个簇也是编号上的2号簇。

  FAT表以"F8 FF FF FF" 开头，为介质描述单元，并不参与FAT表簇链关系，下图展示了FAT表的内容。为了定位一个文件或目录，需要先找到其对应的目录项，然后根据首簇号查找FAT中的相应表项。

modem | factory | asdf

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00011264	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	øÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011280	09	00	0A	00	0B	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿ
00011296	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011312	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011328	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011344	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011360	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011376	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00011392	41	00	42	00	FF	FF	44	00	45	00	FF	FF	47	00	48	00	A B ÿÿD E ÿÿG H
00011408	FF	FF	4A	00	4B	00	FF	FF	4D	00	4E	00	FF	FF	50	00	ÿÿJ K ÿÿM N ÿÿP
00011424	51	00	FF	FF	53	00	54	00	FF	FF	56	00	57	00	FF	FF	Q ÿÿS T ÿÿV W ÿÿ
00011440	59	00	5A	00	5B	00	FF	FF	5D	00	5E	00	FF	FF	60	00	Y Z [ÿÿ] ^ ÿÿ`
00011456	61	00	62	00	FF	FF	64	00	65	00	66	00	FF	FF	68	00	a b ÿÿd e f ÿÿh

假设一个文件，其首簇号是2，则查看FAT的相对偏移0x4~0x5的内容，此处为0xFFFF，表示存储在2号簇上的文件是个小文件，只占用1个簇便结束了。

假设另一个文件的首簇号是8，则查看FAT的相对偏移0x10-0x11的内容，为0x0009，就是说该文件继8号簇后的内容会存放在9号簇中；继续查看9号簇的FAT表项内容是0x000A，该文件继9簇后的内容会存放在10号簇中，继续这个过程，会发现文件在11号簇结束。

三、实验要求

1、实验任务

- 补全代码包simple_fat16.c中的标记TODO（一共9处）的部分，实现一个只读的FAT16文件系统。
- **提示**：有需要的话，可采用Linux下的xxd和hexdump等命令或者Windows下的WinHex等十六进制文件编辑工具，分析对应的磁盘镜像文件。

2、运行与测试

- 使用如下的命令编译并测试程序：

```
#进入源码目录
make

#测试一
./simple_fat16 --test
```

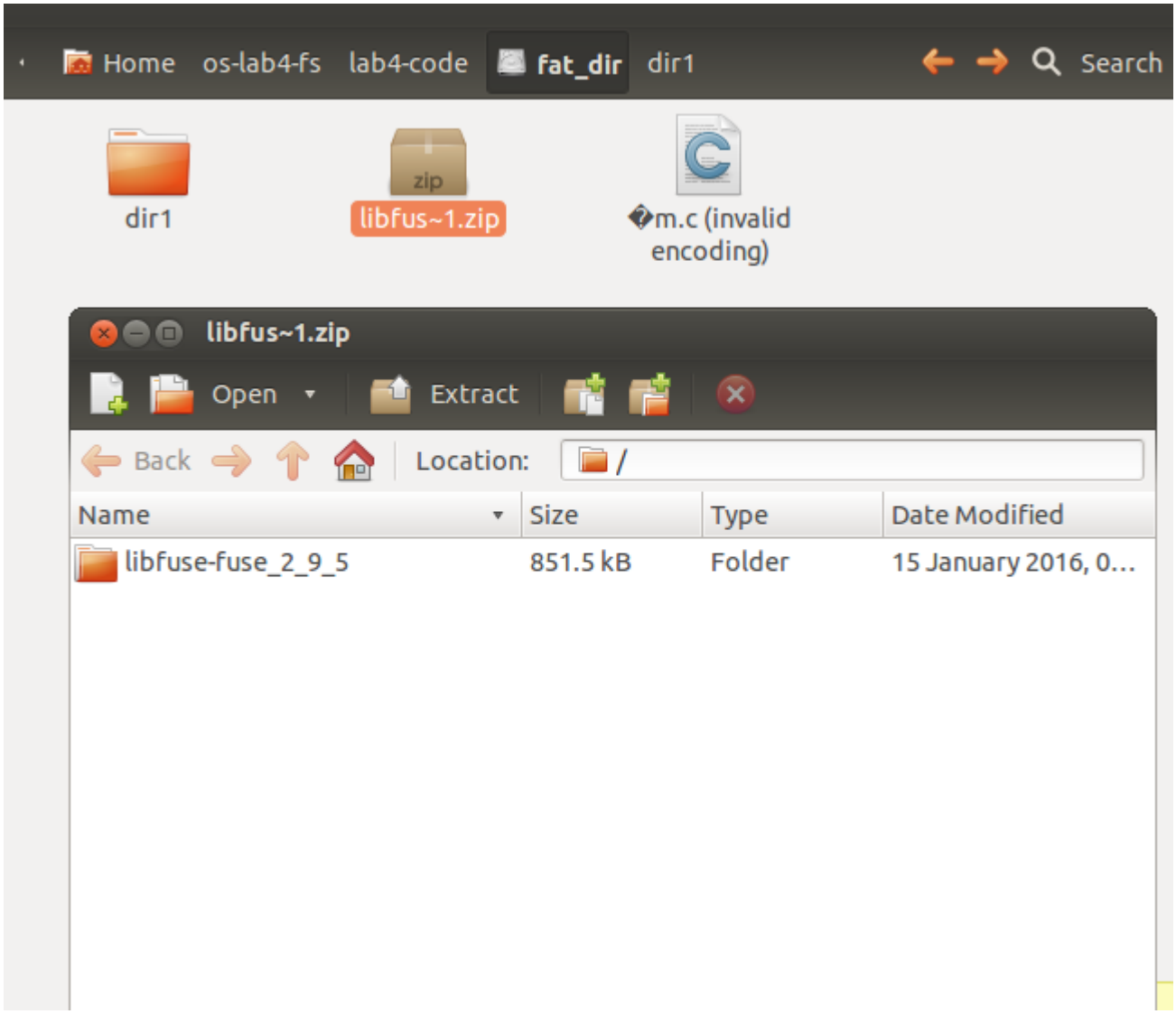
测试一是为了验证程序的FAT相关部分的代码正确性（以fat16_test.img作为磁盘镜像文件），一共有6个测试目标：

```
success in test_path_decode

#3 running test_pre_init_fat16
success in test_pre_init_fat16

#4 running test_fat_entry_by_cluster
test case 1: OK
test case 2: OK
test case 3: OK
success in test_fat_entry_by_cluster

#5 running test_find_root
test case 1: OK
test case 2: OK
test case 3: OK
success in test_find_root

#6 running test_find_subdir
test case 1: OK
test case 2: OK
test case 3: OK
success in test_find_subdir
```

通过测试一后，运行如下的命令进行FUSE功能的测试（以fat16.img作为磁盘镜像文件）：

```
#测试二
./simple_fat16 -d fat_dir
```

这时候在文件管理器中打开fat_dir目录，需要能够看到里面的文件列表，以及正常访问里面的文件：



3、评分标准

- 得分由两部分构成：代码运行得分，回答问题得分。
  - 代码运行得分：未能通过测试一，按照完成的测试目标数给分；能正常通过测试二（看到文件+访问文件）得8分，只能看到文件列表但无法正常访问文件得7分。
  - 回答问题得分是指：针对代码实现，助教提问问题，根据回答情况在代码运行分的基础上加分/扣分。

4、提交方式

- 将实验报告、源代码打包为压缩包提交。
  - 本次实验只要求修改simple_fat16.c文件，不能修改其他文件，所以源代码只提交simple_fat16.c文件即可。
- 提交至邮箱：ustc_os2019@163.com
  - 邮件主题、文件名称、压缩包名称 均采用以下格式命名
    - x-学号-姓名（x：代表第x次实验，本次为第4次实验）
      - 例如张三的第4次实验命名为“4-PB17011010-张三”
    - 未按照规范命名的邮件会被忽略、删除
- 实验验收和报告提交截止日期请关注课程主页。

拓展实验(选做):

实验目标

- 实现一个功能完整的FAT 16文件系统

实验内容

1、支持长文件名的FAT 16文件系统

- 在之前的实验中，我们只关注了短文件名（8+3格式）的实现，这种这种方式有很大的局限性。
- **要求**：参考相关资料，实现能够支持长文件名的FAT文件系统，要求能够正常显示fat.img中的文件名。

2、支持写操作的FAT 16文件系统

- 在已有的只读文件基础上，实现可支持创建文件，写文件，删除文件的功能，同时能够支持文件属性的修改。
- **要求**：能够在文件系统中支持touch，mkdir，cp，rm等命令，支持的功能越多越好。

3、实现线程安全的FAT 16文件系统

- **要求**：在多线程访问文件系统时，需要保证读写请求被正确处理，能够使用fio工具测试文件系统，并且有不错的性能结果。