Lab 3 添加一个加密文件系统

1 实验目的

文件系统是操作系统中最直观的部分，因为用户可以通过文件直接地和操作系统交互，操作系统也必须为用户提供数据计算、数据存储的功能。本实验通过添加一个文件系统，进一步理解 Linux 中的文件系统原理及其实现。

- 深入理解操作系统文件系统原理
- 学习理解 Linux 的 VFS 文件系统管理技术
- 学习理解 Linux 的 ext2 文件系统实现技术
- 设计和实现加密文件系统

2 实验内容

本实验要添加一个类似于 ext2 的自定义文件系统 myext2。

对 myext2 文件系统的描述如下：

1. myext2 文件系统的物理格式定义与 ext2 基本一致，但 myext2 的 magic number 是 `0x6666`，而 ext2 的 magic number 是 `0xEF53`
2. myext2 是 ext2 的定制版本，它不但支持原来 ext2 文件系统的部分操作，还添加了用加密数据进行读写的操作

实验主要内容：

- 添加一个类似 ext2 的文件系统 myext2
- 修改 myext2 的 magic number
- 添加文件系统创建工具
- 添加加密文件系统操作，包括 `read_crypt`，`write_crypt`，使其增加对加密数据的读写

3 实验环境

Linux 版本：Ubuntu 14.04

Linux 内核版本：linux-3.13.0

4 实验过程

4.1 添加 myext2 文件系统

4.1.1 源代码复制

在 Linux 的 shell 下，执行如下操作：

```
# 进入内核源码目录
cd fs
cp -R ext2 myext2
cd myext2
mv ext2.h myext2.h

cd /lib/modules/$(uname -r)/build /include/linux
cp ext2_fs.h myext2_fs.h
cd /lib/modules/$(uname -r)/build /include/asm-generic/bitops
cp ext2-atomic.h myext2-atomic.h
cp ext2-atomic-setbit.h myext2-atomic-setbit.h
```

这样就完成了克隆文件系统工作的第一步——源代码复制。对于克隆文件系统来说，这样当然还远远不够，因为文件里面的数据结构名、函数名、以及相关的宏等内容还没有根据 myext2 改掉，连编译都通不过。

4.1.2 修改复制文件的内容

为了简单起见，做一个最简单的替换：

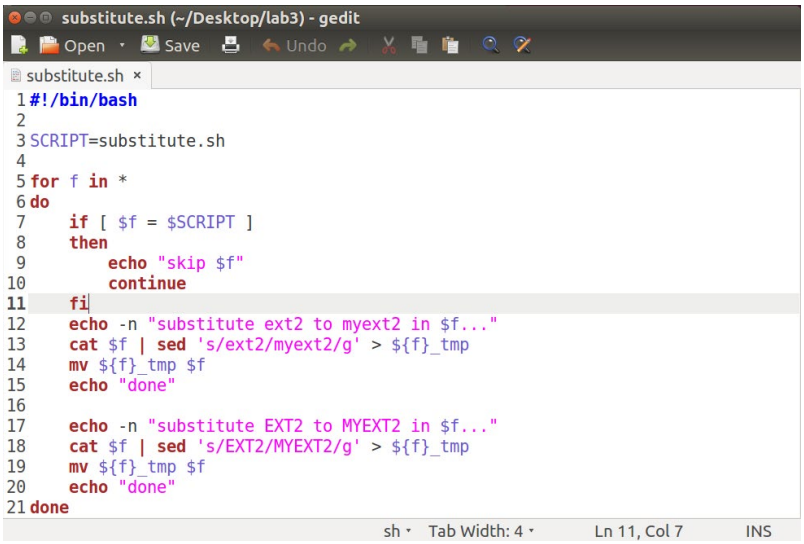
- 将 EXT2 替换成 MYEXT2
- 将 ext2 替换成 myext2

对于 fs/myext2 下面文件中字符串的替换，使用下面的脚本：

```
#!/bin/bash
SCRIPT=substitute.sh
for f in *
do
```

```
    if [ $f = $SCRIPT ]
    then
        echo "skip $f"
        continue
    fi
    echo -n "substitute ext2 to myext2 in $f..."
    cat $f | sed 's/ext2/myext2/g' > ${f}_tmp
    mv ${f}_tmp $f
    echo "done"
    echo -n "substitute EXT2 to MYEXT2 in $f..."
    cat $f | sed 's/EXT2/MYEXT2/g' > ${f}_tmp
    mv ${f}_tmp $f
    echo "done"
done
```

将脚本命名为 substitute.sh



将脚本放到 myext2 文件夹下，用 chmod 命令使脚本可以执行

```
oslab@oslab-ubuntu-vm:myext2$ chmod 777 substitute.sh
oslab@oslab-ubuntu-vm:myext2$ ls -l
total 332
-rw-rw-r-- 1 oslab oslab  9609 12月   1 22:04 acl.c
-rw-rw-r-- 1 oslab oslab  1499 12月   1 22:04 acl.h
-rw-rw-r-- 1 oslab oslab 45517 12月   1 22:04 balloc.c
-rw-rw-r-- 1 oslab oslab 18119 12月   1 22:04 dir.c
-rw-rw-r-- 1 oslab oslab  2829 12月   1 22:04 file.c
-rw-rw-r-- 1 oslab oslab 18690 12月   1 22:04 ialloc.c
-rw-rw-r-- 1 oslab oslab 45910 12月   1 22:04 inode.c
-rw-rw-r-- 1 oslab oslab  4487 12月   1 22:04 ioctl.c
-rw-rw-r-- 1 oslab oslab  1736 12月   1 22:04 Kconfig
-rw-rw-r-- 1 oslab oslab   380 12月   1 22:04 Makefile
-rw-rw-r-- 1 oslab oslab    61 12月   1 22:04 modules.order
-rw-rw-r-- 1 oslab oslab     0 12月   1 22:04 Module.symvers
-rw-rw-r-- 1 oslab oslab 28131 12月   1 22:04 myext2.h
-rw-rw-r-- 1 oslab oslab 10120 12月   1 22:04 myext2.mod.c
-rw-rw-r-- 1 oslab oslab 10188 12月   1 22:04 namei.c
-rwxrwxrwx 1 oslab oslab   398 12月   1 22:14 substitute.sh
-rw-rw-r-- 1 oslab oslab 43041 12月   1 22:04 super.c
-rw-rw-r-- 1 oslab oslab  1344 12月   1 22:04 symlink.c
```

运行脚本

```
oslab@oslab-ubuntu-vm:myext2$ ./substitute.sh
substitute ext2 to myext2 in acl.c...done
substitute EXT2 to MYEXT2 in acl.c...done
substitute ext2 to myext2 in acl.h...done
substitute EXT2 to MYEXT2 in acl.h...done
substitute ext2 to myext2 in balloc.c...done
substitute EXT2 to MYEXT2 in balloc.c...done
substitute ext2 to myext2 in dir.c...done
substitute EXT2 to MYEXT2 in dir.c...done
substitute ext2 to myext2 in file.c...done
substitute EXT2 to MYEXT2 in file.c...done
substitute ext2 to myext2 in ialloc.c...done
substitute EXT2 to MYEXT2 in ialloc.c...done
substitute ext2 to myext2 in inode.c...done
substitute EXT2 to MYEXT2 in inode.c...done
substitute ext2 to myext2 in ioctl.c...done
substitute EXT2 to MYEXT2 in ioctl.c...done
substitute ext2 to myext2 in Kconfig...done
substitute EXT2 to MYEXT2 in Kconfig...done
substitute ext2 to myext2 in Makefile...done
substitute EXT2 to MYEXT2 in Makefile...done
substitute ext2 to myext2 in modules.order...done
substitute EXT2 to MYEXT2 in modules.order...done
substitute ext2 to myext2 in Module.symvers...done
substitute EXT2 to MYEXT2 in Module.symvers...done
substitute ext2 to myext2 in myext2.h...done
substitute EXT2 to MYEXT2 in myext2.h...done
substitute ext2 to myext2 in myext2.mod.c...done
substitute EXT2 to MYEXT2 in myext2.mod.c...done
substitute ext2 to myext2 in namei.c...done
```

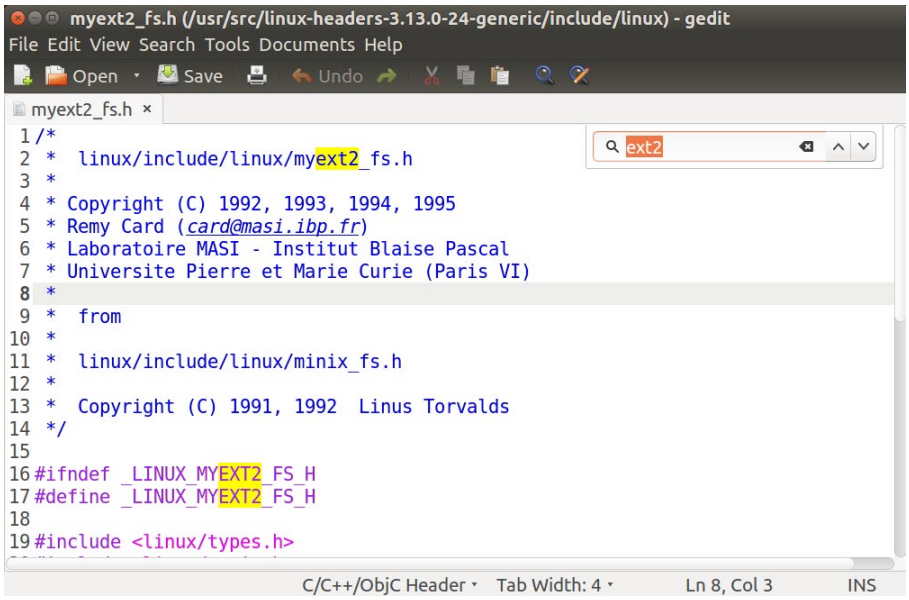
```
substitute ext2 to myext2 in modules.order...done
substitute EXT2 to MYEXT2 in modules.order...done
substitute ext2 to myext2 in Module.symvers...done
substitute EXT2 to MYEXT2 in Module.symvers...done
substitute ext2 to myext2 in myext2.h...done
substitute EXT2 to MYEXT2 in myext2.h...done
substitute ext2 to myext2 in myext2.mod.c...done
substitute EXT2 to MYEXT2 in myext2.mod.c...done
substitute ext2 to myext2 in namei.c...done
substitute EXT2 to MYEXT2 in namei.c...done
skip substitute.sh
substitute ext2 to myext2 in super.c...done
substitute EXT2 to MYEXT2 in super.c...done
substitute ext2 to myext2 in symlink.c...done
substitute EXT2 to MYEXT2 in symlink.c...done
substitute ext2 to myext2 in xattr.c...done
substitute EXT2 to MYEXT2 in xattr.c...done
substitute ext2 to myext2 in xattr.h...done
substitute EXT2 to MYEXT2 in xattr.h...done
substitute ext2 to myext2 in xattr_security.c...done
substitute EXT2 to MYEXT2 in xattr_security.c...done
substitute ext2 to myext2 in xattr_trusted.c...done
substitute EXT2 to MYEXT2 in xattr_trusted.c...done
substitute ext2 to myext2 in xattr_user.c...done
substitute EXT2 to MYEXT2 in xattr_user.c...done
substitute ext2 to myext2 in xip.c...done
substitute EXT2 to MYEXT2 in xip.c...done
substitute ext2 to myext2 in xip.h...done
substitute EXT2 to MYEXT2 in xip.h...done
```

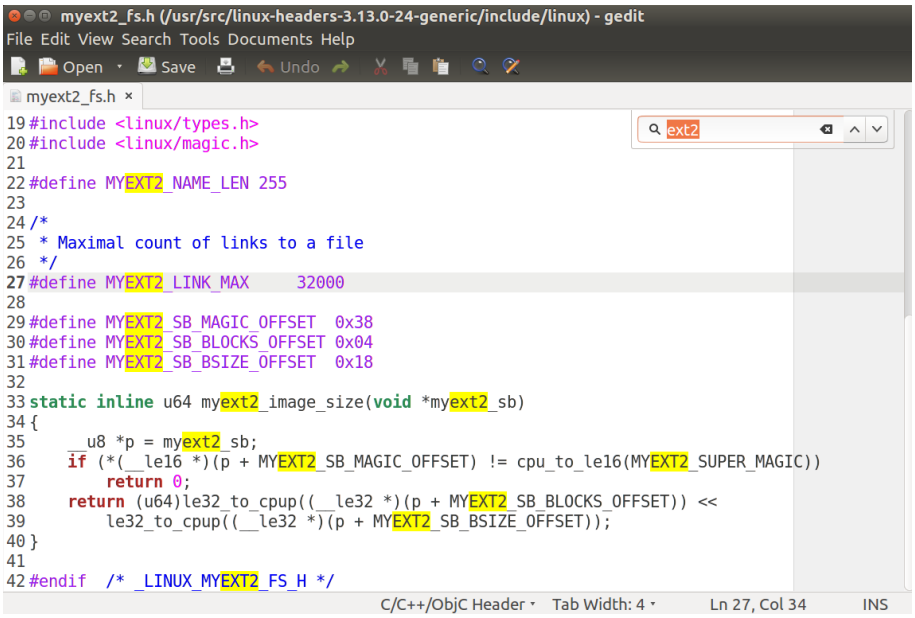
对系统库中的文件内容进行修改，同样将 ext2 改为 myext2，将 EXT2 改为 MYEXT2

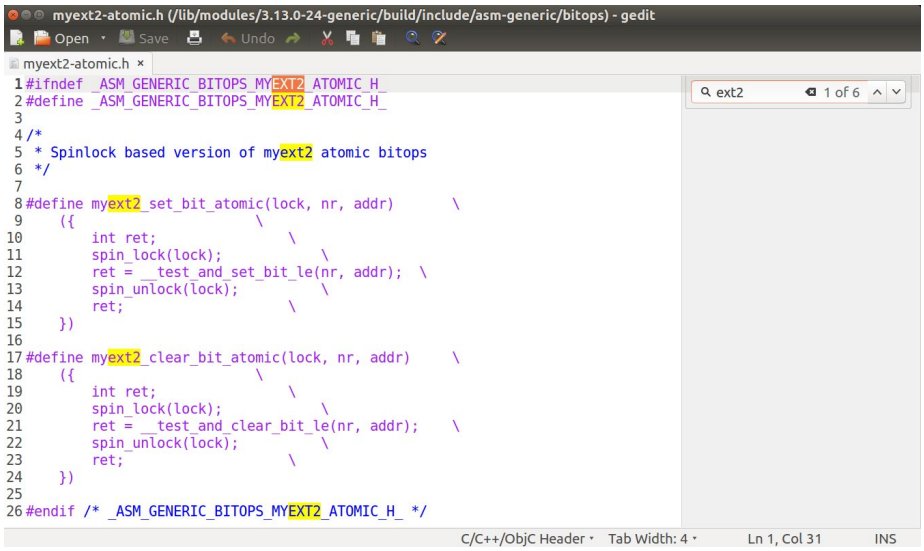
要进行替换的文件有：

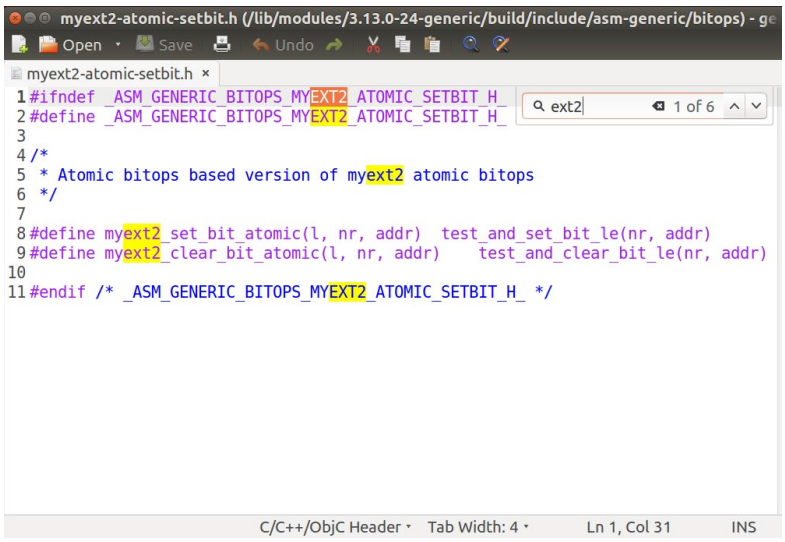
`/lib/modules/$(uname -r)/build/include/linux/myext2_fs.h`

`/lib/modules/$(uname -r)/build/include/asm-generic/bitops/`
下的 `myext2-atomic.h` 与 `myext2-atomic-setbit.h`

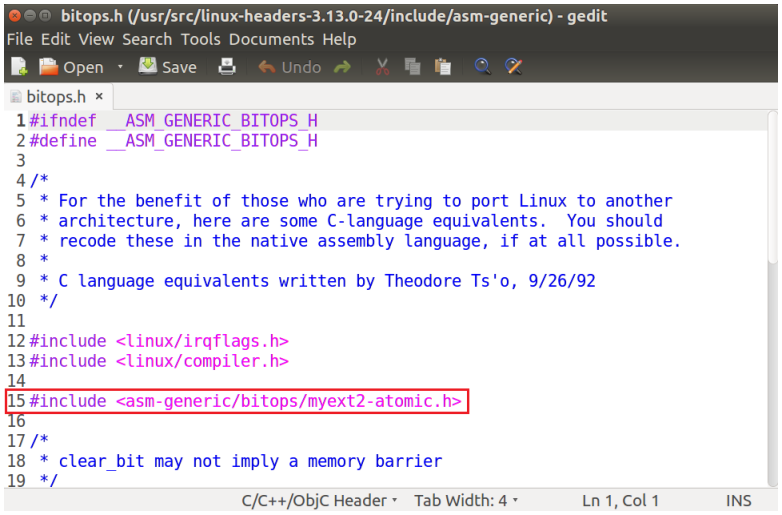

myext2_fs.h (/usr/src/linux-headers-3.13.0-24-generic/include/linux) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
myext2_fs.h
ext2
1 /*
2 * linux/include/linux/myext2_fs.h
3 *
4 * Copyright (C) 1992, 1993, 1994, 1995
5 * Remy Card (card@masi.ibp.fr)
6 * Laboratoire MASI - Institut Blaise Pascal
7 * Universite Pierre et Marie Curie (Paris VI)
8 *
9 * from
10 *
11 * linux/include/linux/minix_fs.h
12 *
13 * Copyright (C) 1991, 1992 Linus Torvalds
14 */
15
16 #ifndef _LINUX_MYEXT2_FS_H
17 #define _LINUX_MYEXT2_FS_H
18
19 #include <linux/types.h>
C/C++/ObjC Header Tab Width: 4 Ln 8, Col 3 INS


myext2_fs.h (/usr/src/linux-headers-3.13.0-24-generic/include/linux) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
myext2_fs.h
ext2
19 #include <linux/types.h>
20 #include <linux/magic.h>
21
22 #define MYEXT2_NAME_LEN 255
23
24 /*
25 * Maximal count of links to a file
26 */
27 #define MYEXT2_LINK_MAX 32000
28
29 #define MYEXT2_SB_MAGIC_OFFSET 0x38
30 #define MYEXT2_SB_BLOCKS_OFFSET 0x04
31 #define MYEXT2_SB_BSIZE_OFFSET 0x18
32
33 static inline u64 myext2_image_size(void *myext2_sb)
34 {
35 __u8 *p = myext2_sb;
36 if (*(__le16 *)(p + MYEXT2_SB_MAGIC_OFFSET) != cpu_to_le16(MYEXT2_SUPER_MAGIC))
37 return 0;
38 return (u64)le32_to_cpup((__le32 *)(p + MYEXT2_SB_BLOCKS_OFFSET)) <<
39 le32_to_cpup((__le32 *)(p + MYEXT2_SB_BSIZE_OFFSET));
40 }
41
42 #endif /* _LINUX_MYEXT2_FS_H */
C/C++/ObjC Header Tab Width: 4 Ln 27, Col 34 INS

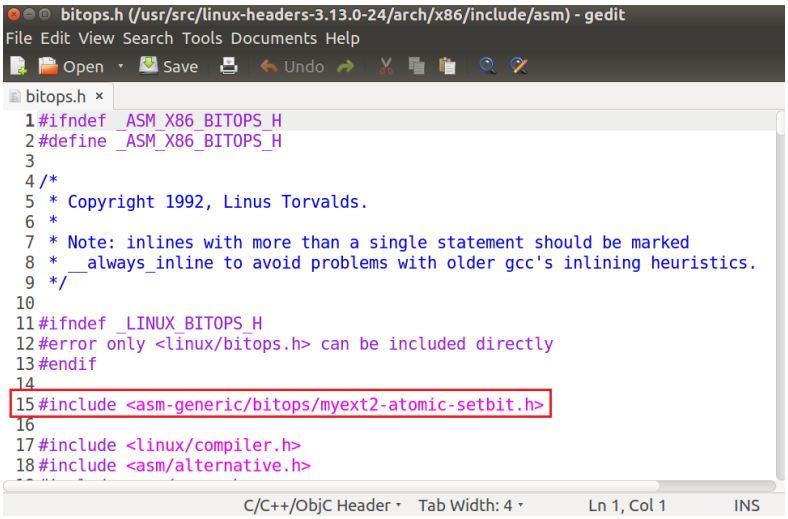

myext2-atomic.h (/lib/modules/3.13.0-24-generic/build/include/asm-generic/bitops) - gedit
Open Save Undo
myext2-atomic.h
ext2
1 of 6
1 #ifndef _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_
2 #define _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_
3
4 /*
5 * Spinlock based version of myext2 atomic bitops
6 */
7
8 #define myext2_set_bit_atomic(lock, nr, addr) \
9 ({ \
10 int ret; \
11 spin_lock(lock); \
12 ret = __test_and_set_bit_le(nr, addr); \
13 spin_unlock(lock); \
14 ret; \
15 })
16
17 #define myext2_clear_bit_atomic(lock, nr, addr) \
18 ({ \
19 int ret; \
20 spin_lock(lock); \
21 ret = __test_and_clear_bit_le(nr, addr); \
22 spin_unlock(lock); \
23 ret; \
24 })
25
26 #endif /* _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_ */
C/C++/ObjC Header Tab Width: 4 Ln 1, Col 31 INS



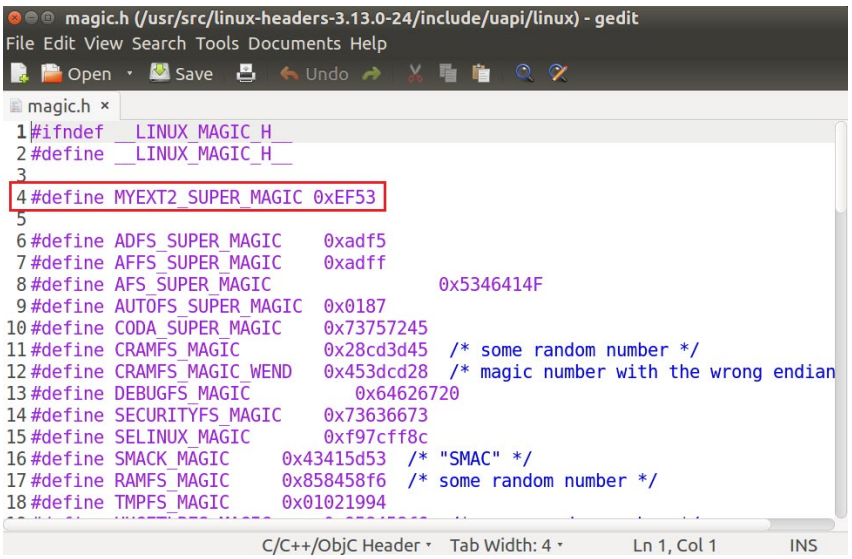
在/lib/modules/$(uname -r)/build/include/asm-generic/bitops.h 文件中添加：#include <asm-generic/bitops/myext2-atomic.h>



在/lib/modules/$(uname -r)/build/arch/x86/include/asm/bitops.h 文件中添加：#include <asm-generic/bitops/myext2-atomic-setbit.h>



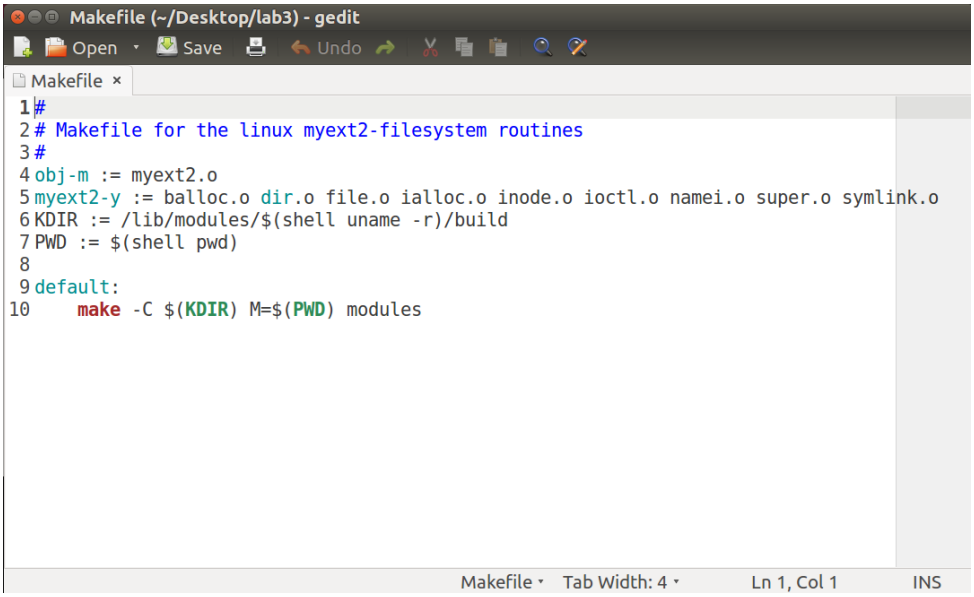
在/lib/modules/$(uname -r)/build/include/uapi/linux/magic.h 文件中添加：#define MYEXT2_SUPER_MAGIC 0xEF53



4.1.3 将 myext2 编译为内核模块

修改 myext2 文件夹下的 Makefile 文件为：

```
# Makefile for the linux myext2-filesystem routines
obj-m := myext2.o
myext2-y := balloc.o dir.o file.o ialloc.o inode.o ioctl.o namei.o \
super.o symlink.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
    make -C $(KDIR) M=$(PWD) modules
```



在 myext2 文件夹下执行 make 命令

```
oslab@oslab-ubuntu-vm:myext2$ make
make -C /lib/modules/3.13.0-24-generic/build M=/home/oslab/Desktop/lab
3/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-24-generic'
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/file.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/namei.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/super.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-24-generic'
```

安装 myext2 模块

```
oslab@oslab-ubuntu-vm:myext2$ sudo insmod myext2.ko
[sudo] password for oslab:
oslab@oslab-ubuntu-vm:myext2$ cat /proc/filesystems | grep myext2
        myext2
```

可以看到 myext2 文件系统已经成功加载

找一个目录进行测试，先造一个全是 0 的磁盘块

```
oslab@oslab-ubuntu-vm:test$ dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.0021191 s, 495 MB/s
oslab@oslab-ubuntu-vm:test$ /sbin/mkfs.ext2 myfs
mke2fs 1.42.9 (4-Feb-2014)
myfs is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
Maximum filesystem blocks=1048576
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

尝试将 myext2 应用于文件块

```
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep /mnt
/home/oslab/Desktop/lab3/test/myfs on /mnt type myext2 (rw)
oslab@oslab-ubuntu-vm:test$ sudo umount /mnt
oslab@oslab-ubuntu-vm:test$ sudo mount -t ext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep /mnt
/home/oslab/Desktop/lab3/test/myfs on /mnt type ext2 (rw)
oslab@oslab-ubuntu-vm:test$ sudo umount /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep /mnt
oslab@oslab-ubuntu-vm:test$
```

可以看到，myext2 以及可以和 ext2 一样地应用于磁盘块

4.2 修改 myext2 的 magic number

修改/lib/modules/$(uname -r)/build/include/uapi/linux/magic.h 文件

```
- #define MYEXT2_SUPER_MAGIC  0xEF53
+ #define MYEXT2_SUPER_MAGIC  0x6666
```

重新用 make 命令编译模块

```
oslab@oslab-ubuntu-vm:myext2$ make
make -C /lib/modules/3.13.0-24-generic/build M=/home/oslab/Desktop/lab
3/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-24-generic'
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/file.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/namei.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/super.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-24-generic'
oslab@oslab-ubuntu-vm:myext2$ sudo insmod myext2.ko
[sudo] password for oslab:
oslab@oslab-ubuntu-vm:myext2$
```

写程序 changeMN.c 来修改创建文件系统的 magic number

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]){
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];
    char infile[100] = {0};
    char outfile[100] = {0};

    if(argc < 2){
        printf("no arg, default source file: myfs\n");
        printf("no arg, default target file: myfs.new\n");
        strcpy(infile, "./myfs");
        strcpy(outfile, "./myfs.new");
    }else if(argc < 3){
        printf("no arg:target, default target file: myfs.new\n");
        strcpy(infile, argv[1]);
        strcpy(outfile, "./myfs.new");
```

```
    }else{
        strcpy(infile, argv[1]);
        strcpy(outfile, argv[2]);
    }

    fp_read = fopen(infile, "rb");
    if(fp_read == NULL){
        printf("open %s failed!\n", infile);
        return 1;
    }
    printf("open %s success!\n", infile);

    fp_write = fopen(outfile, "wb");
    if(fp_write == NULL){
        printf("open %s failed!\n", outfile);
        return 2;
    }
    printf("open %s success!\n", outfile);

    ret = fread(buf, sizeof(unsigned char), 2048, fp_read);
    printf("previous magic number is 0x%x%x\n", buf[0x439],
buf[0x438]);

    buf[0x438] = 0x66;
    buf[0x439] = 0x66;

    fwrite(buf, sizeof(unsigned char), 2048, fp_write);
    printf("current magic number is 0x%x%x\n", buf[0x439],
buf[0x438]);

    while(ret == 2048){
        ret = fread(buf, sizeof(unsigned char), 2048, fp_read);
        fwrite(buf, sizeof(unsigned char), 2048, fp_write);
    }

    if(ret < 2048 && feof(fp_read)){
        printf("change magic number ok!\n");
    }

    fclose(fp_write);
    fclose(fp_read);

    return 0;
}
```

*changeMN.c ×

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]){
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];
    char infile[100] = {0};
    char outfile[100] = {0};

    if(argc < 2){
        printf("no arg, default source file: myfs\n");
        printf("no arg, default target file: myfs.new\n");
        strcpy(infile, "./myfs");
        strcpy(outfile, "./myfs.new");
    }else if(argc < 3){
        printf("no arg:target, default target file: myfs.new\n");
        strcpy(infile, argv[1]);
        strcpy(outfile, "./myfs.new");
    }else{
        strcpy(infile, argv[1]);
        strcpy(outfile, argv[2]);
    }

    fp_read = fopen(infile, "rb");
    if(fp_read == NULL){
        printf("open %s failed!\n", infile);
        return 1;
    }
```

C ▾ Tab Width: 4 ▾ Ln 43, Col 24 INS

*changeMN.c ×

```c
    }
    printf("open %s success!\n", infile);

    fp_write = fopen(outfile, "wb");
    if(fp_write == NULL){
        printf("open %s failed!\n", outfile);
        return 2;
    }
    printf("open %s success!\n", outfile);

    ret = fread(buf, sizeof(unsigned char), 2048, fp_read);
    printf("previous magic number is 0x%x%x\n", buf[0x439], buf[0x438]);

    buf[0x438] = 0x66; buf[0x439] = 0x66;

    fwrite(buf, sizeof(unsigned char), 2048, fp_write);
    printf("current magic number is 0x%x%x\n", buf[0x439], buf[0x438]);

    while(ret == 2048){
        ret = fread(buf, sizeof(unsigned char), 2048, fp_read);
        fwrite(buf, sizeof(unsigned char), 2048, fp_write);
    }

    if(ret < 2048 && feof(fp_read)){printf("change magic number ok!\n");}

    fclose(fp_write);
    fclose(fp_read);

    return 0;
}
```

C ▾ Tab Width: 4 ▾ Ln 43, Col 24 INS

编译程序

```
oslab@oslab-ubuntu-vm:test$ ls
changeMN.c  myfs
oslab@oslab-ubuntu-vm:test$ gcc changeMN.c
oslab@oslab-ubuntu-vm:test$ ls
a.out  changeMN.c  myfs
```

先用 dd 命令拷贝一个全 0 磁盘块，用于测试

再将 ext2 应用于磁盘块，然后用 myext2 格式 mount

然后用 changeMN 修改磁盘块的 magic number

取消 mount，

然后尝试用 ext2 格式 mount

```
oslab@oslab-ubuntu-vm:test$ dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.0543315 s, 19.3 MB/s
oslab@oslab-ubuntu-vm:test$ /sbin/mkfs.ext2 myfs
mke2fs 1.42.9 (4-Feb-2014)
myfs is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
Maximum filesystem blocks=1048576
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

```
oslab@oslab-ubuntu-vm:test$ ls
changeMN  changeMN.c  myfs
oslab@oslab-ubuntu-vm:test$ ./changeMN myfs myfs.new
open myfs success!
open myfs.new success!
previous magic number is 0xef53
current magic number is 0x6666
change magic number ok!
oslab@oslab-ubuntu-vm:test$ ls
changeMN  changeMN.c  myfs  myfs.new
oslab@oslab-ubuntu-vm:test$
```

```
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs.new /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep /mnt
/home/oslab/Desktop/lab3/test/myfs.new on /mnt type myext2 (rw)
oslab@oslab-ubuntu-vm:test$ sudo umount /mnt
oslab@oslab-ubuntu-vm:test$ sudo mount -t ext2 -o loop ./myfs.new /mnt
mount: wrong fs type, bad option, bad superblock on /dev/loop0,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so

oslab@oslab-ubuntu-vm:test$
```

发现用 changeMN 修改 magic number 后已经不能用 ext2 格式来 mount

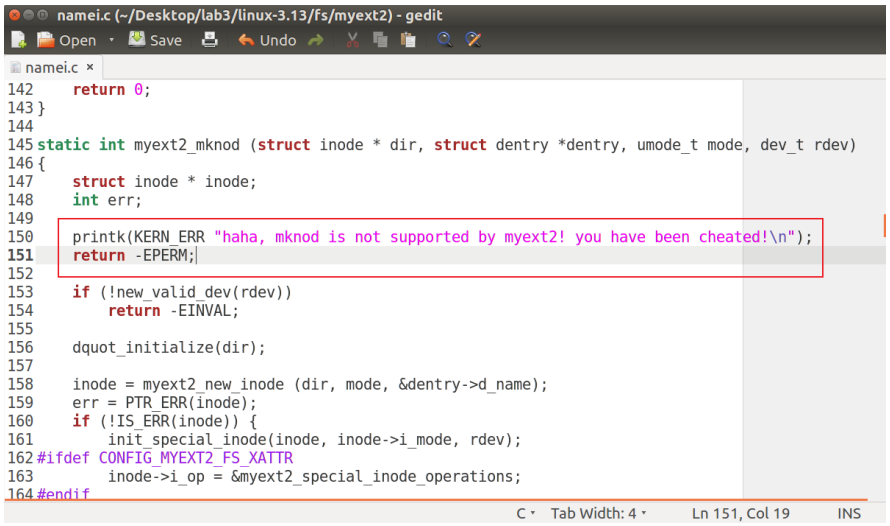
说明 myext2 的 magic number 已经修改成功

4.3 修改文件系统操作

myext2 只是一个实验性质的文件系统，它只要能支持简单的文件操作即可。现在已经完成了 myext2 的总体框架，可以来修改掉 myext2 支持的一些操作，来加深对操作系统对文件系统的操作的理解。

下面裁减 myext2 的 mknod 操作

修改内核源码中的 `fs/ext2/namei.c` 文件，让 `myext2_mknod` 函数输出错误信息后直接返回 `EPERM`，即告诉 shell 在 myext2 文件系统中 mknod 不受支持



重新编译内核模块，并重新安装

```
oslab@oslab-ubuntu-vm:myext2$ make
make -C /lib/modules/3.13.0-24-generic/build M=/home/oslab/Desktop/lab
3/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-24-generic'
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/namei.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-24-generic'
```

试着使用 mknod 命令

```
oslab@oslab-ubuntu-vm:test$ ls
a.out  changeMN.c  myfs  myfs.new
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs.new /mnt
oslab@oslab-ubuntu-vm:test$ cd /mnt
oslab@oslab-ubuntu-vm:mnt$ sudo mknod myfifo p
mknod: ‘myfifo’: Operation not permitted
oslab@oslab-ubuntu-vm:mnt$ dmesg | grep haha
[  253.901373] haha, mknod is not supported by myext2! you have been cheated!
oslab@oslab-ubuntu-vm:mnt$
```

发现提示 mknod 命令不被允许，而且 dmesg 查看 log 发现输出了预期的错误信息

说明之前对 myext2 中 mknod 操作的修改起到了作用

4.4 添加文件系统创建工具

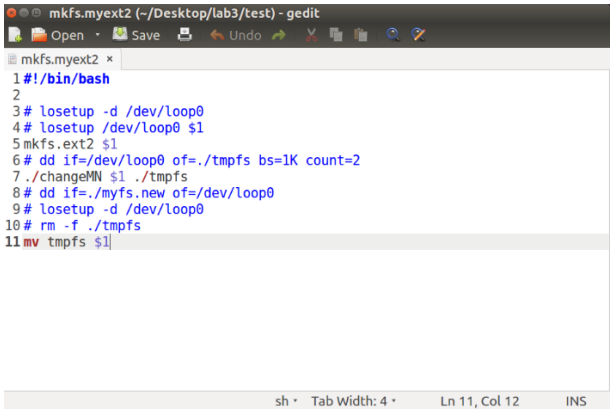
文件系统的创建对于一个文件系统来说是首要的。因为如果不存在一个文件系统，所有对它的操作都是空操作，也是无用的操作。

下面创建类似于 mkfs.ext2 的 myext2 文件系统的创建工具，mkfs.myext2

首先需要确定的是该程序的输入和输出，为了灵活和方便起见，输入为一个文件，这个文件的大小，就是 myext2 文件系统的大小，输出就是带了 myext2 文件系统的文件。

mkfs.myext2 源程序

```
#!/bin/bash
mkfs.ext2 $1
./changeMN $1 ./tmpfs
mv tmpfs $1
```



新创建一个空磁盘块

```
oslab@oslab-ubuntu-vm:test$ dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.0434649 s, 24.1 MB/s
```

尝试用 mkfs.myext2 为磁盘块创建文件系统

```
oslab@oslab-ubuntu-vm:test$ ./mkfs.myext2 ./myfs
mke2fs 1.42.9 (4-Feb-2014)
./myfs is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
Maximum filesystem blocks=1048576
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

open ./myfs success!
open ./tmpfs success!
previous magic number is 0xef53
current magic number is 0x6666
change magic number ok!
oslab@oslab-ubuntu-vm:test$ ls
changeMN  changeMN.c  mkfs.myext2  myfs
```

试着用 myext2 去 mount

```
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep myext2
/home/oslab/Desktop/lab3/test/myfs on /mnt type myext2 (rw)
oslab@oslab-ubuntu-vm:test$
```

可以看到已经可以用 myext2 去 mount 用 mkfs.myext2 创建了文件系统的磁盘块

说明 mkfs.myext2 发挥了正确的作用

4.5 修改加密文件系统的 read 和 write 操作

修改 file.c 的代码，添加两个函数 `new_sync_read_crypt` 和 `new_sync_write_crypt`

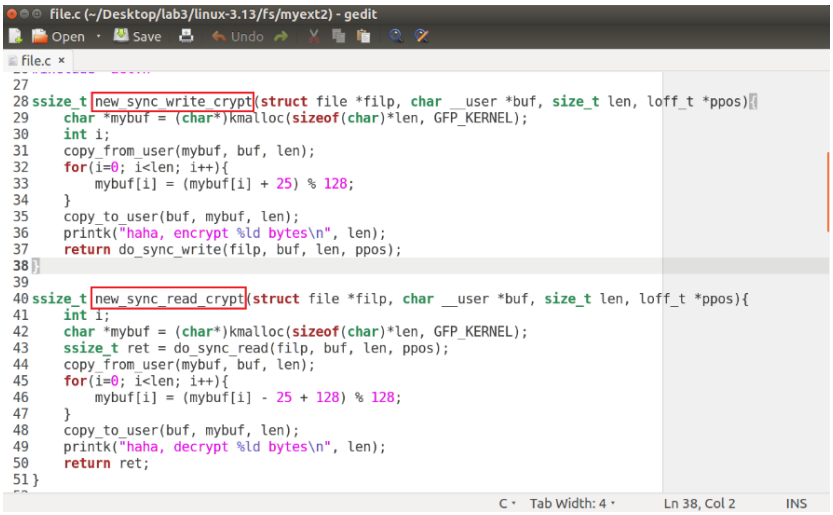
`new_sync_write_crypt` 会在写入前将每个字节向后移 25 位(字符值+25)

`new_sync_read_crypt` 会在写入前将每个字节向前移 25 位(字符值-25)

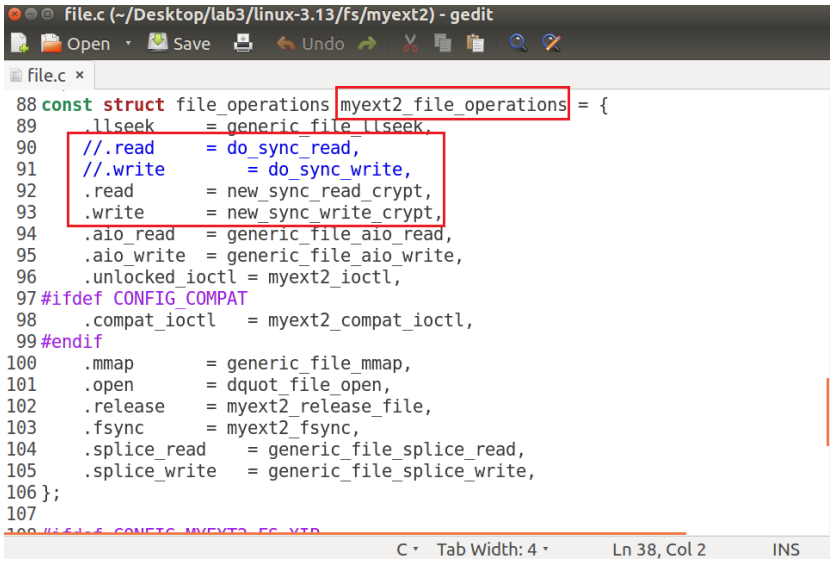
两个函数的源码：

```
ssize_t new_sync_write_crypt(struct file *filp, char __user *buf,
size_t len, loff_t *ppos){
    char *mybuf = (char*)kmalloc(sizeof(char)*len, GFP_KERNEL);
    int i;
    copy_from_user(mybuf, buf, len);
    for(i=0; i<len; i++){
        mybuf[i] = (mybuf[i] + 25) % 128;
    }
    copy_to_user(buf, mybuf, len);
    printk("haha, encrypt %ld bytes\n", len);
    return do_sync_write(filp, buf, len, ppos);
}

ssize_t new_sync_read_crypt(struct file *filp, char __user *buf,
size_t len, loff_t *ppos){
    int i;
    char *mybuf = (char*)kmalloc(sizeof(char)*len, GFP_KERNEL);
    ssize_t ret = do_sync_read(filp, buf, len, ppos);
    copy_from_user(mybuf, buf, len);
    for(i=0; i<len; i++){
        mybuf[i] = (mybuf[i] - 25 + 128) % 128;
    }
    copy_to_user(buf, mybuf, len);
    printk("haha, decrypt %ld bytes\n", len);
    return ret;
}
```



将这两个函数指针赋给 `myext2_file_operations` 结构中的 `read` 和 `write` 操作



重新编译 myext2 模块

```
oslab@oslab-ubuntu-vm:myext2$ make clean
rm *.o *.ko
oslab@oslab-ubuntu-vm:myext2$ make
make -C /lib/modules/3.13.0-24-generic/build M=/home/oslab/Desktop/lab3/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-24-generic'
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/file.o
/home/oslab/Desktop/lab3/linux-3.13/fs/myext2/file.c:93:2: warning: initialization from incompatible pointer type [en
abled by default]
  .write  = new_sync_write_crypt,
  ^
/home/oslab/Desktop/lab3/linux-3.13/fs/myext2/file.c:93:2: warning: (near initialization for ‘myext2_file_operations.
write’) [enabled by default]
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/namei.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/super.o
  CC [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/oslab/Desktop/lab3/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-24-generic'
oslab@oslab-ubuntu-vm:myext2$
```

用 myext2 去 mount 一个创建了 myext2 文件系统的磁盘块

在 mnt 目录下创建文件 test，写入 123456，并尝试读取

然后将 test 文件复制到其他目录，再次尝试读取

```
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ mount | grep myext2
/home/oslab/Desktop/lab3/test/myfs on /mnt type myext2 (rw)
oslab@oslab-ubuntu-vm:test$ cd /mnt
oslab@oslab-ubuntu-vm:mnt$ ls
lost+found
oslab@oslab-ubuntu-vm:mnt$ sudo vim test
oslab@oslab-ubuntu-vm:mnt$ cat test
123456
oslab@oslab-ubuntu-vm:mnt$ cp test ~/Desktop/lab3/test/
oslab@oslab-ubuntu-vm:mnt$ cat ~/Desktop/lab3/test/test
123456
oslab@oslab-ubuntu-vm:mnt$
```

可以看到均可以正常显示文件内容

用 dmesg 查看 log

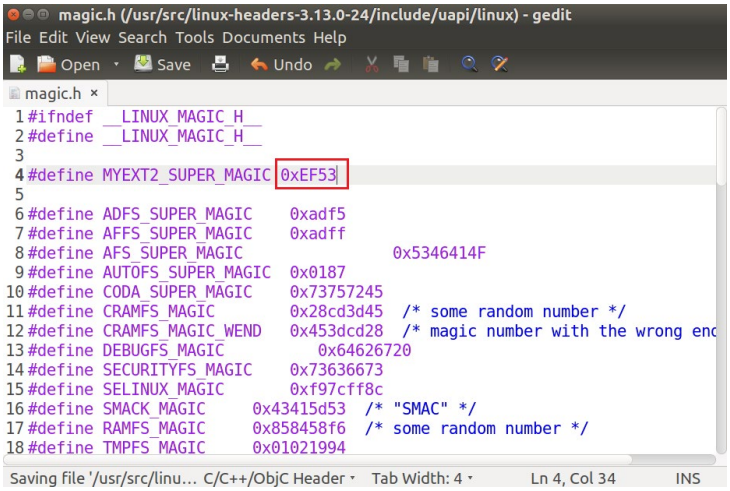
```
oslab@oslab-ubuntu-vm:myext2$ dmesg |grep haha
[ 3233.621582] haha, encrypt 7 bytes
[ 3237.949364] haha, decrypt 65536 bytes
[ 3237.949652] haha, decrypt 65536 bytes
[ 3296.760680] haha, decrypt 65536 bytes
[ 3296.760897] haha, decrypt 65536 bytes
```

可以看到输出了加解密的信息

使用文件管理器复制，然后用 gedit 查看

发现此时在读取文件时没有进行解密

将 myext2 的 magic number 改回 0xEF53



获取一个新的磁盘块并创建 ext2 文件系统

```
oslab@oslab-ubuntu-vm:test$ dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00414091 s, 253 MB/s
oslab@oslab-ubuntu-vm:test$ mkfs.ext2 ./myfs
mke2fs 1.42.9 (4-Feb-2014)
./myfs is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
Maximum filesystem blocks=1048576
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

oslab@oslab-ubuntu-vm:test$
```

用 myext2 去 mount，在 mnt 目录下重新创建 test 文件，写入 123456，并尝试读取 test

取消 myext2 的 mount 并用 ext2 重新 mount，然后再次尝试读取 test

```
oslab@oslab-ubuntu-vm:test$ sudo mount -t myext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ cd /mnt
oslab@oslab-ubuntu-vm:mnt$ ls
lost+found
oslab@oslab-ubuntu-vm:mnt$ sudo vim test
oslab@oslab-ubuntu-vm:mnt$ ls
lost+found  test
oslab@oslab-ubuntu-vm:mnt$ cat test
123456
oslab@oslab-ubuntu-vm:mnt$ cd ~/Desktop/lab3/test/
oslab@oslab-ubuntu-vm:test$ sudo umount /mnt
oslab@oslab-ubuntu-vm:test$ sudo mount -t ext2 -o loop ./myfs /mnt
oslab@oslab-ubuntu-vm:test$ cd /mnt
oslab@oslab-ubuntu-vm:mnt$ ls
lost+found  test
oslab@oslab-ubuntu-vm:mnt$ cat test
JKLMNO#oslab@oslab-ubuntu-vm:mnt$
```

可以看到用 ext2 重新 mount 后，读取文件时不会进行解密

5 遇到的困难与解决方案

1. 编译内核模块出错

刚开始使用了实验指导书中提到的 linux-3.18.24 版本的内核，然后在添加了 myext2 后尝试编译内核模块，发现报错，于是回退到修改前，直接编译 ext2 内核模块，发现仍然报错，猜测是内核版本的问题，于是换成了和系统内核相同版本的 linux-3.13.0 内核，顺利通过编译。

2. 在修改 magic number 时发现 `changeMN.c` 程序与测试时的输入不相符

给定的 `changeMN.c` 程序是一个无参程序，但是给的测试指令中却有参数，而且之后还要再次修改程序以适应不同的要求，于是直接修改了 `changeMN.c` 程序使其可以自定义输入输出，此外，纠正了原先 `changeMN.c` 程序中输出 magic number 的方式。

3. 为 myext2 文件系统读写添加加解密时出错

按照实验指导书中的描述修改文件后编译内核模块时出错，发现实验指导书中原先的文件读写操作绑定的是 `new_sync_read` 和 `new_sync_write` 函数，而自己的内核源码中，读写操作绑定的是 `do_sync_read` 和 `do_sync_write`，于是将新添加的 `new_sync_read_crypt` 中的 `new_sync_read` 函数改为了 `do_sync_read`，将 `new_sync_write_crypt` 中的 `new_sync_write` 改为了 `do_sync_write`，然后顺利通过编译。

6 心得与体会

本次实验没有编译内核，所以很快就做完了，实验的主要内容是添加一个类似 ext2 的文件系统，并做一些个性化更改，整个实验过程循序渐进，条里分明，通过这次实验，对 Linux 文件系统的理解更深了，对整个文件系统的运作流程有了一个大致的了解，知道了如何添加文件系统，如何建立自己想要的文件系统。

实验中遇到了许多困难，大部分是因为实验指导书上的描述和实际实验环境不相符所导致的，查阅资料学习相关知识后都得到了解决，也因此实验中学到了许多东西，希望可以在之后的课程学习中做的更好。