浙江大学

本科实验报告

课程名称：	操作系统
实验名称：	添加一个加密文件系统
姓　　名：	张溢弛
学　　院：	计算机学院
系：	软件工程
专　　业：	软件工程
学　　号：	3180103772
指导教师：	季江民

2020 年　　12 月　　27 日

目 录

浙江大学实验报告

课程名称：操作系统 实验类型：综合型

实验项目名称：添加一个加密文件系统

学生姓名：张溢弛 学号：3180103772

电子邮件地址：3180103772@zju.edu.cn

实验日期：2020 年 12 月 25 日

一、实验环境

- 操作系统 Windows 10 中文版
- 处理器：i7 芯片
- 内存 16G，硬盘空间 256G
- 虚拟机 Vmware Workstation Pro14
- Ubuntu 版本 12.04 发行版
- Linux 内核版本 3.13.0

二、实验内容以及结果分析

2.1 实验内容

本次试验需要添加一个类似于 ext2 但是在磁盘上对数据进行加密的文件系统 myext2，实验的主要内容包括：

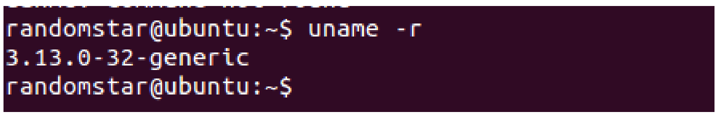
- 添加一个文件系统 muext2
- 修改 myext2 系统的 magic number
- 添加文件系统的创建工具
- 添加加密文件系统操作，包括 read_crypt 和 write_crypt 使其增加对加密数据的读写

主要的任务是修改内核的代码、编译并插入对应的内核模块，并对 myext2 文件系统进

行一些测试确保其功能有效。

2.2 实验过程

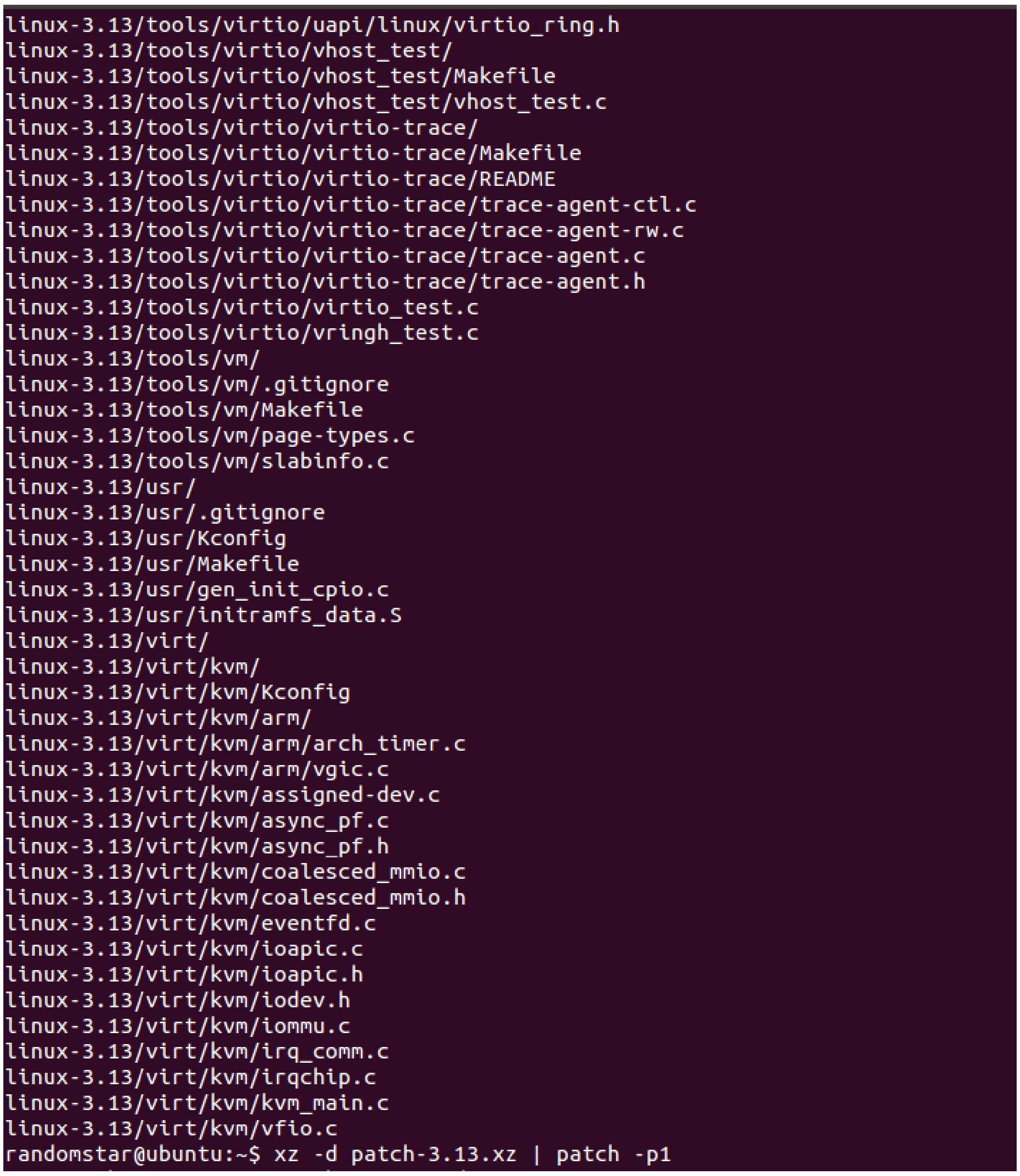
本次实验过程中原定将使用更换了 3.18.24 内核的 ubuntu16.04 版本来进行。但是我下载的时候发现 3.18.24 的版本内核已经被下架了，查阅资料之后发现 4 +版本以上的内核来做这个实验可能会需要客服一些原本不存在的困难，因此我重新安装了 Ubuntu12.04，使用其自带的 3.13.0 内核进行实验。



```
randomstar@ubuntu:~$ uname -r
3.13.0-32-generic
randomstar@ubuntu:~$
```

2.2.1 配置内核版本

按照实验 2 的步骤在阿里云镜像中下载一份 3.13.0 版本的内核源码和不定，使用 lab2 中类似的命令 tar －xvf 和 patch 命令进行解压并打补丁，完成之后正式开始实验。



```
linux-3.13/tools/virtio/uapi/linux/virtio_ring.h
linux-3.13/tools/virtio/vhost_test/
linux-3.13/tools/virtio/vhost_test/Makefile
linux-3.13/tools/virtio/vhost_test/vhost_test.c
linux-3.13/tools/virtio/virtio-trace/
linux-3.13/tools/virtio/virtio-trace/Makefile
linux-3.13/tools/virtio/virtio-trace/README
linux-3.13/tools/virtio/virtio-trace/trace-agent-ctl.c
linux-3.13/tools/virtio/virtio-trace/trace-agent-rw.c
linux-3.13/tools/virtio/virtio-trace/trace-agent.c
linux-3.13/tools/virtio/virtio-trace/trace-agent.h
linux-3.13/tools/virtio/virtio_test.c
linux-3.13/tools/virtio/vringh_test.c
linux-3.13/tools/vm/
linux-3.13/tools/vm/.gitignore
linux-3.13/tools/vm/Makefile
linux-3.13/tools/vm/page-types.c
linux-3.13/tools/vm/slabinfo.c
linux-3.13/usr/
linux-3.13/usr/.gitignore
linux-3.13/usr/Kconfig
linux-3.13/usr/Makefile
linux-3.13/usr/gen_init_cpio.c
linux-3.13/usr/initramfs_data.S
linux-3.13/virt/
linux-3.13/virt/kvm/
linux-3.13/virt/kvm/Kconfig
linux-3.13/virt/kvm/arm/
linux-3.13/virt/kvm/arm/arch_timer.c
linux-3.13/virt/kvm/arm/vgic.c
linux-3.13/virt/kvm/assigned-dev.c
linux-3.13/virt/kvm/async_pf.c
linux-3.13/virt/kvm/async_pf.h
linux-3.13/virt/kvm/coalesced_mmio.c
linux-3.13/virt/kvm/coalesced_mmio.h
linux-3.13/virt/kvm/eventfd.c
linux-3.13/virt/kvm/ioapic.c
linux-3.13/virt/kvm/ioapic.h
linux-3.13/virt/kvm/iodev.h
linux-3.13/virt/kvm/iommu.c
linux-3.13/virt/kvm/irq_comm.c
linux-3.13/virt/kvm/irqchip.c
linux-3.13/virt/kvm/kvm_main.c
linux-3.13/virt/kvm/vfio.c
randomstar@ubuntu:~$ xz -d patch-3.13.xz | patch -p1
```

2.2.2 源代码复制

首先按照实验指导书对内核源代码下面～/linux-3.18.24 中的代码进行一定的修改，首先新建一个 myext2 文件夹并将 ext2 中的代码复制过去，修改 ext2.h 为 myext2.h，使用的指令为：

```
#cd ~/linux-3.13  /* 内核源代码目录，假设内核源代码解压在主目录的 Linux-3.18.24 子目录中*/
```

```
#cd fs
#cp －R ext2 myext2
#cd ~/linux-3.13/fs/myext2
#mv ext2.h myext2.h
```

然后然后修改内核路径下的代码文件，复制一份对应的 myext2，具体要修改的有如下几个文件，这一部分截图丢失了，因此只放一下使用的指令，我想这样无伤大雅

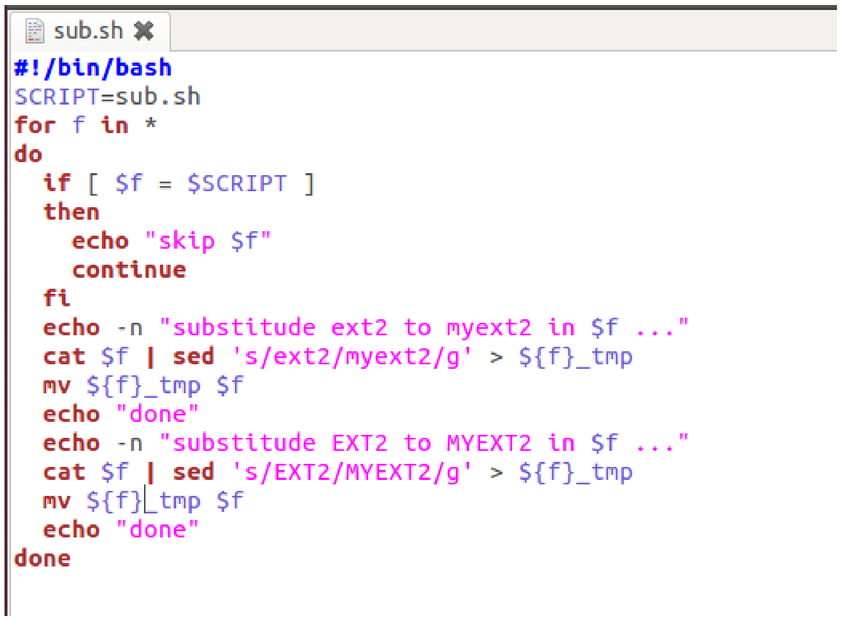
```
#cd /lib/modules/$(uname -r)/build /include/linux
#cp ext2_fs.h myext2_fs.h
#cd /lib/modules/$(uname -r)/build /include/asm-generic/bitops
#cp ext2-atomic.h myext2-atomic.h
#cp ext2-atomic-setbit.h myext2-atomic-setbit.h
```

2.2.3 内容替换

使用实验指导书上面的脚本 sub.sh 将克隆的文件系统目录下面的所有 ext2 和 EXT2 替换成 myext2 和 MYEXT2，sub 脚本如下：

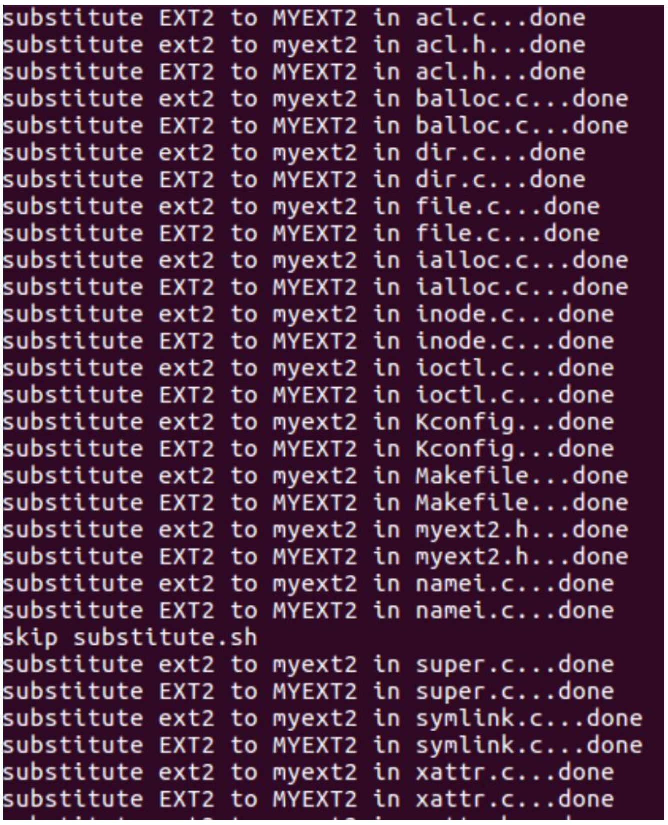
```
#!/bin/bash

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
```



```
#!/bin/bash
SCRIPT=sub.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
```

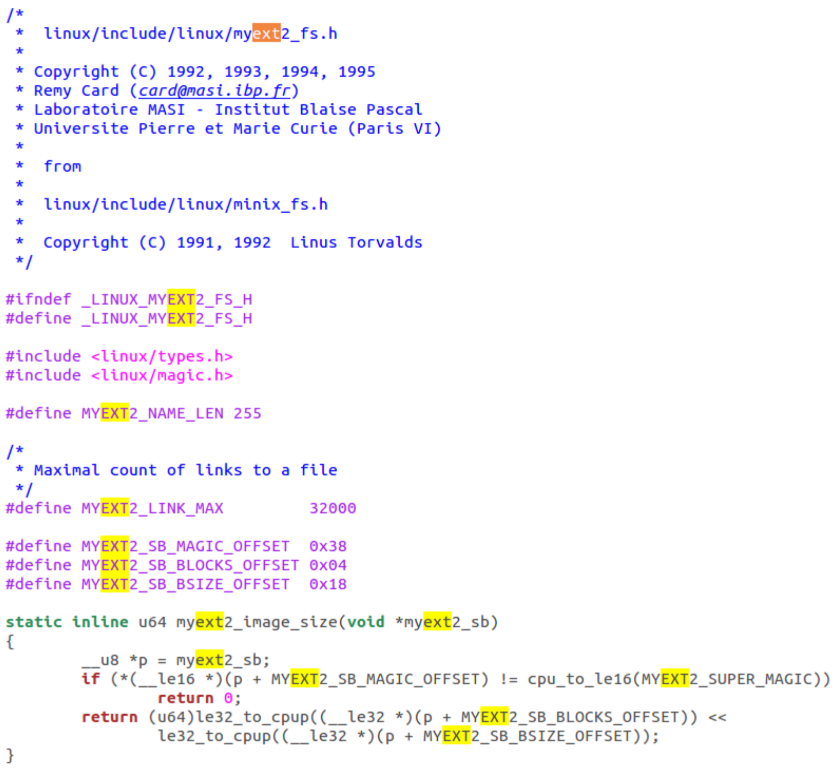
这里要注意不能直接复制，我自己在 gedit 中将这个脚本抄写了一遍，替换时的运行情况如图：



```
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 myext2.h...done
substitute EXT2 to MYEXT2 in myext2.h...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
```

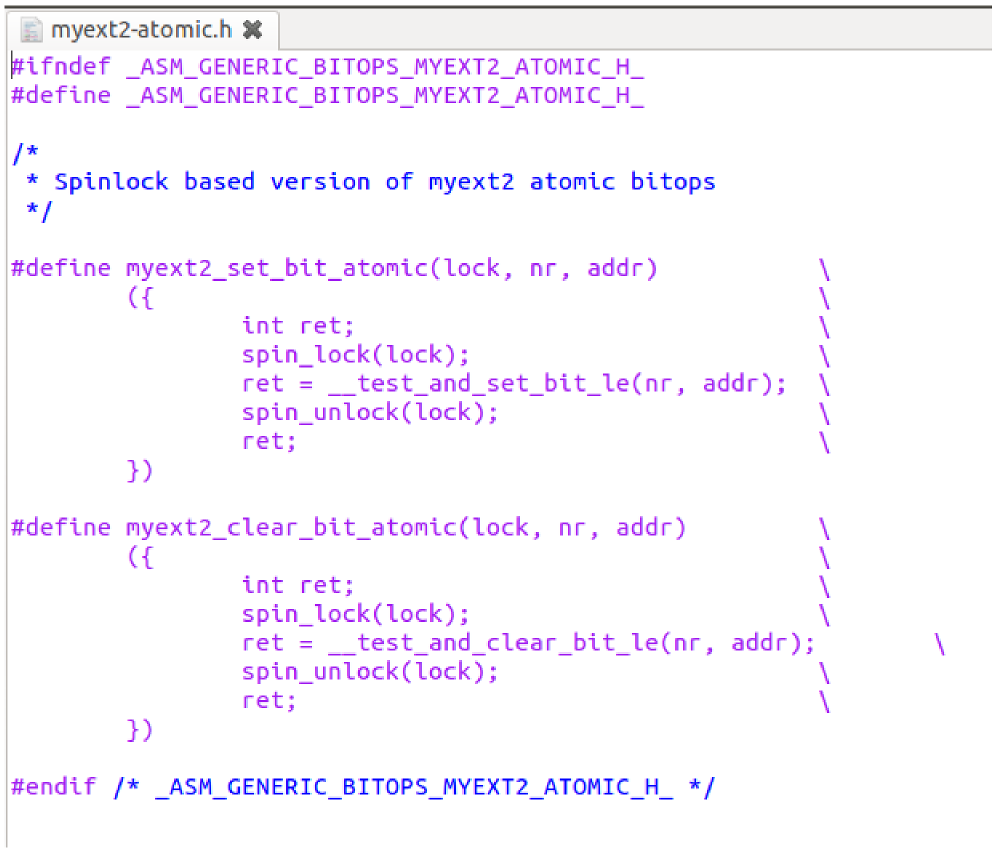
然后使用管理员模式运行文件管理器，进入系统内核的根目录将有关文件中的 ext2 全部进行替换，并添加相关的头文件，具体的截图如下图所示：

- 替换 myext2_fs.h



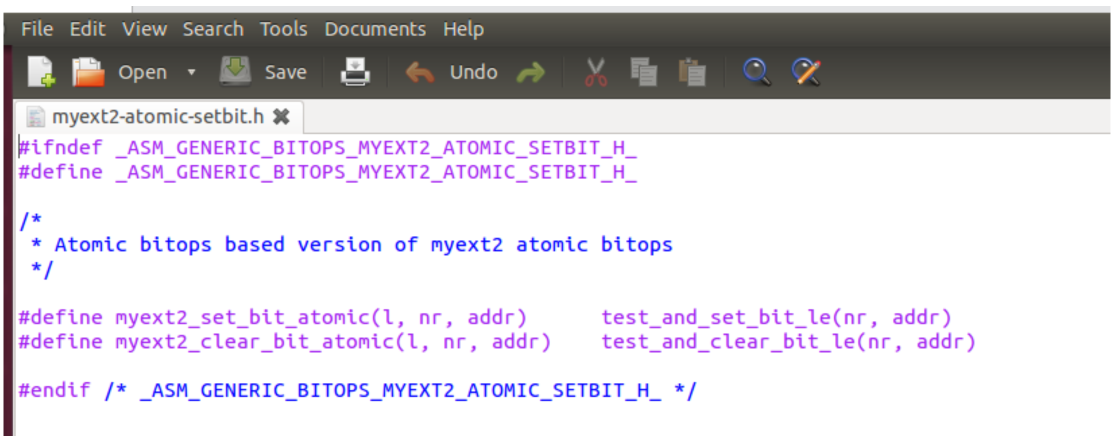
```
/*
 *  linux/include/linux/myext2_fs.h
 *
 * Copyright (C) 1992, 1993, 1994, 1995
 * Remy Card (card@masi.ibp.fr)
 * Laboratoire MASI - Institut Blaise Pascal
 * Universite Pierre et Marie Curie (Paris VI)
 *
 *  from
 *
 *  linux/include/linux/minix_fs.h
 *
 *  Copyright (C) 1991, 1992  Linus Torvalds
 */

#ifndef _LINUX_MYEXT2_FS_H
#define _LINUX_MYEXT2_FS_H

#include <linux/types.h>
#include <linux/magic.h>

#define MYEXT2_NAME_LEN 255

/*
 * Maximal count of links to a file
 */
#define MYEXT2_LINK_MAX         32000

#define MYEXT2_SB_MAGIC_OFFSET  0x38
#define MYEXT2_SB_BLOCKS_OFFSET 0x04
#define MYEXT2_SB_BSIZE_OFFSET  0x18

static inline u64 myext2_image_size(void *myext2_sb)
{
        __u8 *p = myext2_sb;
        if (*(__le16 *)(p + MYEXT2_SB_MAGIC_OFFSET) != cpu_to_le16(MYEXT2_SUPER_MAGIC))
                return 0;
        return (u64)le32_to_cpup((__le32 *)(p + MYEXT2_SB_BLOCKS_OFFSET)) <<
                le32_to_cpup((__le32 *)(p + MYEXT2_SB_BSIZE_OFFSET));
}
```

- 替换 myext2-atomic.h



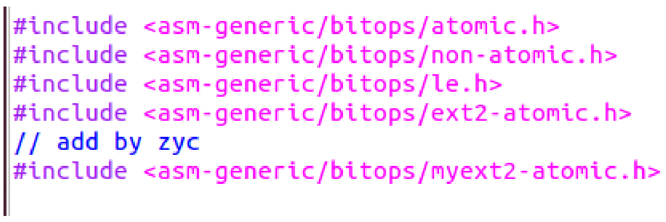
```
#ifndef _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_
#define _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_

/*
 * Spinlock based version of myext2 atomic bitops
 */

#define myext2_set_bit_atomic(lock, nr, addr)           \
        ({                                              \
                int ret;                                \
                spin_lock(lock);                        \
                ret = __test_and_set_bit_le(nr, addr);  \
                spin_unlock(lock);                      \
                ret;                                    \
        })

#define myext2_clear_bit_atomic(lock, nr, addr)         \
        ({                                              \
                int ret;                                \
                spin_lock(lock);                        \
                ret = __test_and_clear_bit_le(nr, addr);        \
                spin_unlock(lock);                      \
                ret;                                    \
        })

#endif /* _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_ */
```

- 替换 myext2-atomic-setbit.h

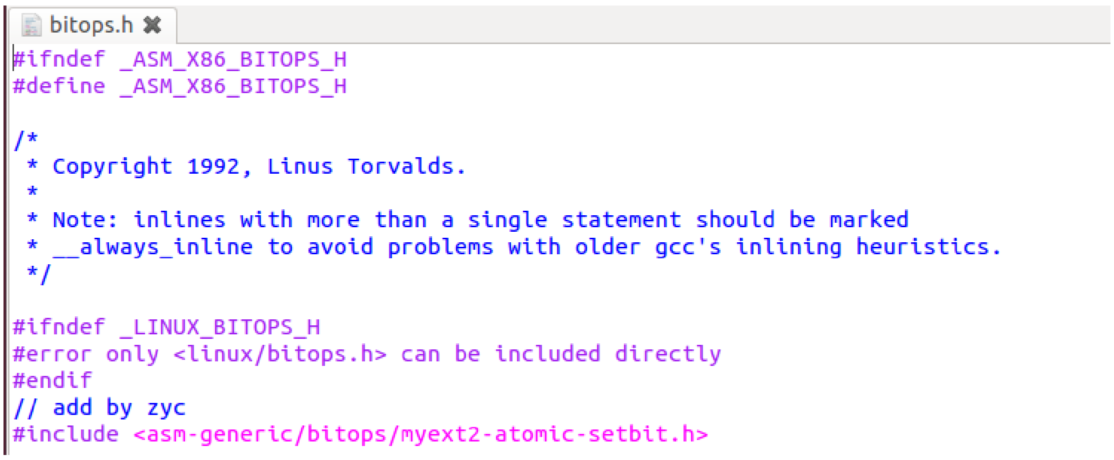


```
#ifndef _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_SETBIT_H_
#define _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_SETBIT_H_

/*
 * Atomic bitops based version of myext2 atomic bitops
 */

#define myext2_set_bit_atomic(l, nr, addr)      test_and_set_bit_le(nr, addr)
#define myext2_clear_bit_atomic(l, nr, addr)    test_and_clear_bit_le(nr, addr)

#endif /* _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_SETBIT_H_ */
```

- 添加若干个头文件

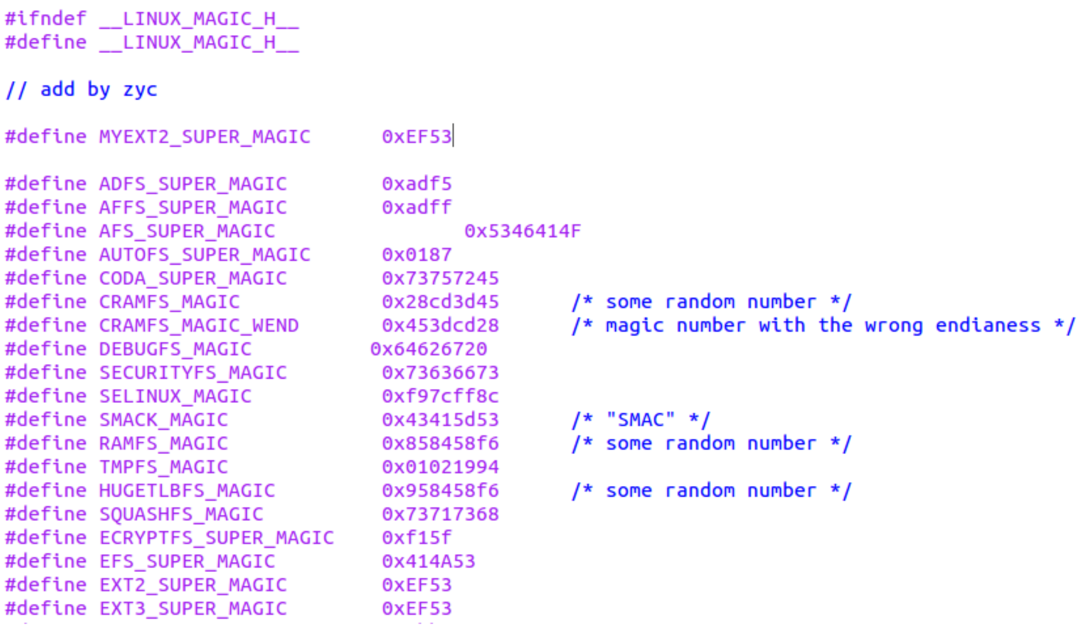


```
#include <asm-generic/bitops/atomic.h>
#include <asm-generic/bitops/non-atomic.h>
#include <asm-generic/bitops/le.h>
#include <asm-generic/bitops/ext2-atomic.h>
// add by zyc
#include <asm-generic/bitops/myext2-atomic.h>
```



```
#ifndef _ASM_X86_BITOPS_H
#define _ASM_X86_BITOPS_H

/*
 * Copyright 1992, Linus Torvalds.
 *
 * Note: inlines with more than a single statement should be marked
 * __always_inline to avoid problems with older gcc's inlining heuristics.
 */

#ifndef _LINUX_BITOPS_H
#error only <linux/bitops.h> can be included directly
#endif
// add by zyc
#include <asm-generic/bitops/myext2-atomic-setbit.h>
```

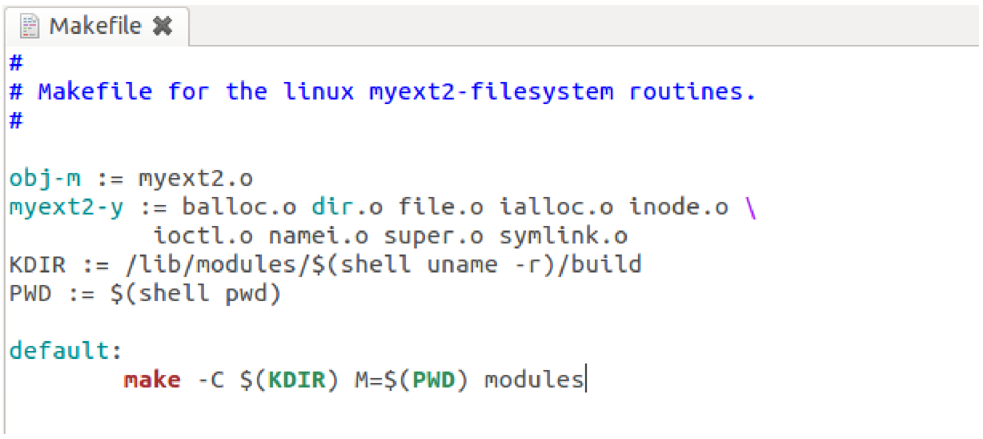
● 添加 magic number



```
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

// add by zyc

#define MYEXT2_SUPER_MAGIC      0xEF53

#define ADFS_SUPER_MAGIC        0xadf5
#define AFFS_SUPER_MAGIC        0xadff
#define AFS_SUPER_MAGIC                 0x5346414F
#define AUTOFS_SUPER_MAGIC      0x0187
#define CODA_SUPER_MAGIC        0x73757245
#define CRAMFS_MAGIC            0x28cd3d45      /* some random number */
#define CRAMFS_MAGIC_WEND       0x453dcd28      /* magic number with the wrong endianess */
#define DEBUGFS_MAGIC          0x64626720
#define SECURITYFS_MAGIC        0x73636673
#define SELINUX_MAGIC           0xf97cff8c
#define SMACK_MAGIC             0x43415d53      /* "SMAC" */
#define RAMFS_MAGIC             0x858458f6      /* some random number */
#define TMPFS_MAGIC             0x01021994
#define HUGETLBFS_MAGIC         0x958458f6      /* some random number */
#define SQUASHFS_MAGIC          0x73717368
#define ECRYPTFS_SUPER_MAGIC    0xf15f
#define EFS_SUPER_MAGIC         0x414A53
#define EXT2_SUPER_MAGIC        0xEF53
#define EXT3_SUPER_MAGIC        0xEF53
```

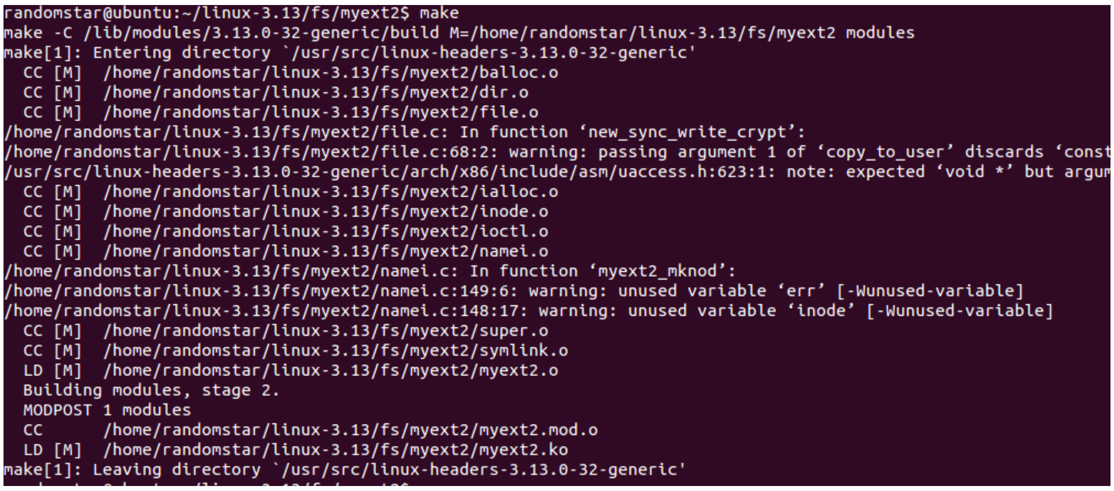
2.2.4 内核模块的编译

修改原本的 Makefile 文件并进行 make，编译成功之后就生成了内核模块



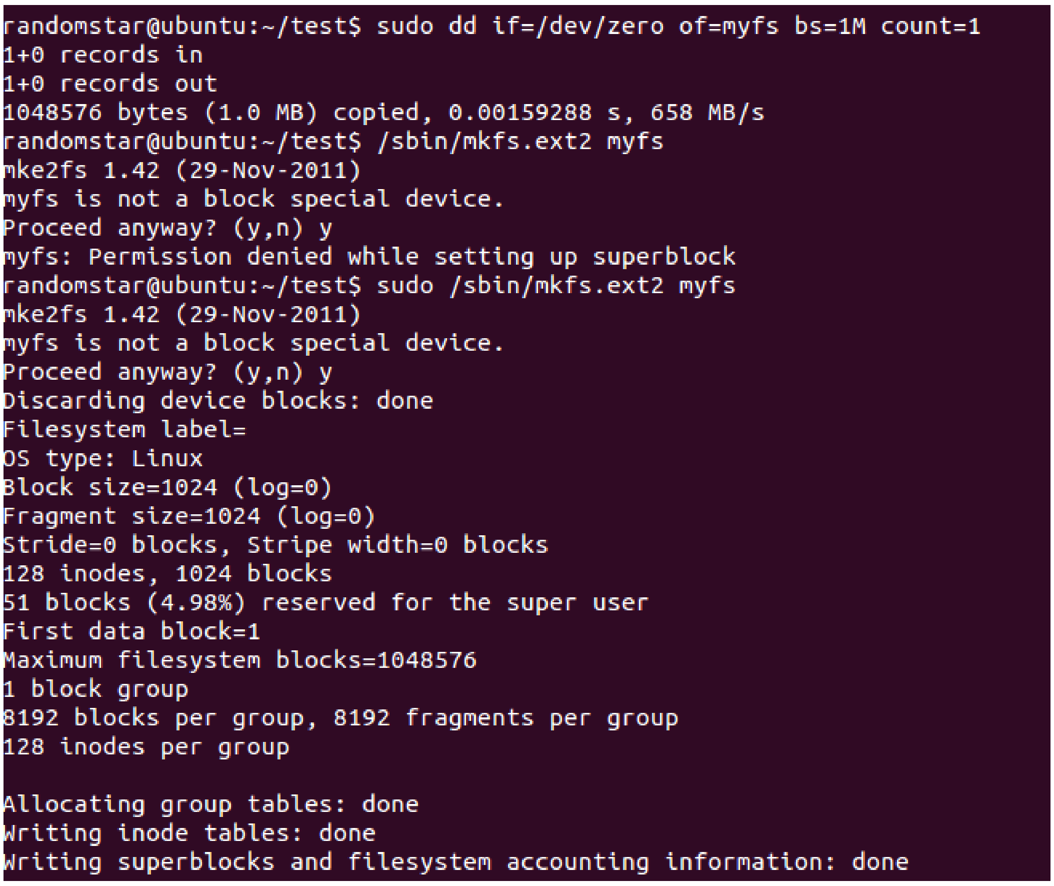
```
#
# Makefile for the linux myext2-filesystem routines.
#

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
```

编译并将生成的文件系统内核模块装载到内核中



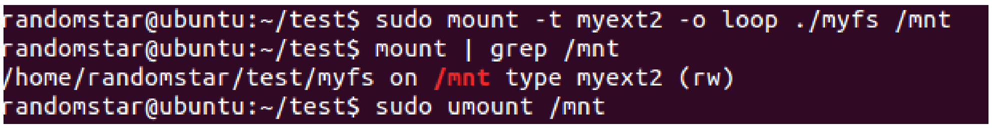
```
randomstar@ubuntu:~/linux-3.13/fs/myext2$ make
make -C /lib/modules/3.13.0-32-generic/build M=/home/randomstar/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-32-generic'
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/file.o
/home/randomstar/linux-3.13/fs/myext2/file.c: In function ‘new_sync_write_crypt’:
/home/randomstar/linux-3.13/fs/myext2/file.c:68:2: warning: passing argument 1 of ‘copy_to_user’ discards ‘const
/usr/src/linux-headers-3.13.0-32-generic/arch/x86/include/asm/uaccess.h:623:1: note: expected ‘void *’ but argum
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/namei.o
/home/randomstar/linux-3.13/fs/myext2/namei.c: In function ‘myext2_mknod’:
/home/randomstar/linux-3.13/fs/myext2/namei.c:149:6: warning: unused variable ‘err’ [-Wunused-variable]
/home/randomstar/linux-3.13/fs/myext2/namei.c:148:17: warning: unused variable ‘inode’ [-Wunused-variable]
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/super.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/randomstar/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-32-generic'
```

执行 cat /proc/filesystems | grep myext2 发现文件系统已经加载成功，然后挂载新的文件系统 myext2 并测试 mount 命令



```
randomstar@ubuntu:~/test$ sudo dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00159288 s, 658 MB/s
randomstar@ubuntu:~/test$ /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
myfs is not a block special device.
Proceed anyway? (y,n) y
myfs: Permission denied while setting up superblock
randomstar@ubuntu:~/test$ sudo /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
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
```

发现 myext2 可以被成功挂载，步骤 1 成功：

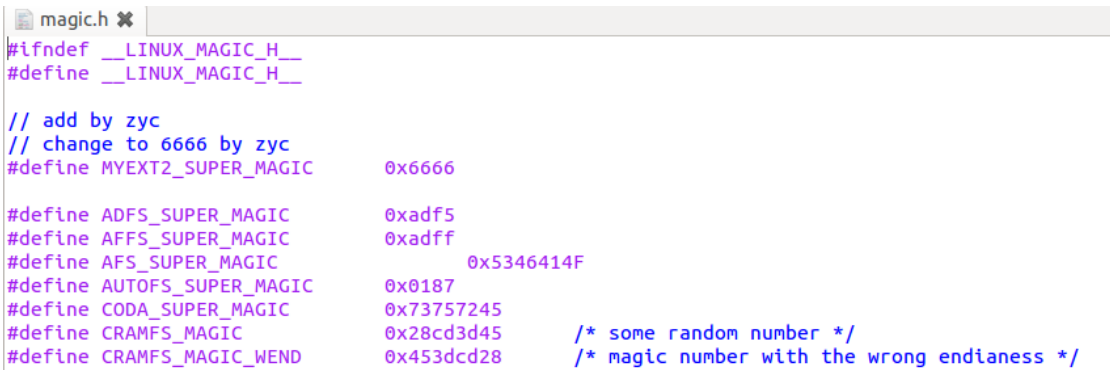


```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ mount | grep /mnt
/home/randomstar/test/myfs on /mnt type myext2 (rw)
randomstar@ubuntu:~/test$ sudo umount /mnt
```

挂载成功之后需要卸载该文件系统和 myext2 模块，用 umount 和 rmmod 分别卸载文件系统和内核模块即可

2.2.5 修改 magic number

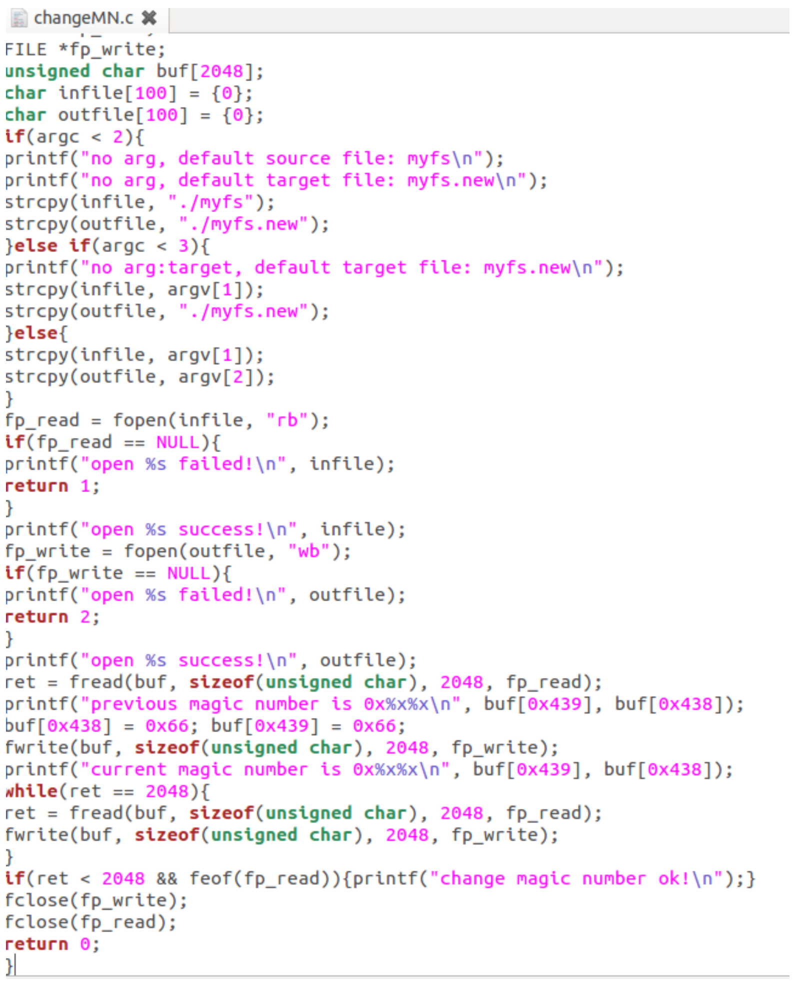
上面的几个步骤之后文件系统已经成功挂载，然后就需要找到 myext2 的 magic number 并修改成 0x6666，此处的 magic.h 是内核代码中的 magic number



```
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

// add by zyc
// change to 6666 by zyc
#define MYEXT2_SUPER_MAGIC      0x6666

#define ADFS_SUPER_MAGIC        0xadf5
#define AFFS_SUPER_MAGIC        0xadff
#define AFS_SUPER_MAGIC                 0x5346414F
#define AUTOFS_SUPER_MAGIC      0x0187
#define CODA_SUPER_MAGIC        0x73757245
#define CRAMFS_MAGIC            0x28cd3d45      /* some random number */
#define CRAMFS_MAGIC_WEND       0x453dcd28      /* magic number with the wrong endianess */
```

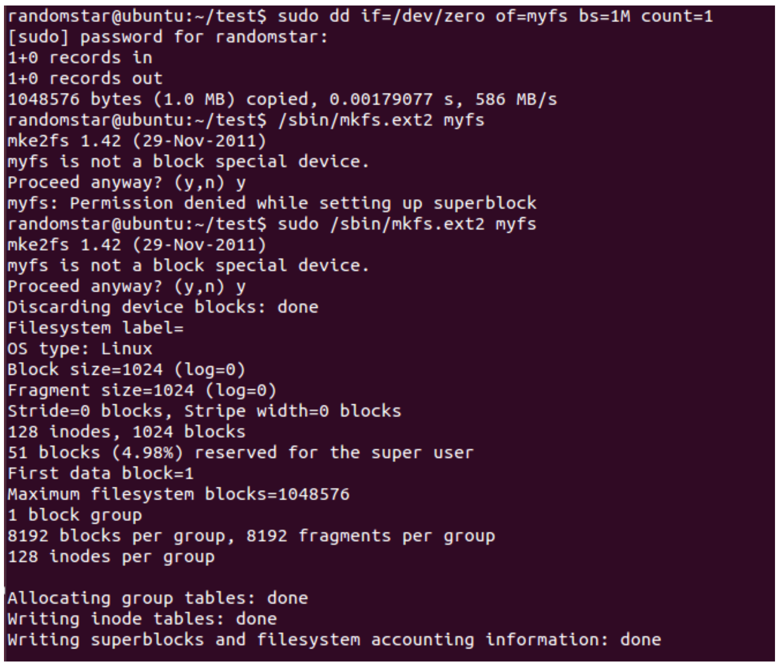
然后重新编译并插入模块，挂载文件系统之后用 changeMN.c 来测试 magic number 是否已经被更换

● ChangeMN 脚本如下图所示，从学在浙大中下载

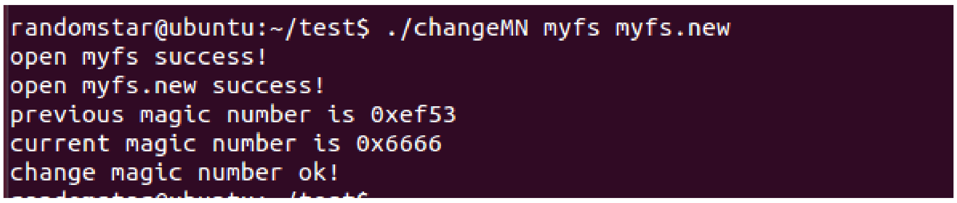


```
FILE *fp_write;
unsigned char buf[2048];
char infile[100] = {0};
char outfile[100] = {0};
if(argc < 2){
printf("no arg, default source file: myfs\n");
printf("no arg, default target file: myfs.new\n");
strcpy(infile, "./myfs");
strcpy(outfile, "./myfs.new");
}else if(argc < 3){
printf("no arg:target, 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;
}
```

● 运行脚本进行 magic number 的测试

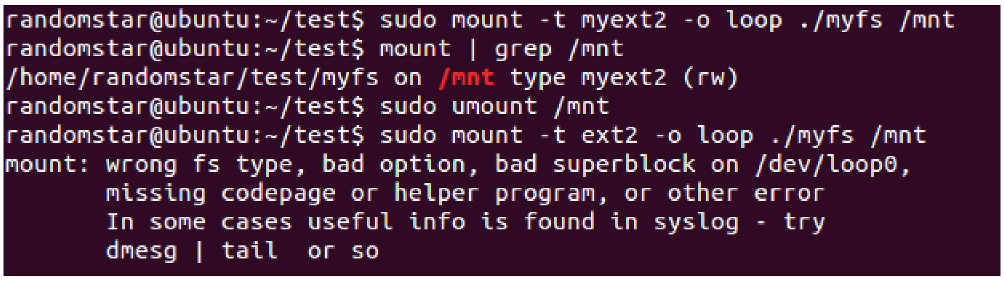


```
randomstar@ubuntu:~/test$ sudo dd if=/dev/zero of=myfs bs=1M count=1
[sudo] password for randomstar:
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00179077 s, 586 MB/s
randomstar@ubuntu:~/test$ /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
myfs is not a block special device.
Proceed anyway? (y,n) y
myfs: Permission denied while setting up superblock
randomstar@ubuntu:~/test$ sudo /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
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
```



```
randomstar@ubuntu:~/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!
```

之后卸载/mnt 并重新挂载基于 ext2 的文件系统再测试 mount 发现报错，因为两个文件系统的 magic number 不匹配导致的，说明我们修改 magic number 已经成功了。然后需要卸载 myext2 模块

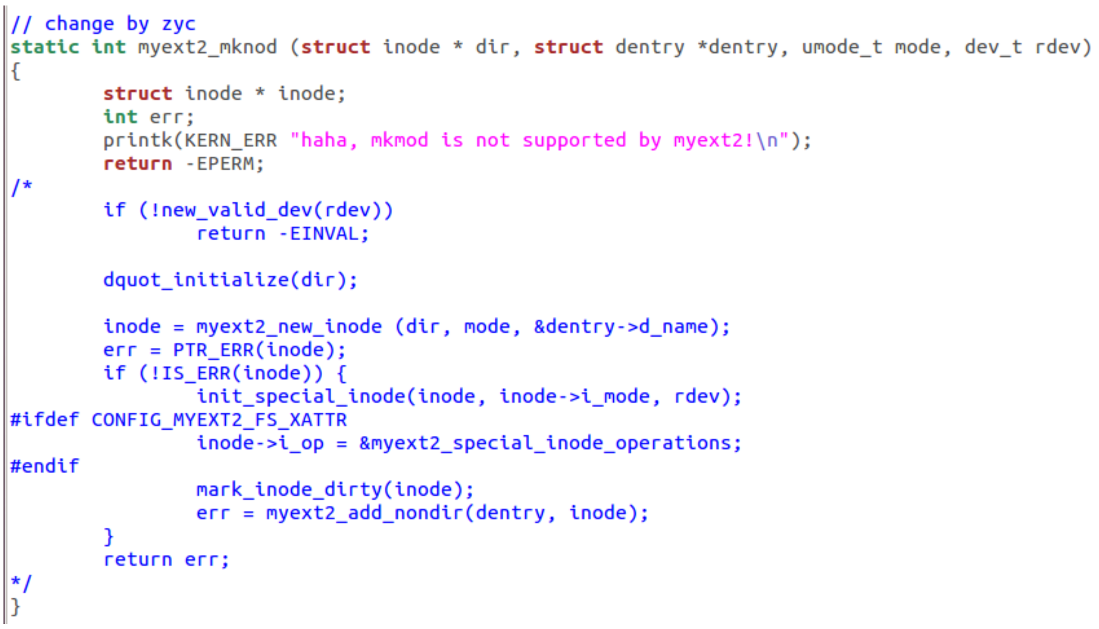


```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ mount | grep /mnt
/home/randomstar/test/myfs on /mnt type myext2 (rw)
randomstar@ubuntu:~/test$ sudo umount /mnt
randomstar@ubuntu:~/test$ sudo mount -t ext2 -o loop ./myfs /mnt
mount: wrong fs type, bad 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
```

2.2.6 修改文件系统操作

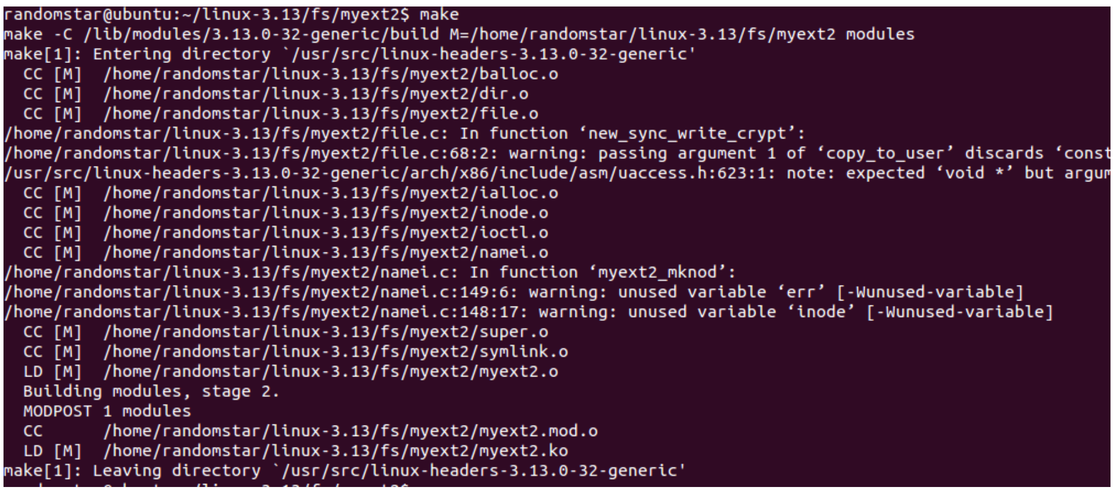
首先修改 mknod 函数的功能，修改之后重新 make 并插入模块，发现 mknod 的操作已经不被允许，可见裁剪取得了预期的效果

● 修改 mknod 功能



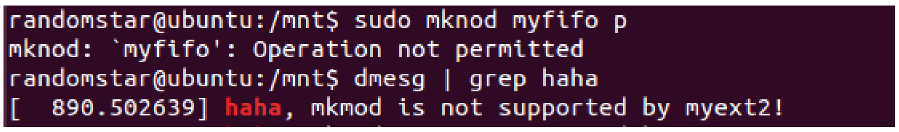
```
// change by zyc
static int myext2_mknod (struct inode * dir, struct dentry *dentry, umode_t mode, dev_t rdev)
{
        struct inode * inode;
        int err;
        printk(KERN_ERR "haha, mkmod is not supported by myext2!\n");
        return -EPERM;
/*
        if (!new_valid_dev(rdev))
                return -EINVAL;

        dquot_initialize(dir);

        inode = myext2_new_inode (dir, mode, &dentry->d_name);
        err = PTR_ERR(inode);
        if (!IS_ERR(inode)) {
                init_special_inode(inode, inode->i_mode, rdev);
#ifdef CONFIG_MYEXT2_FS_XATTR
                inode->i_op = &myext2_special_inode_operations;
#endif
                mark_inode_dirty(inode);
                err = myext2_add_nondir(dentry, inode);
        }
        return err;
*/
}
```

● 编译并插入新的模块



```
randomstar@ubuntu:~/linux-3.13/fs/myext2$ make
make -C /lib/modules/3.13.0-32-generic/build M=/home/randomstar/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-32-generic'
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/file.o
/home/randomstar/linux-3.13/fs/myext2/file.c: In function ‘new_sync_write_crypt’:
/home/randomstar/linux-3.13/fs/myext2/file.c:68:2: warning: passing argument 1 of ‘copy_to_user’ discards ‘const
/usr/src/linux-headers-3.13.0-32-generic/arch/x86/include/asm/uaccess.h:623:1: note: expected ‘void *’ but argu
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/namei.o
/home/randomstar/linux-3.13/fs/myext2/namei.c: In function ‘myext2_mknod’:
/home/randomstar/linux-3.13/fs/myext2/namei.c:149:6: warning: unused variable ‘err’ [-Wunused-variable]
/home/randomstar/linux-3.13/fs/myext2/namei.c:148:17: warning: unused variable ‘inode’ [-Wunused-variable]
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/super.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/randomstar/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-32-generic'
```

● 尝试使用 mknod 指令，发现被拒绝执行，然后查看 dmesg 中的系统日志，发现输出了 haha

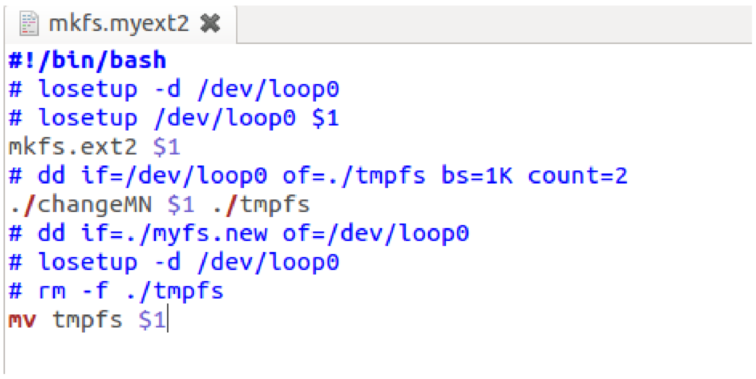


```
randomstar@ubuntu:/mnt$ sudo mknod myfifo p
mknod: `myfifo': Operation not permitted
randomstar@ubuntu:/mnt$ dmesg | grep haha
[  890.502639] haha, mkmod is not supported by myext2!
```

2.2.7 添加文件系统创建工具

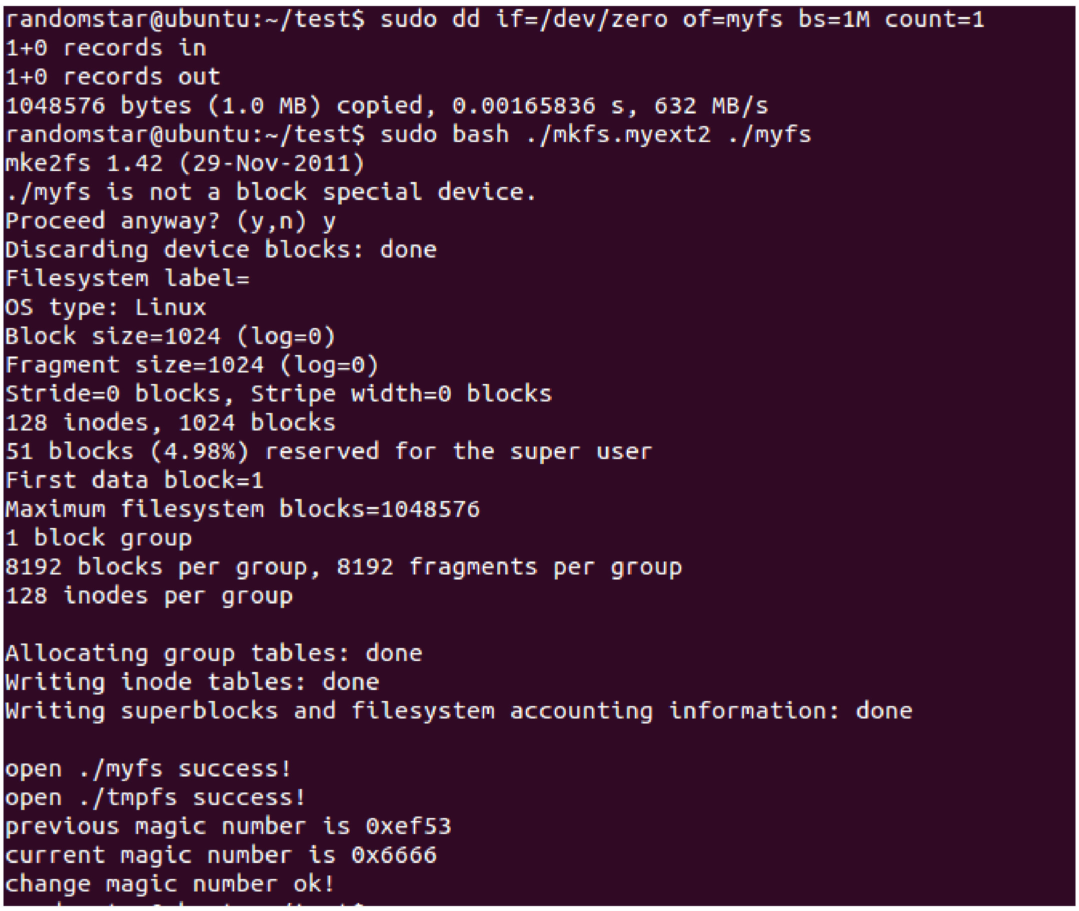
首先按照实验指导书编辑一个 mkfs.myext2 脚本，但是为了让其可以适应 3.13 内核版本，我将其进行了一定的修改，但是并不影响文件系统创建工具的使用。

这一步实验的主要目的在于尝试添加一个文件系统的创建工具，因此虽然这里的创建脚本和老师发的实验指导书上略有区别，也不影响尝试添加创建工具这一实验目的，因为老师给的版本的脚本似乎确实不适用于这个版本的内核：

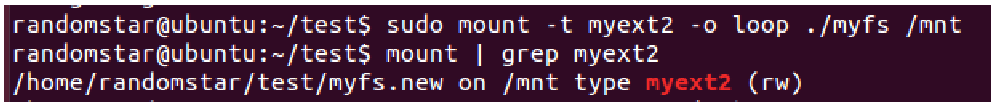


```
#!/bin/bash
# losetup -d /dev/loop0
# losetup /dev/loop0 $1
mkfs.ext2 $1
# dd if=/dev/loop0 of=./tmpfs bs=1K count=2
./changeMN $1 ./tmpfs
# dd if=./myfs.new of=/dev/loop0
# losetup -d /dev/loop0
# rm -f ./tmpfs
mv tmpfs $1
```

然后修改 changeMN.c 的代码并接受命令行的参数，运行的结果如下：



```
randomstar@ubuntu:~/test$ sudo dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00165836 s, 632 MB/s
randomstar@ubuntu:~/test$ sudo bash ./mkfs.myext2 ./myfs
mke2fs 1.42 (29-Nov-2011)
./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!
```



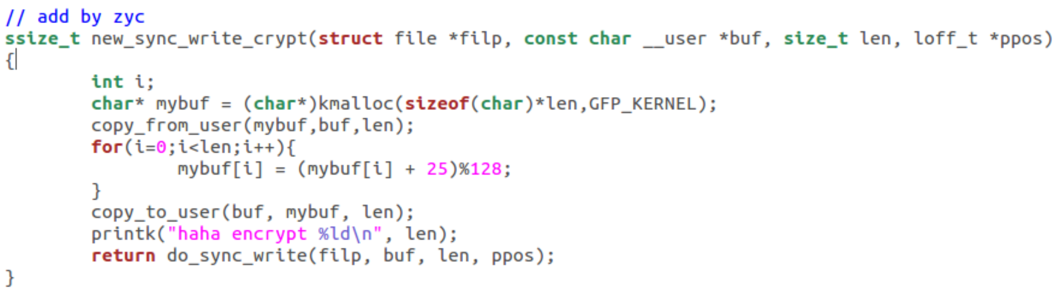
```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ mount | grep myext2
/home/randomstar/test/myfs.new on /mnt type myext2 (rw)
```

- 上述输出结果说明我们成功添加了文件系统的创建系统

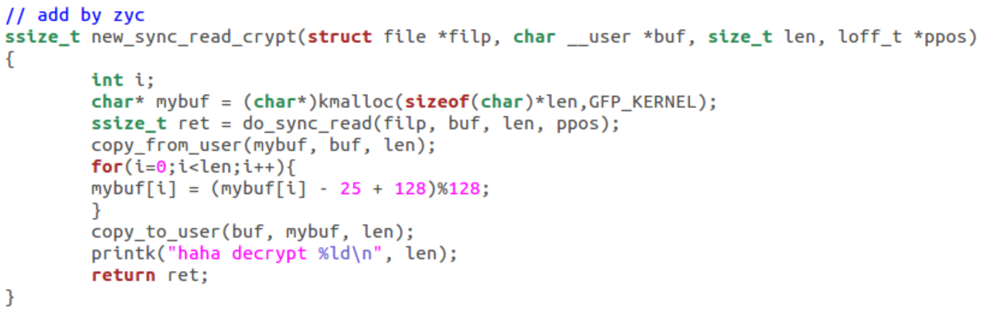
2.2.8 修改 read 和 write 操作

修改 file.c 的代码添加两个新的函数，实现对用户传入数据 buf 的加密和解密，并将两个函数指针赋给 myext2_file_oprations 的结构中的 read 和 write

- 两个加密函数的代码

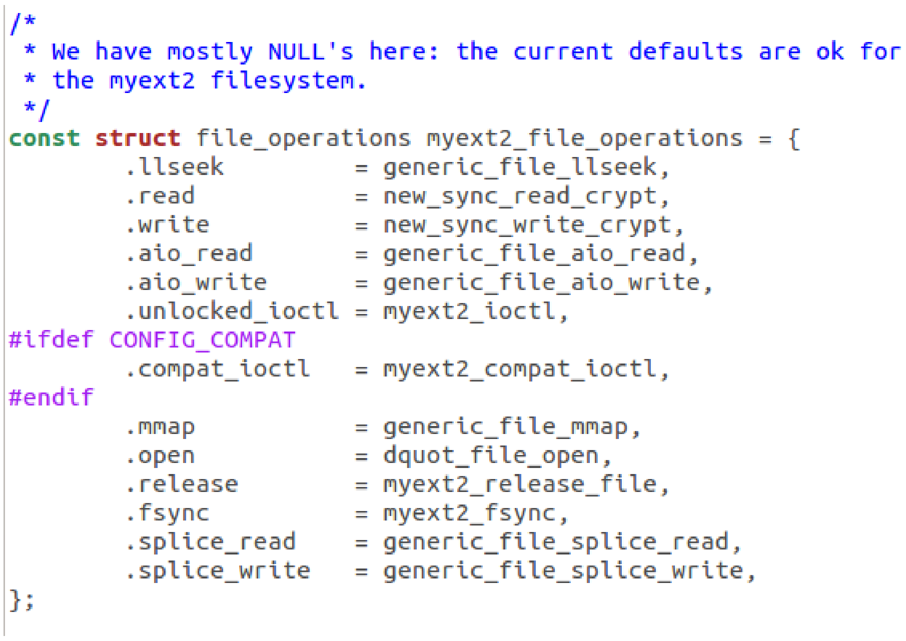


```
// add by zyc
ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t len, loff_t *ppos)
{
        int i;
        char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
        copy_from_user(mybuf,buf,len);
        for(i=0;i<len;i++){
                mybuf[i] = (mybuf[i]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



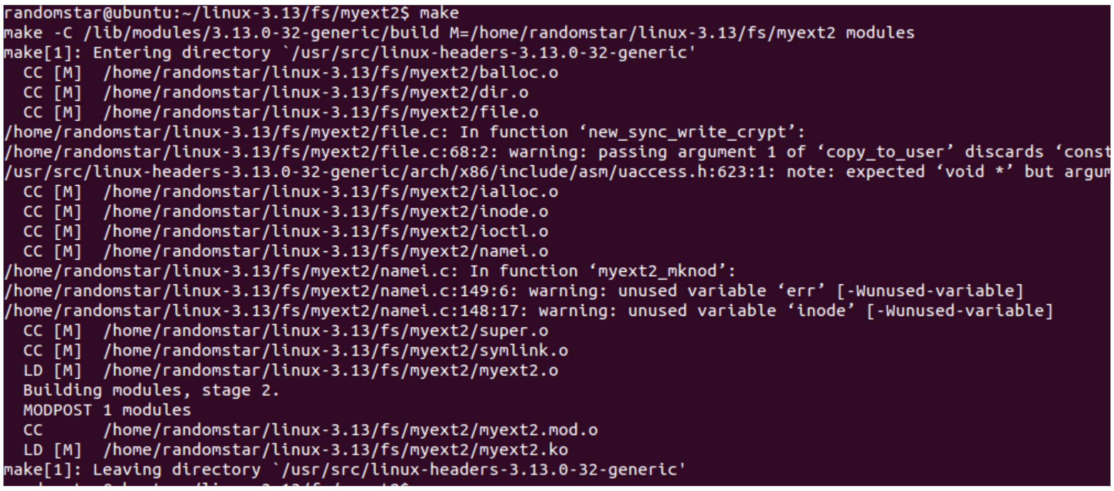
```
// add by zyc
ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len, loff_t *ppos)
{
        int i;
        char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
        ssize_t ret = do_sync_read(filp, buf, len, ppos);
        copy_from_user(mybuf, buf, len);
        for(i=0;i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

- 这里要注意新的加密读写函数要和原本的明文读写函数一一对应，否则编译的时候会出错
- file_operations 结构体的修改



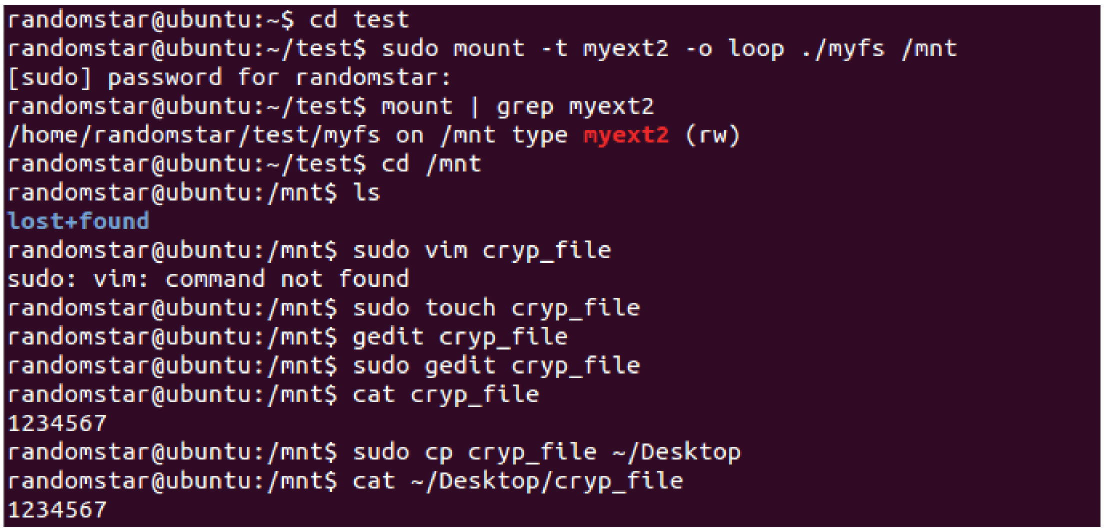
```
/*
 * We have mostly NULL's here: the current defaults are ok for
 * the myext2 filesystem.
 */
const struct file_operations myext2_file_operations = {
	.llseek		= generic_file_llseek,
	.read		= new_sync_read_crypt,
	.write		= new_sync_write_crypt,
	.aio_read	= generic_file_aio_read,
	.aio_write	= generic_file_aio_write,
	.unlocked_ioctl = myext2_ioctl,
#ifdef CONFIG_COMPAT
	.compat_ioctl	= myext2_compat_ioctl,
#endif
	.mmap		= generic_file_mmap,
	.open		= dquot_file_open,
	.release	= myext2_release_file,
	.fsync		= myext2_fsync,
	.splice_read	= generic_file_splice_read,
	.splice_write	= generic_file_splice_write,
};
```

然后重新编译 myext2 内核模块，使用命令 insmod 安装模块，重新加载 myext2 的内核模块并创建一个 myext2 文件系统，尝试在文件中写入一个字符串文件



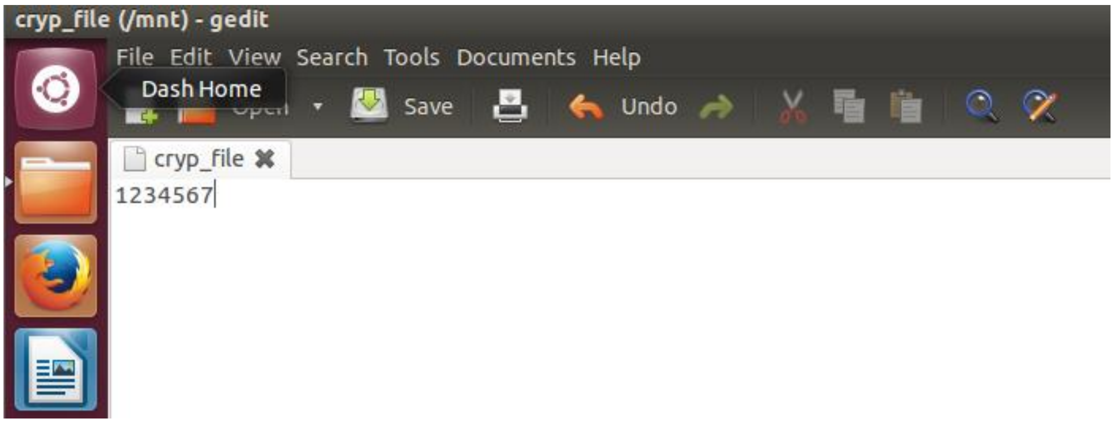
```
randomstar@ubuntu:~/linux-3.13/fs/myext2$ make
make -C /lib/modules/3.13.0-32-generic/build M=/home/randomstar/linux-3.13/fs/myext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-32-generic'
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/file.o
/home/randomstar/linux-3.13/fs/myext2/file.c: In function ‘new_sync_write_crypt’:
/home/randomstar/linux-3.13/fs/myext2/file.c:68:2: warning: passing argument 1 of ‘copy_to_user’ discards ‘const
/usr/src/linux-headers-3.13.0-32-generic/arch/x86/include/asm/uaccess.h:623:1: note: expected ‘void *’ but argum
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/namei.o
/home/randomstar/linux-3.13/fs/myext2/namei.c: In function ‘myext2_mknod’:
/home/randomstar/linux-3.13/fs/myext2/namei.c:149:6: warning: unused variable ‘err’ [-Wunused-variable]
/home/randomstar/linux-3.13/fs/myext2/namei.c:148:17: warning: unused variable ‘inode’ [-Wunused-variable]
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/super.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/randomstar/linux-3.13/fs/myext2/myext2.mod.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-32-generic'
```

- 成功之后在 mnt 下面新建并编辑一个文件，写入 1234567



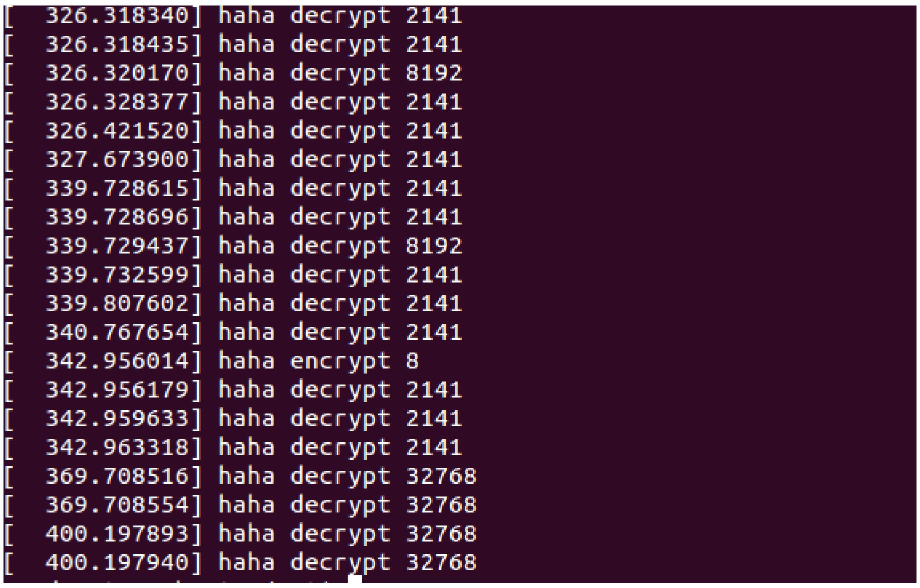
```
randomstar@ubuntu:~$ cd test
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
[sudo] password for randomstar:
randomstar@ubuntu:~/test$ mount | grep myext2
/home/randomstar/test/myfs on /mnt type myext2 (rw)
randomstar@ubuntu:~/test$ cd /mnt
randomstar@ubuntu:/mnt$ ls
lost+found
randomstar@ubuntu:/mnt$ sudo vim cryp_file
sudo: vim: command not found
randomstar@ubuntu:/mnt$ sudo touch cryp_file
randomstar@ubuntu:/mnt$ gedit cryp_file
randomstar@ubuntu:/mnt$ sudo gedit cryp_file
randomstar@ubuntu:/mnt$ cat cryp_file
1234567
randomstar@ubuntu:/mnt$ sudo cp cryp_file ~/Desktop
randomstar@ubuntu:/mnt$ cat ~/Desktop/cryp_file
1234567
```

- Mnt 下面用 gedit 打开看到的依然是 1234567



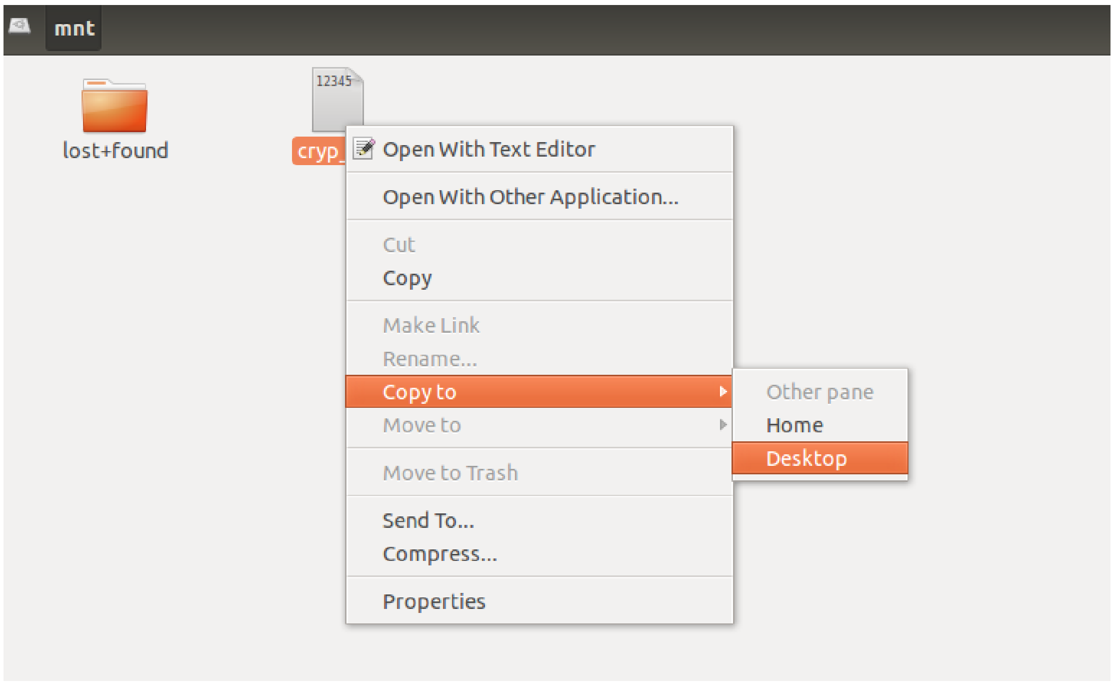
```
1234567
```

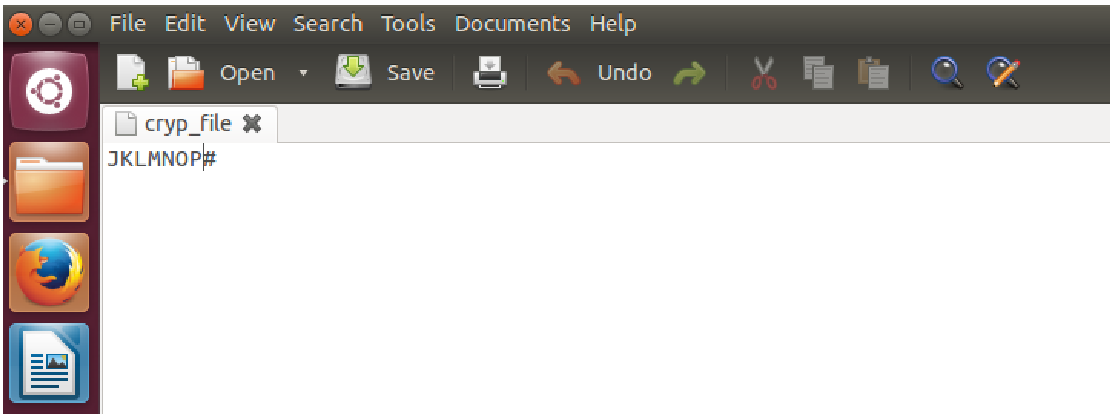
- 查看系统日志，发现进行了加密（根据修改的内核代码，我这里输出格式是 ld，因此并不是加密了 1234567，而输出成这样可能也和操作系统读取数据的大小端规则有一定的关系）



```
[  326.318340] haha decrypt 2141
[  326.318435] haha decrypt 2141
[  326.320170] haha decrypt 8192
[  326.328377] haha decrypt 2141
[  326.421520] haha decrypt 2141
[  327.673900] haha decrypt 2141
[  339.728615] haha decrypt 2141
[  339.728696] haha decrypt 2141
[  339.729437] haha decrypt 8192
[  339.732599] haha decrypt 2141
[  339.807602] haha decrypt 2141
[  340.767654] haha decrypt 2141
[  342.956014] haha encrypt 8
[  342.956179] haha decrypt 2141
[  342.959633] haha decrypt 2141
[  342.963318] haha decrypt 2141
[  369.708516] haha decrypt 32768
[  369.708554] haha decrypt 32768
[  400.197893] haha decrypt 32768
[  400.197940] haha decrypt 32768
```

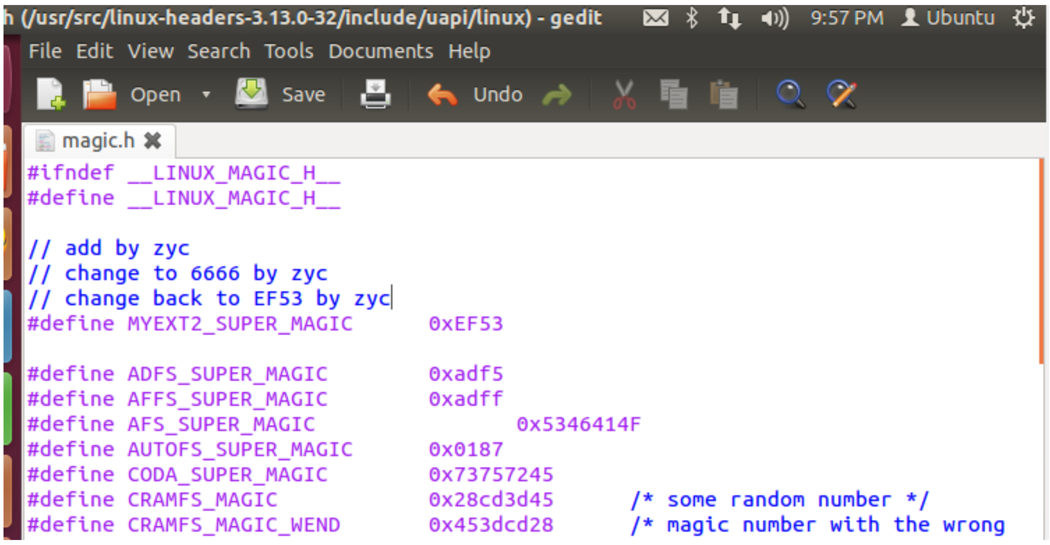
在 mnt 下面查看文件之后使用 cp 命令 copy 到主目录下进行查看，发现看到的内容还是 1234567，但是在使用文件管理器复制查看的时候显示出来的是已经被加密的文本





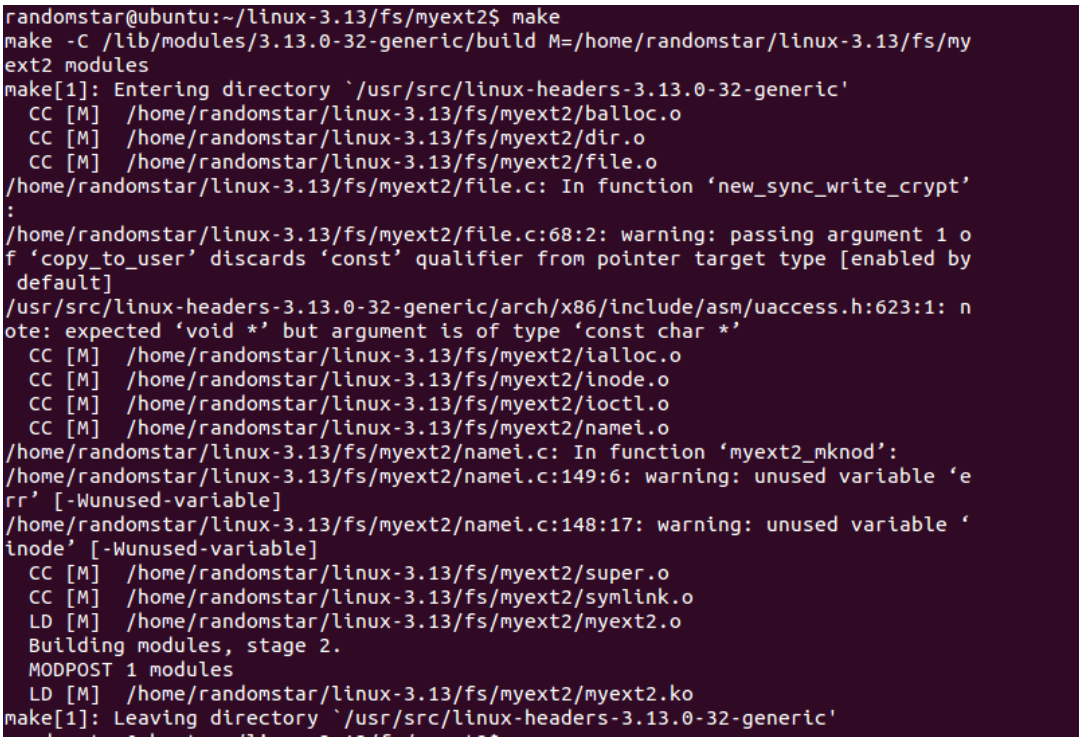
```
JKLMNOP#
```

此时将 magic number 再修改回来之后重新编译，发现修改了之后显示出来的依然是加密的文件


h (/usr/src/linux-headers-3.13.0-32/include/uapi/linux) - gedit
9:57 PM
Ubuntu
File Edit View Search Tools Documents Help
Open
Save
Undo
magic.h

```
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

// add by zyc
// change to 6666 by zyc
// change back to EF53 by zyc
#define MYEXT2_SUPER_MAGIC      0xEF53

#define ADFS_SUPER_MAGIC        0xadf5
#define AFFS_SUPER_MAGIC        0xadff
#define AFS_SUPER_MAGIC                 0x5346414F
#define AUTOFS_SUPER_MAGIC      0x0187
#define CODA_SUPER_MAGIC        0x73757245
#define CRAMFS_MAGIC            0x28cd3d45      /* some random number */
#define CRAMFS_MAGIC_WEND       0x453dcd28      /* magic number with the wrong
```

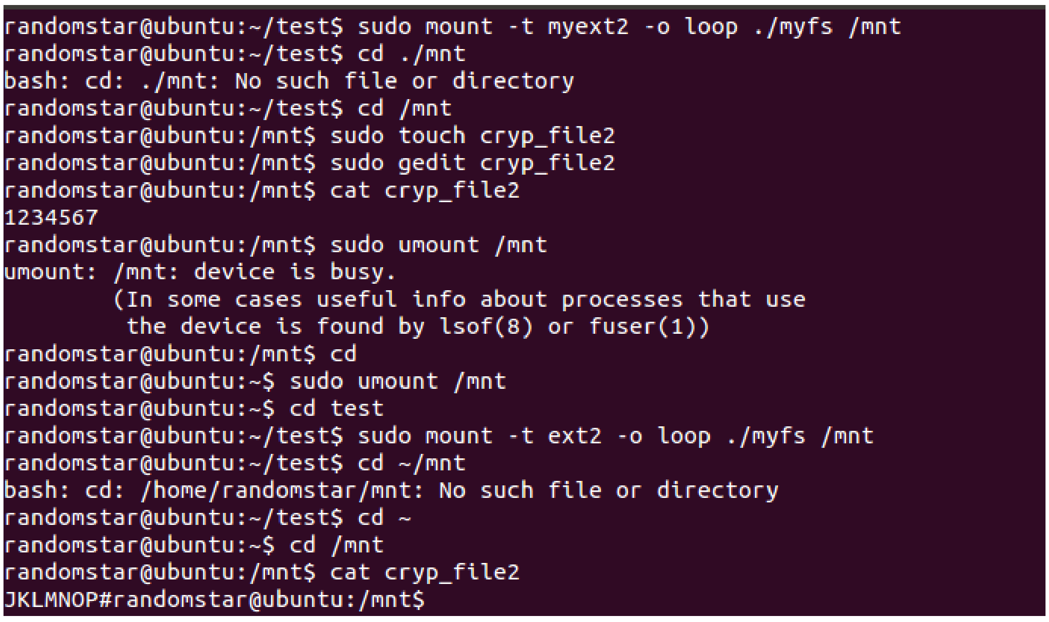
重新编译一次 myext2 的模块并挂载



```
randomstar@ubuntu:~/linux-3.13/fs/myext2$ make
make -C /lib/modules/3.13.0-32-generic/build M=/home/randomstar/linux-3.13/fs/my
ext2 modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-32-generic'
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/balloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/dir.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/file.o
/home/randomstar/linux-3.13/fs/myext2/file.c: In function ‘new_sync_write_crypt’
:
/home/randomstar/linux-3.13/fs/myext2/file.c:68:2: warning: passing argument 1 o
f ‘copy_to_user’ discards ‘const’ qualifier from pointer target type [enabled by
 default]
/usr/src/linux-headers-3.13.0-32-generic/arch/x86/include/asm/uaccess.h:623:1: n
ote: expected ‘void *’ but argument is of type ‘const char *’
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ialloc.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/inode.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/ioctl.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/namei.o
/home/randomstar/linux-3.13/fs/myext2/namei.c: In function ‘myext2_mknod’:
/home/randomstar/linux-3.13/fs/myext2/namei.c:149:6: warning: unused variable ‘e
rr’ [-Wunused-variable]
/home/randomstar/linux-3.13/fs/myext2/namei.c:148:17: warning: unused variable ‘
inode’ [-Wunused-variable]
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/super.o
  CC [M]  /home/randomstar/linux-3.13/fs/myext2/symlink.o
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/randomstar/linux-3.13/fs/myext2/myext2.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-32-generic'
```

进行测试，发现仍然显示加密后的内容：

中间的一些指令出现了操作上的小失误不过我立马纠正了，具体的而结果如下：



```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ cd ./mnt
bash: cd: ./mnt: No such file or directory
randomstar@ubuntu:~/test$ cd /mnt
randomstar@ubuntu:/mnt$ sudo touch cryp_file2
randomstar@ubuntu:/mnt$ sudo gedit cryp_file2
randomstar@ubuntu:/mnt$ cat cryp_file2
1234567
randomstar@ubuntu:/mnt$ sudo umount /mnt
umount: /mnt: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
randomstar@ubuntu:/mnt$ cd
randomstar@ubuntu:~$ sudo umount /mnt
randomstar@ubuntu:~$ cd test
randomstar@ubuntu:~/test$ sudo mount -t ext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ cd ~/mnt
bash: cd: /home/randomstar/mnt: No such file or directory
randomstar@ubuntu:~/test$ cd ~
randomstar@ubuntu:~$ cd /mnt
randomstar@ubuntu:/mnt$ cat cryp_file2
JKLMNOP#randomstar@ubuntu:/mnt$
```

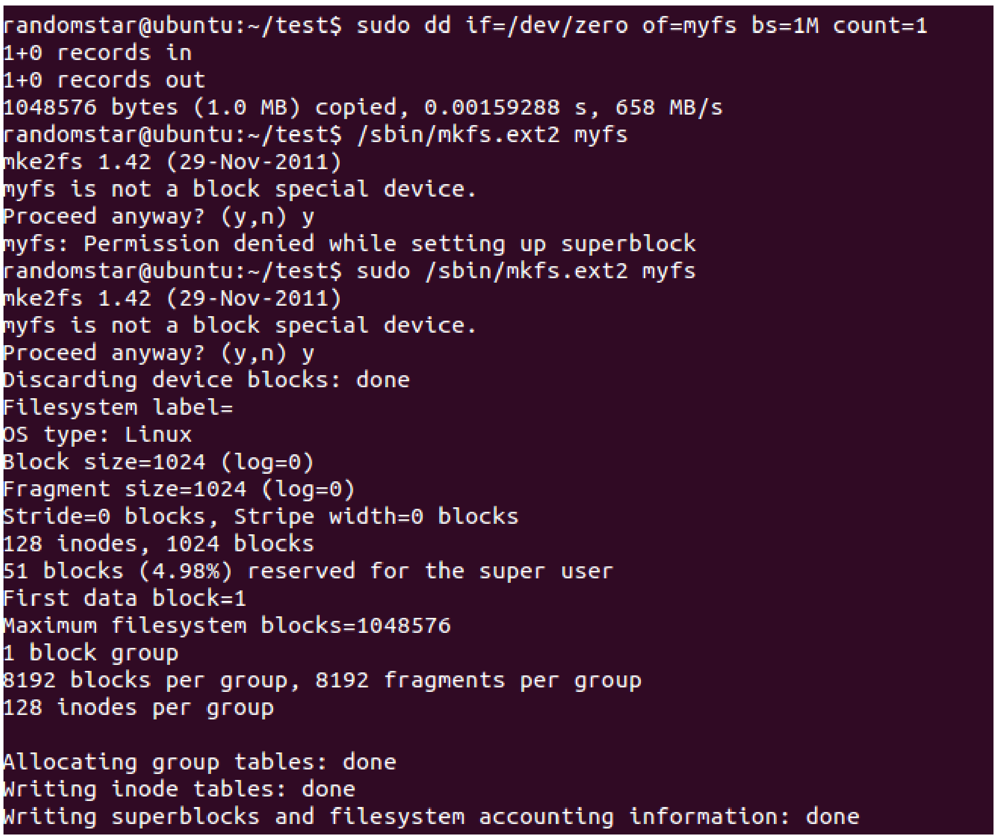
至此实验的五个大步骤全部顺利完成。

2.3 测试程序运行的结果截图

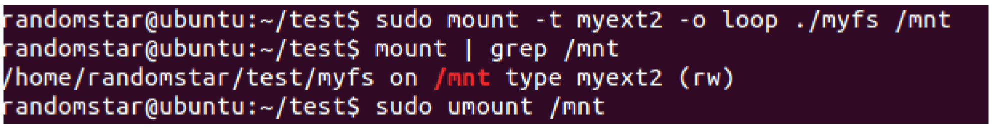
本次实验中主要有如下几个关键的测试点，其实上面的截图中已经给出，但是这里我还是把它们再贴一次：

2.3.1 第一步测试

- 尝试挂载文件系统并查看挂载的情况：



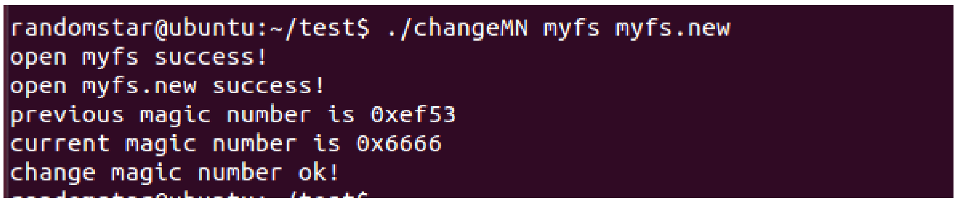
```
randomstar@ubuntu:~/test$ sudo dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00159288 s, 658 MB/s
randomstar@ubuntu:~/test$ /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
myfs is not a block special device.
Proceed anyway? (y,n) y
myfs: Permission denied while setting up superblock
randomstar@ubuntu:~/test$ sudo /sbin/mkfs.ext2 myfs
mke2fs 1.42 (29-Nov-2011)
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
```



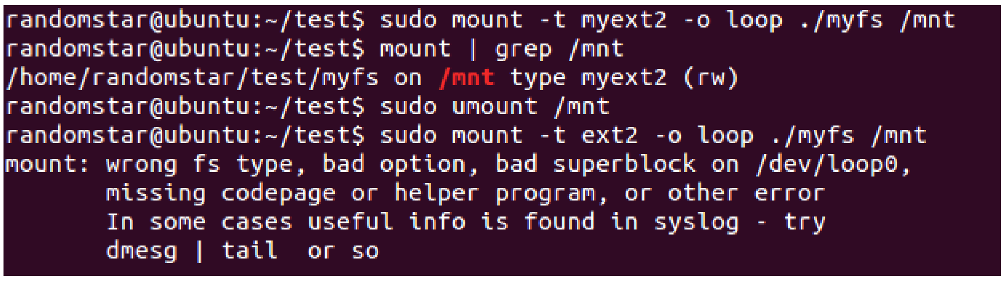
```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ mount | grep /mnt
/home/randomstar/test/myfs on /mnt type myext2 (rw)
randomstar@ubuntu:~/test$ sudo umount /mnt
```

2.3.2 第二步测试

- changeMN 运行时的情况



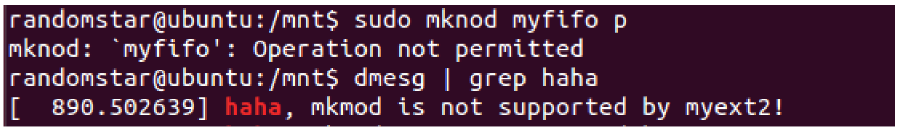
```
randomstar@ubuntu:~/test$ ./changeMN myfs myfs.new
open myfs success!
open myfs.new success!
previous magic number is 0xef53
current magic number is 0x6666
change magic number ok!
```



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

2.3.3 第三步测试

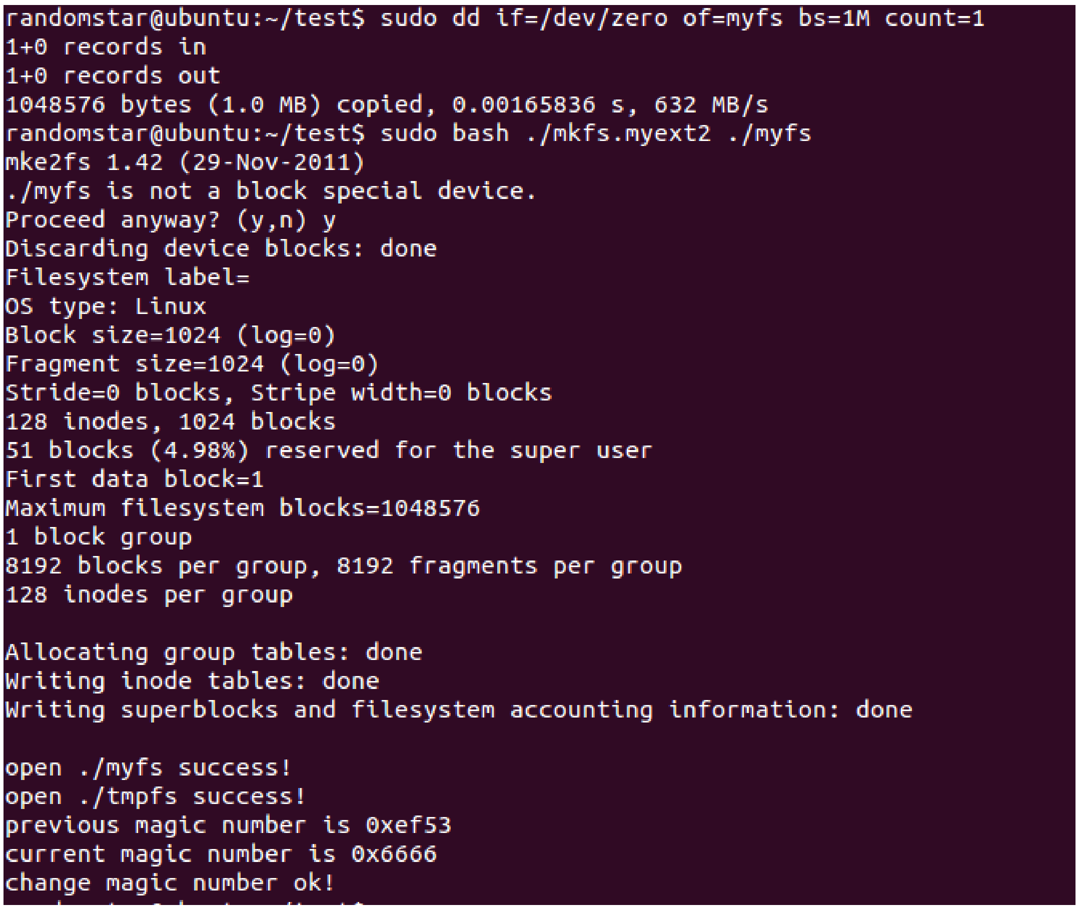
● 测试使用 mknod 指令时候的情况



```
randomstar@ubuntu:/mnt$ sudo mknod myfifo p
mknod: `myfifo': Operation not permitted
randomstar@ubuntu:/mnt$ dmesg | grep haha
[  890.502639] haha, mkmod is not supported by myext2!
```

2.3.4 第四步测试

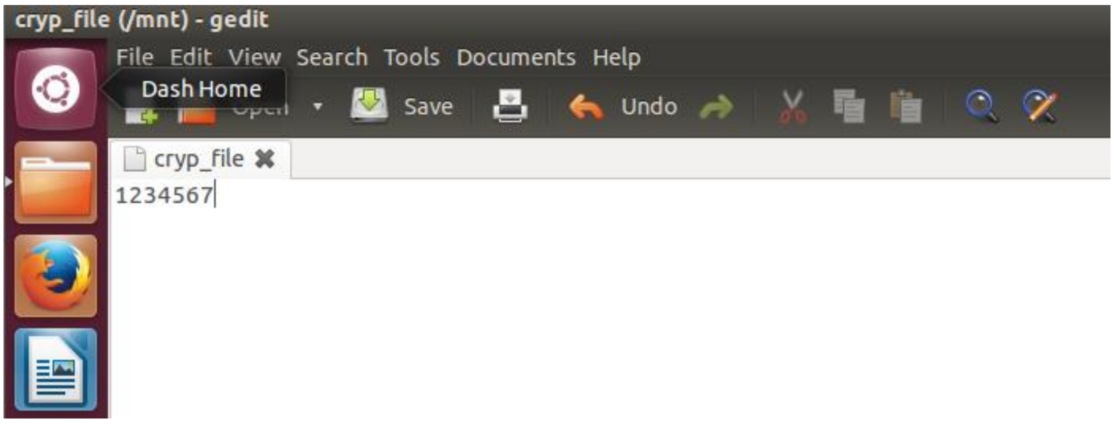
● 文件系统创建工具运行时的情况



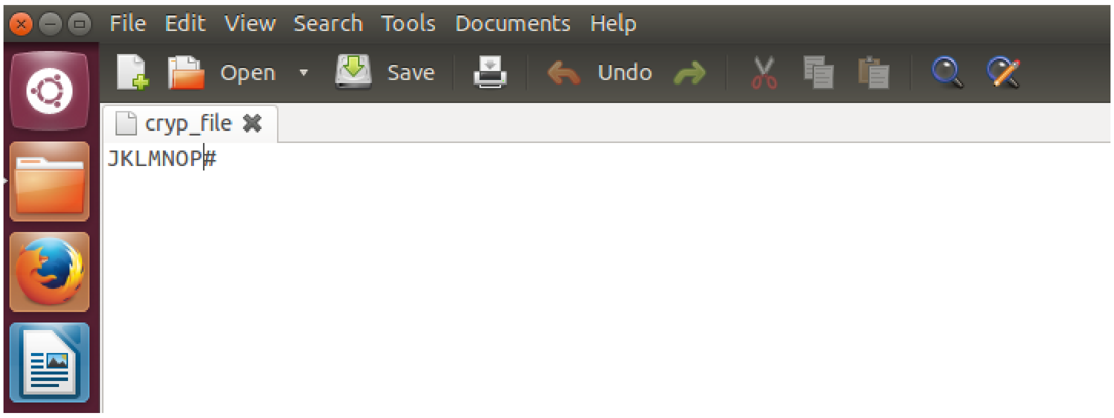
```
randomstar@ubuntu:~/test$ sudo dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00165836 s, 632 MB/s
randomstar@ubuntu:~/test$ sudo bash ./mkfs.myext2 ./myfs
mke2fs 1.42 (29-Nov-2011)
./myfs is not a block special device.
Proceed anyway? (y,n) 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!
```

2.3.5 第五步测试

● 主要测试点是两种情况下看到的 1234567 分别是明文和密文

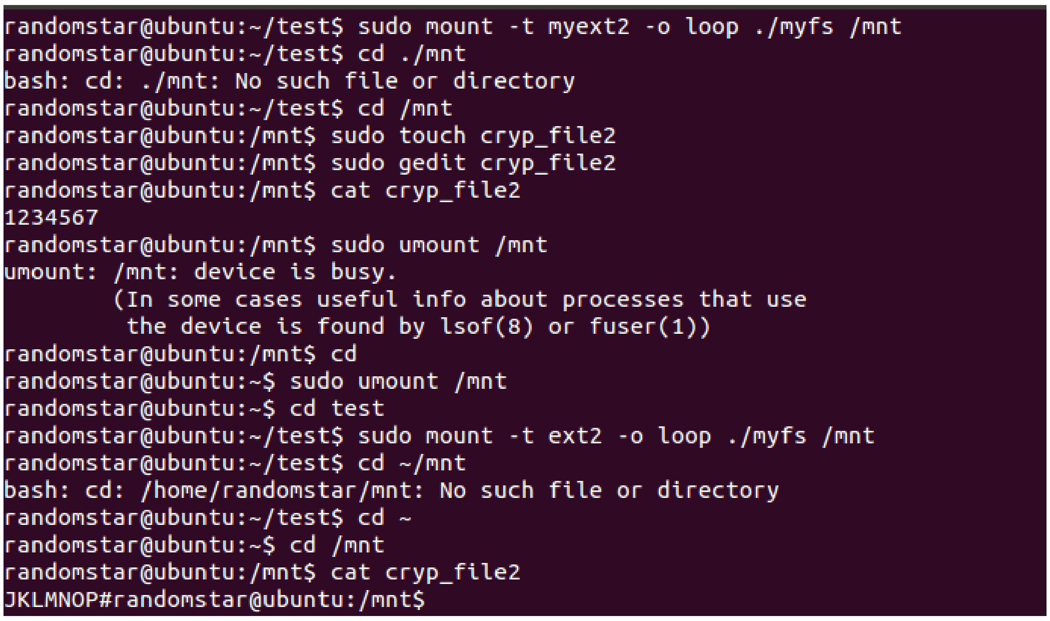


1234567



JKLMNOP#

● 改回 magic number 之后重新编译、安装 myext2 之后的一系列操作：



```
randomstar@ubuntu:~/test$ sudo mount -t myext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ cd ./mnt
bash: cd: ./mnt: No such file or directory
randomstar@ubuntu:~/test$ cd /mnt
randomstar@ubuntu:/mnt$ sudo touch cryp_file2
randomstar@ubuntu:/mnt$ sudo gedit cryp_file2
randomstar@ubuntu:/mnt$ cat cryp_file2
1234567
randomstar@ubuntu:/mnt$ sudo umount /mnt
umount: /mnt: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
randomstar@ubuntu:/mnt$ cd
randomstar@ubuntu:~$ sudo umount /mnt
randomstar@ubuntu:~$ cd test
randomstar@ubuntu:~/test$ sudo mount -t ext2 -o loop ./myfs /mnt
randomstar@ubuntu:~/test$ cd ~/mnt
bash: cd: /home/randomstar/mnt: No such file or directory
randomstar@ubuntu:~/test$ cd ~
randomstar@ubuntu:~$ cd /mnt
randomstar@ubuntu:/mnt$ cat cryp_file2
JKLMNOP#randomstar@ubuntu:/mnt$
```

2.4 实验结果分析

- 本次实验中的第五步使用 cp 命令将写了 1234567 的文件拷贝到桌面之后没有被加密，但是使用文件管理器进行拷贝就被加密了，我初步分析得出原因是：cp 命令是先 read 再 write 的，因此经过了一轮加密解密，所以不影响原本内容，而文件管理器是直接把文件中的二进制内容 write 到桌面中而没有进行读，因此此时的 1234567 是加密状态的
- 即时使用了 ext2 文件系统的 magic number，myext2 文件系统中创建的文件都是也依然是加密文件，因为一开始写入的 1234567 室基于 myext2 的 write 函数的，因此被加密了，当 myext2 的 magic number 和 ext2 一致的时候使用 myext 将 mnt 重新重新挂载成 ext2 文件系统并不会报错，而 mnt 编程 ext2 格式的时候里面的 1234567 被当作未加密的数据处理，因此读出来的结果依然是 1234567 加密后的内容
- 综合上述截图和实验步骤，可以认为本次实验的五个步骤都顺利完成。

2.5 实验源代码

- 本次实验修改的地方比较多，我主要放一下 file.c 中修改的加密解密部分的代码

```
// add by zyc
ssize_t new_sync_write_crypt(struct file *filp, const char
__user *buf, size_t len, loff_t *ppos)
{
   int i;
   char* mybuf =
(char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
   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\n", len);
   return do_sync_write(filp, buf, len, ppos);
}

// add by zyc
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\n", len);
   return ret;
}

/*
 * We have mostly NULL's here: the current defaults are ok
for
 * the myext2 filesystem.
 */
const struct file_operations myext2_file_operations = {
   .llseek       = generic_file_llseek,
   .read      = new_sync_read_crypt,
   .write     = new_sync_write_crypt,
   .aio_read  = generic_file_aio_read,
   .aio_write = generic_file_aio_write,
   .unlocked_ioctl = myext2_ioctl,
#ifdef CONFIG_COMPAT
   .compat_ioctl  = myext2_compat_ioctl,
#endif
   .mmap      = generic_file_mmap,
   .open      = dquot_file_open,
   .release    = myext2_release_file,
   .fsync     = myext2_fsync,
   .splice_read    = generic_file_splice_read,
   .splice_write  = generic_file_splice_write,
};
```

三、讨论与心得

本次实验我遇到的问题和心得如下：

- 内核版本选择：一开始想用高版本的内核来完成实验但是之前的 Ubuntu16.04 因为更换过内核，一开始就出现了这样那样的问题，因此更换了更古老的 Ubuntu12.04 版本，使

用其自带的 3.13.0 内核进行实验，此外如果下载的内核源码和当前系统的内核版本不匹配，也会出现编译错误

- 在第五步添加加密和解密函数的时候，要调用到原本的明文读写函数，这里不同版本的明文读写函数名是不一样的，因此需要对代码进行适当的修改，而文件的数据需要先从用户态拷贝到内核态，操作完之后再拷贝回去，不能直接在内核态进行操作，否则编译会出错
- 内核模块如果在一次编译之后没有修改任何相关文件中的内容，直接 make 并不会顺利再次编译，可以删除目录下所有的.o 文件然后再次编译，此外内核模块的 umount 不能在挂载的目录下进行，否则会提示当前目录 busy，dd 命令在 myext2 内核模块重新编译并挂载之后需要重新执行一次。
- 本次试验内容丰富，并且没有编译内核的环节，因此相比上次实验减少了很多等待时间，但是我在做完实验之后对文件系统有了更深的理解，知道了如何添加一个自己的文件系统，并对其进行一系列的改造。但是不幸的是我在完成实验之后不小心把保存的截图都删了，当发现这件事情的时候为时已晚，因此只能重新做一遍。
- 本次实验中虽然我遇到了很多困难，有的问题实验指导书也没能给我正确的解答，但最后在同学的提示下完成了整个实验，也希望实验指导书可以更新一下，与时俱进。
- 至此我操作系统的三个实验已经全部完成了，这一系列的内核实验给我最大的感受就是非常考验自己的心态和忍耐力，大部分实验按照实验指导书上就可以做下来，但是由于内核版本的细微差异和自己的不小心就可能就会导致实验过程中经常需要克服一些原本可能并不存在的困难，但总的来说经历了三个实验自己还是收获了不少的。