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：张佳瑶 学号：3170103240

电子邮件地址：1531077171@qq.com

实验日期：2019 年 12 月 17 日

一、实验环境

处理器：Intel® Core™ i7-6700HQ CPU @ 2.60GHz

Windows10

Linux version 4.15.18 (zjy@ubuntu) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.11)) #5 SMP Sun Dec 15 00:27:02 PST 2019

二、实验内容和结果及分析

实验设计思路：

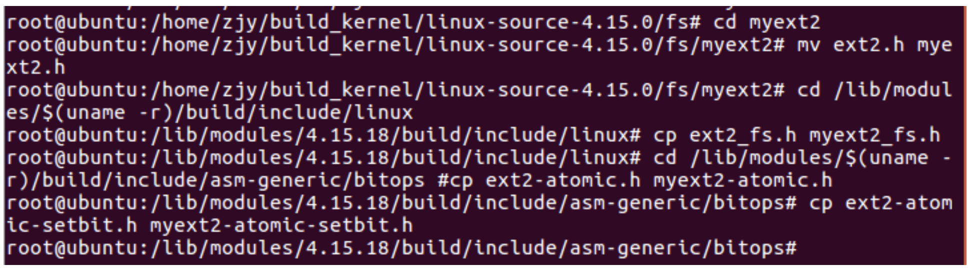
获得实验二编译好的内核源码。在此基础上，修改 Linux 原有的文件系统中的代码，添加自己设计的加密读写程序，编译好后挂载到/mnt 上，最终实现添加一个加密文件系统。

实验步骤及截图：

1. 添加一个类似 ext2 的文件系统 myext2

按照 Linux 源代码的组织结构，把 myext2 文件系统的源代码存放到 fs/myext2 下，头文件放到 include/linux 下。在 Linux 的内核代码文件下，在 Linux 的 shell 下，执行如下操作:

```
#cd fs
#cp –R ext2 myext2
#cd /usr/src/linux/fs/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
```



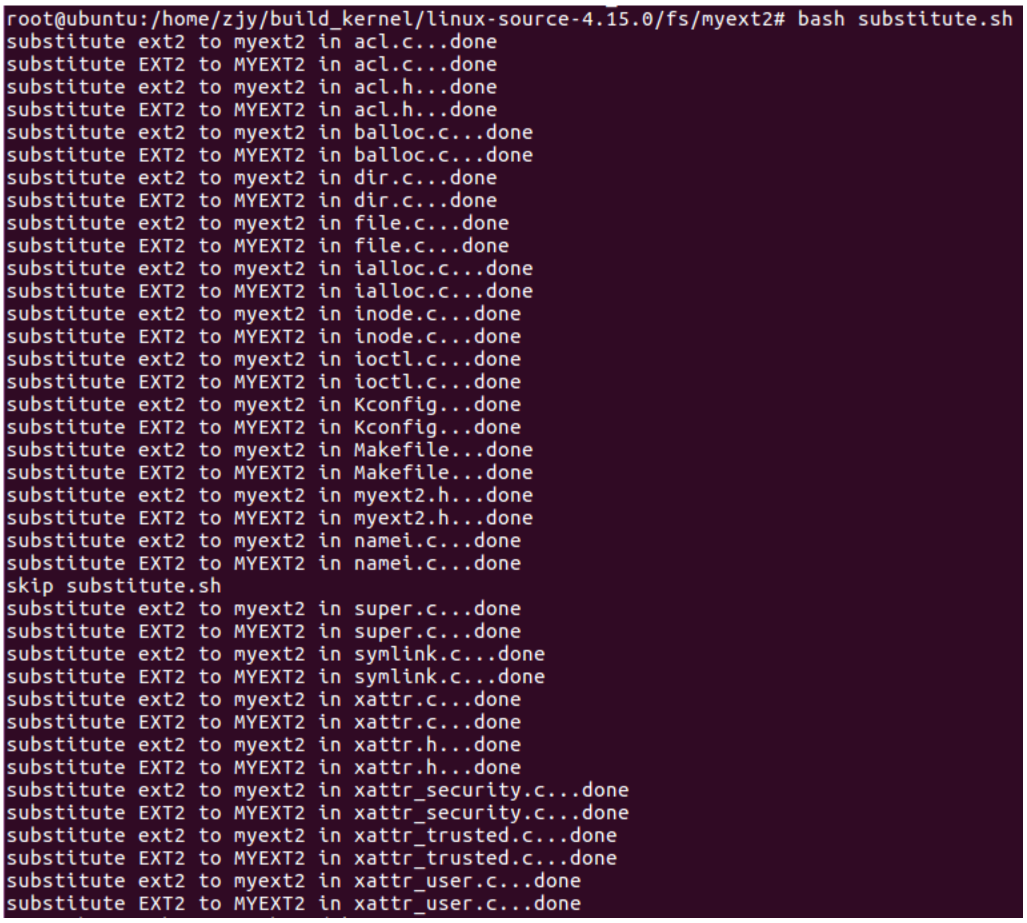
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs# cd myext2
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mv ext2.h mye
xt2.h
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# cd /lib/modul
es/$(uname -r)/build/include/linux
root@ubuntu:/lib/modules/4.15.18/build/include/linux# cp ext2_fs.h myext2_fs.h
root@ubuntu:/lib/modules/4.15.18/build/include/linux# cd /lib/modules/$(uname -
r)/build/include/asm-generic/bitops #cp ext2-atomic.h myext2-atomic.h
root@ubuntu:/lib/modules/4.15.18/build/include/asm-generic/bitops# cp ext2-atom
ic-setbit.h myext2-atomic-setbit.h
root@ubuntu:/lib/modules/4.15.18/build/include/asm-generic/bitops#
```

下面开始克隆文件系统的第二步:修改上面添加的文件的内容。为了简单起见，做了一个最简单的替换:将原来“EXT2”替换成“MYEXT2”,将原来的“ext2” 替换成“myext2”。对于 fs/myext2 下面文件中字符串的替换，也可以使用下面的脚本: #!/bin/bash

```
#!/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
```

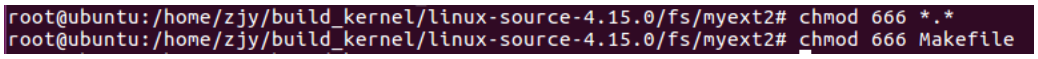
把这个脚本命名为 substitute.sh，放在 fs/myext2 下面，加上可执行权限，运行之后就可以把当前目录里所有文件里面的“ext2”和“EXT2”都替换成对应 的“myext2”和“MYEXT2”。

执行脚本：



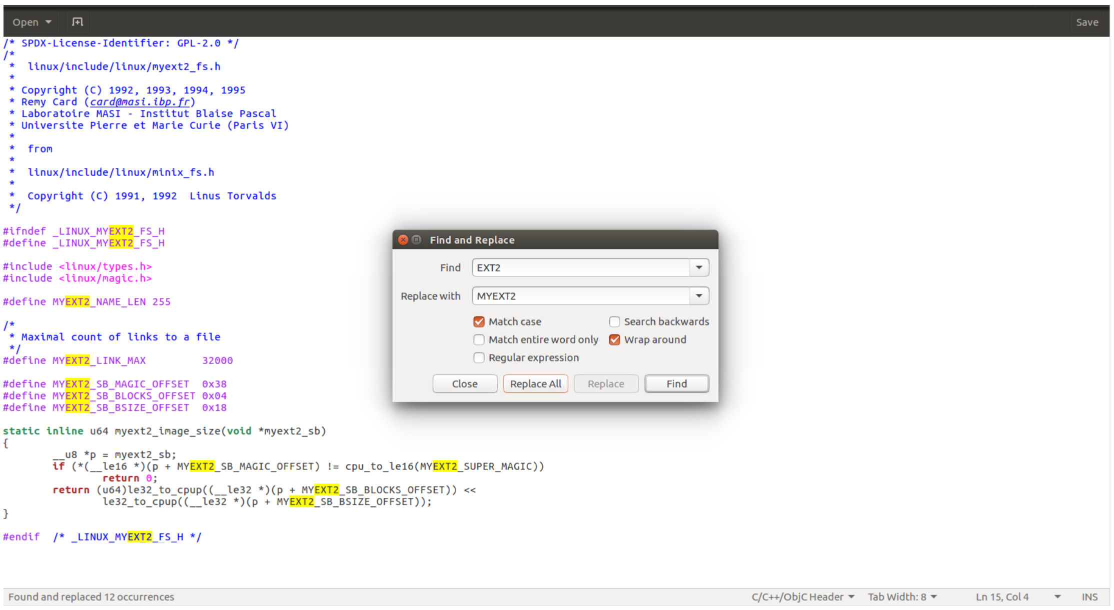
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# bash substitute.sh
substitute ext2 to myext2 in acl.c...done
substitute EXT2 to MYEXT2 in acl.c...done
substitute ext2 to myext2 in acl.h...done
substitute EXT2 to MYEXT2 in acl.h...done
substitute ext2 to myext2 in balloc.c...done
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 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
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
```

执行完成之后发现文件均变成只读文件，于是使用进行权限修改操作：



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# chmod 666 *.*
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# chmod 666 Makefile
```

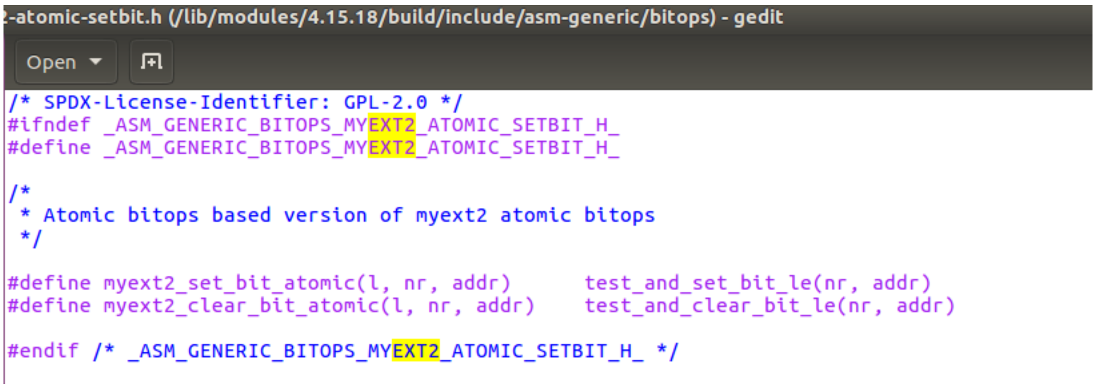
用编辑器的替换功能，把/lib/modules/$(uname -r)/build/include/linux/myext2_fs.h，和 /lib/modules/$(uname -r)/build/include/asm-generic/bitops/ 下的 myext2-atomic.h 与 myext2-atomic-setbit.h 文件中的“ext2”、“EXT2”分别替换成“myext2”、“MYEXT2”。



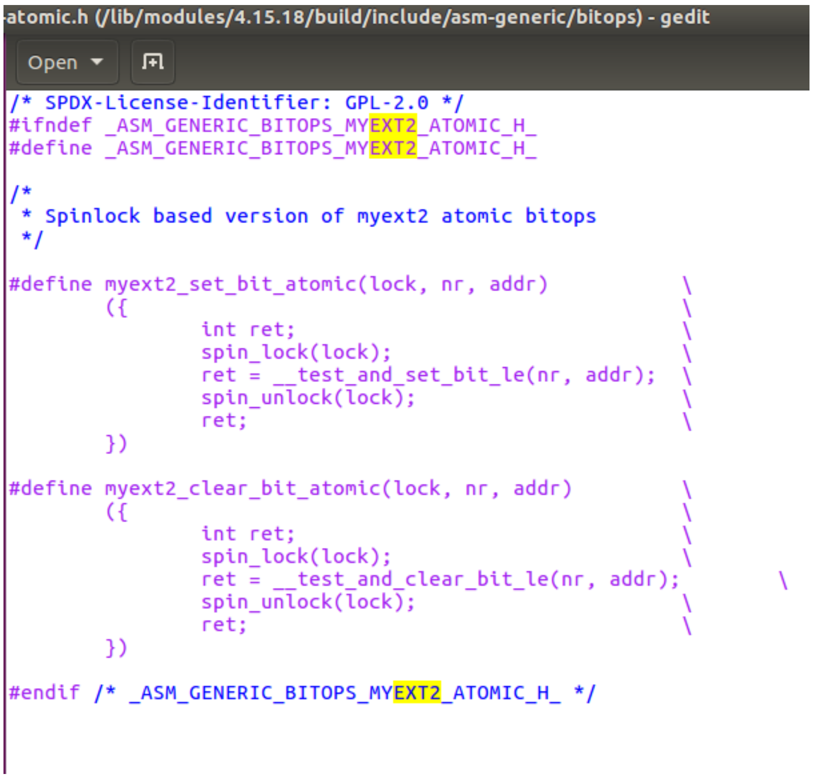
```
/* SPDX-License-Identifier: GPL-2.0 */
/*
 *  linux/include/linux/myext2_fs.h
 *
 * Copyright (C) 1992, 1993, 1994, 1995
 * Remy Card (card@masi.ibp.fr)
 * Laboratoire MASI - Institut Blaise Pascal
 * Universite Pierre et Marie Curie (Paris VI)
 *
 *  from
 *
 *  linux/include/linux/minix_fs.h
 *
 *  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));
}

#endif  /* _LINUX_MYEXT2_FS_H */
```

Find and Replace
Find: EXT2
Replace with: MYEXT2
[✓] Match case
[ ] Search backwards
[ ] Match entire word only
[✓] Wrap around
[ ] Regular expression

Found and replaced 12 occurrences

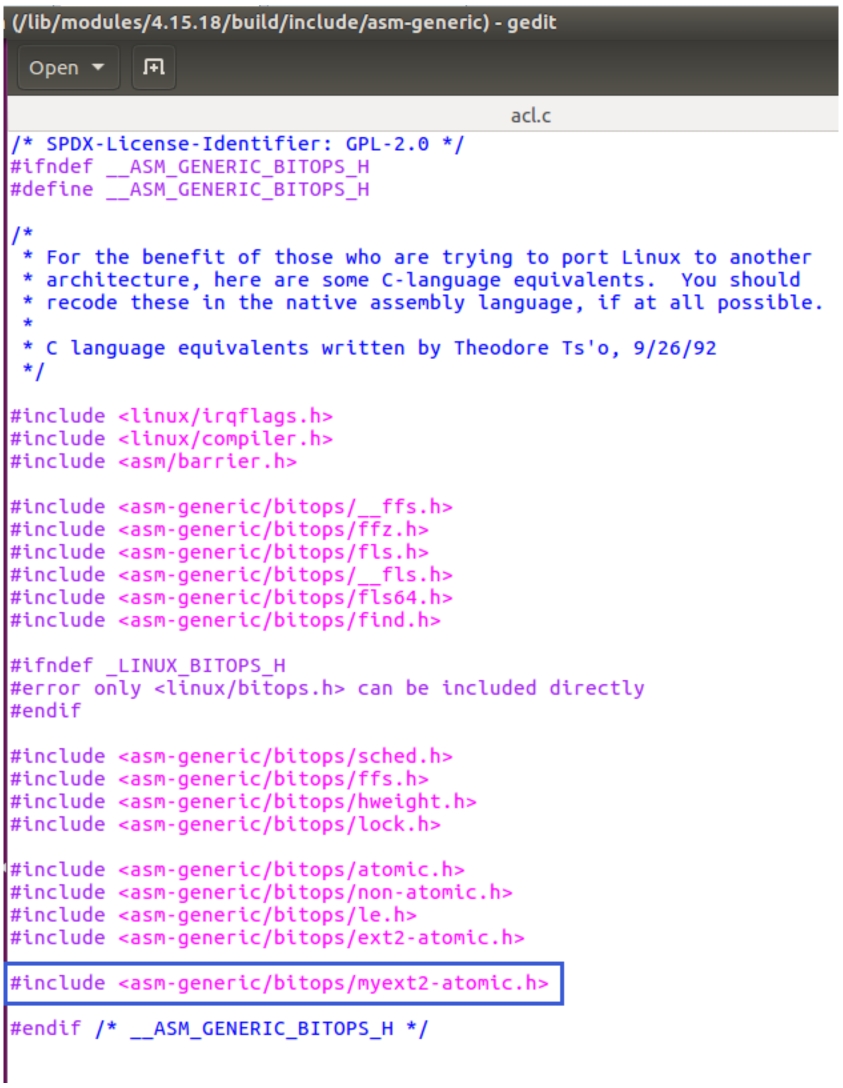

-atomic-setbit.h (/lib/modules/4.15.18/build/include/asm-generic/bitops) - gedit
Open

```
/* SPDX-License-Identifier: GPL-2.0 */
#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_ */
```


atomic.h (/lib/modules/4.15.18/build/include/asm-generic/bitops) - gedit
Open

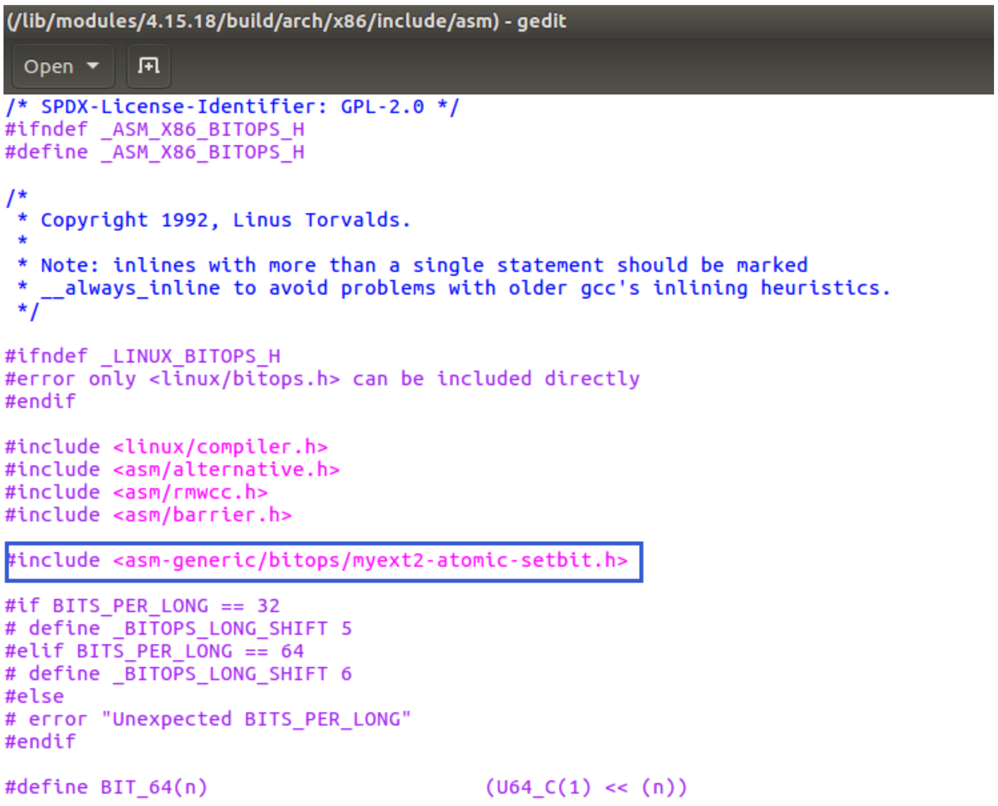
```
/* SPDX-License-Identifier: GPL-2.0 */
#ifndef _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_
#define _ASM_GENERIC_BITOPS_MYEXT2_ATOMIC_H_

/*
 * 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_ */
```

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



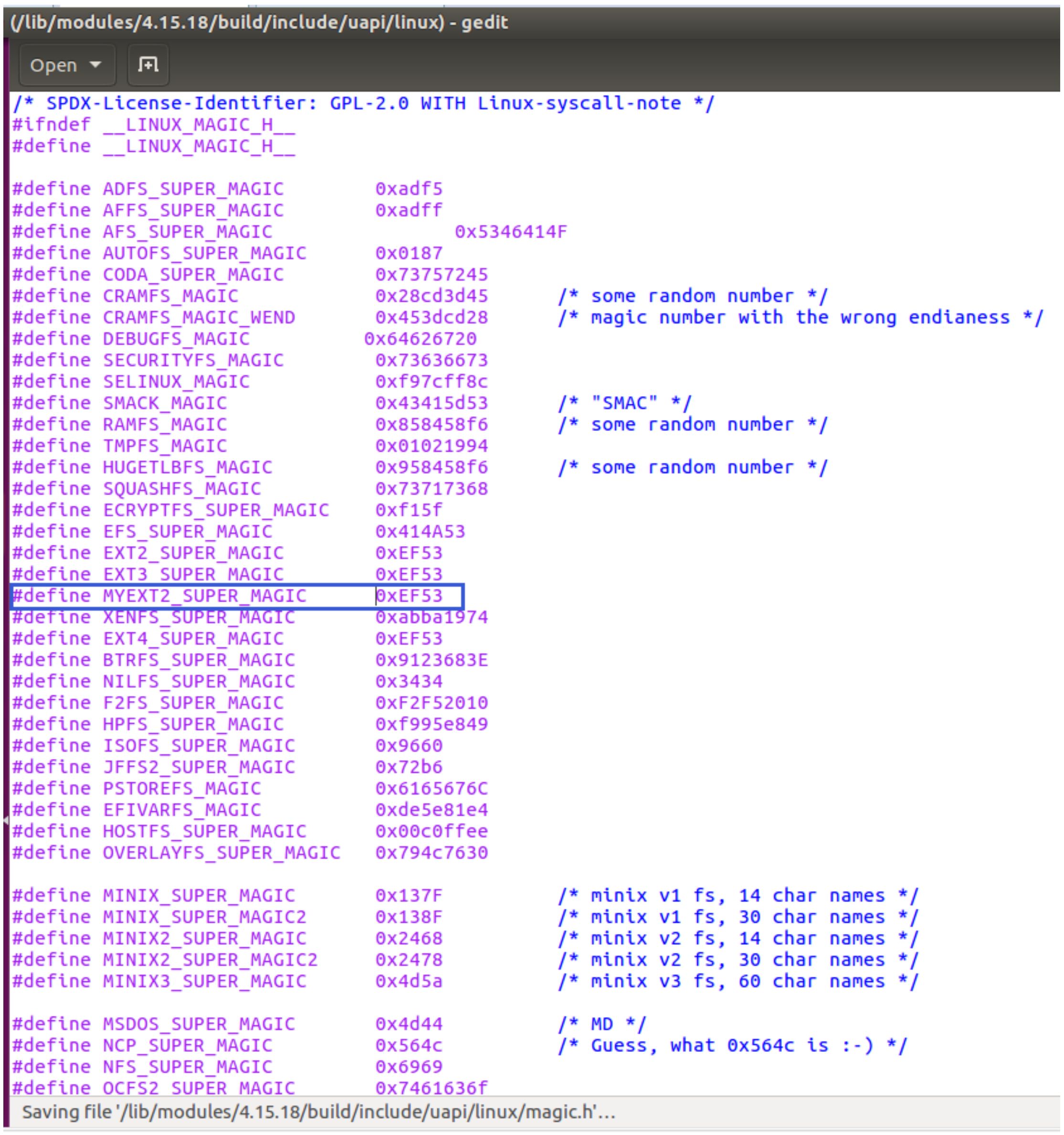
```
/* SPDX-License-Identifier: GPL-2.0 */
#ifndef __ASM_GENERIC_BITOPS_H
#define __ASM_GENERIC_BITOPS_H

/*
 * For the benefit of those who are trying to port Linux to another
 * architecture, here are some C-language equivalents.  You should
 * recode these in the native assembly language, if at all possible.
 *
 * C language equivalents written by Theodore Ts'o, 9/26/92
 */

#include <linux/irqflags.h>
#include <linux/compiler.h>
#include <asm/barrier.h>

#include <asm-generic/bitops/__ffs.h>
#include <asm-generic/bitops/ffz.h>
#include <asm-generic/bitops/fls.h>
#include <asm-generic/bitops/__fls.h>
#include <asm-generic/bitops/fls64.h>
#include <asm-generic/bitops/find.h>

#ifndef _LINUX_BITOPS_H
#error only <linux/bitops.h> can be included directly
#endif

#include <asm-generic/bitops/sched.h>
#include <asm-generic/bitops/ffs.h>
#include <asm-generic/bitops/hweight.h>
#include <asm-generic/bitops/lock.h>

#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>

#include <asm-generic/bitops/myext2-atomic.h>

#endif /* __ASM_GENERIC_BITOPS_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>



```
/* SPDX-License-Identifier: GPL-2.0 */
#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

#include <linux/compiler.h>
#include <asm/alternative.h>
#include <asm/rmwcc.h>
#include <asm/barrier.h>

#include <asm-generic/bitops/myext2-atomic-setbit.h>

#if BITS_PER_LONG == 32
# define _BITOPS_LONG_SHIFT 5
#elif BITS_PER_LONG == 64
# define _BITOPS_LONG_SHIFT 6
#else
# error "Unexpected BITS_PER_LONG"
#endif

#define BIT_64(n)                       (U64_C(1) << (n))
```

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

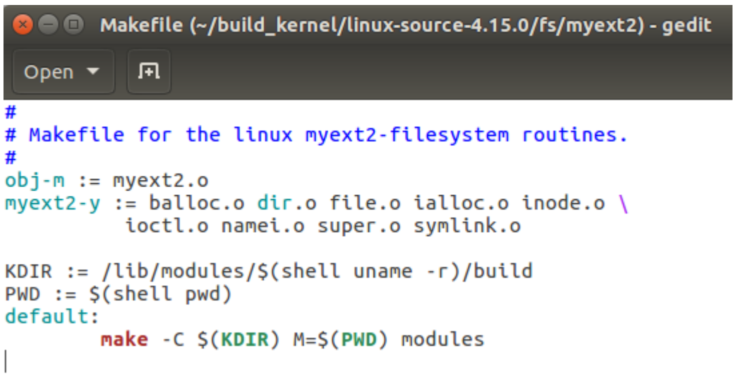


```
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

#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
#define MYEXT2_SUPER_MAGIC      0xEF53
#define XENFS_SUPER_MAGIC       0xabba1974
#define EXT4_SUPER_MAGIC        0xEF53
#define BTRFS_SUPER_MAGIC       0x9123683E
#define NILFS_SUPER_MAGIC       0x3434
#define F2FS_SUPER_MAGIC        0xF2F52010
#define HPFS_SUPER_MAGIC        0xf995e849
#define ISOFS_SUPER_MAGIC       0x9660
#define JFFS2_SUPER_MAGIC       0x72b6
#define PSTOREFS_MAGIC          0x6165676C
#define EFIVARFS_MAGIC          0xde5e81e4
#define HOSTFS_SUPER_MAGIC      0x00c0ffee
#define OVERLAYFS_SUPER_MAGIC   0x794c7630

#define MINIX_SUPER_MAGIC       0x137F          /* minix v1 fs, 14 char names */
#define MINIX_SUPER_MAGIC2      0x138F          /* minix v1 fs, 30 char names */
#define MINIX2_SUPER_MAGIC      0x2468          /* minix v2 fs, 14 char names */
#define MINIX2_SUPER_MAGIC2     0x2478          /* minix v2 fs, 30 char names */
#define MINIX3_SUPER_MAGIC      0x4d5a          /* minix v3 fs, 60 char names */

#define MSDOS_SUPER_MAGIC       0x4d44          /* MD */
#define NCP_SUPER_MAGIC         0x564c          /* Guess, what 0x564c is :-) */
#define NFS_SUPER_MAGIC         0x6969
#define OCFS2_SUPER_MAGIC       0x7461636f
```

Saving file '/lib/modules/4.15.18/build/include/uapi/linux/magic.h'...

源代码的修改工作到此结束。接下来就是第三步工作——把 myext2 编译源成内核模块。要编译内核模块，首先要生成一个 Makefile 文件。我们可以修改 myext2/fm 文件，修改后的 Makefile 文件如下：

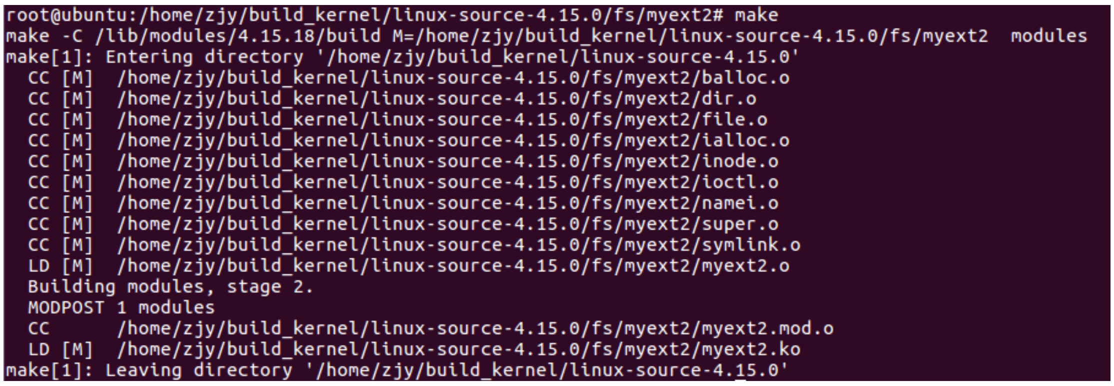
```
#
# Makefile for the linux myext2-filesystem routines.
#
obj-m := myext2.o
myext2-y := balloc.o dir.o file.o ialloc.o inode.o \
        ioctl.o namei.o super.o symlink.o

KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
    make -C $(KDIR) M=$(PWD) modules
```


Makefile (~/build_kernel/linux-source-4.15.0/fs/myext2) - gedit

```
#
# 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 目录下执行命令：

#make



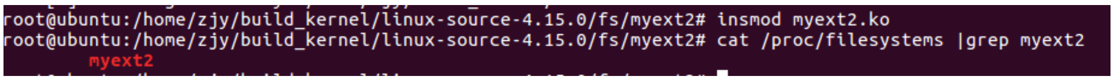
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# make
make -C /lib/modules/4.15.18/build M=/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2  modules
make[1]: Entering directory '/home/zjy/build_kernel/linux-source-4.15.0'
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/balloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/dir.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ialloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/inode.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ioctl.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/super.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/symlink.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.mod.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/zjy/build_kernel/linux-source-4.15.0'
```

使用 insmod 命令加载模块：

#insmod myext2.ko

查看一下 myext2 文件系统是否加载成功：

#cat /proc/filesystems |grep myext2



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# insmod myext2.ko
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# cat /proc/filesystems |grep myext2
        myext2
```

确认 myext2 文件系统加载成功后，可以对添加的 myext2 文件系统进行测试了，输入命令 cd 先把当前目录设置成主目录。

对添加的 myext2 文件系统测试命令如下：

#dd if=/dev/zero of=myfs bs=1M count=1

#/sbin/mkfs.ext2 myfs

#mount -t myext2 -o loop ./myfs /mnt

#mount

……

…… on /mnt type myext2 (rw)

#umount /mnt

#mount -t ext2 -o loop ./myfs /mnt

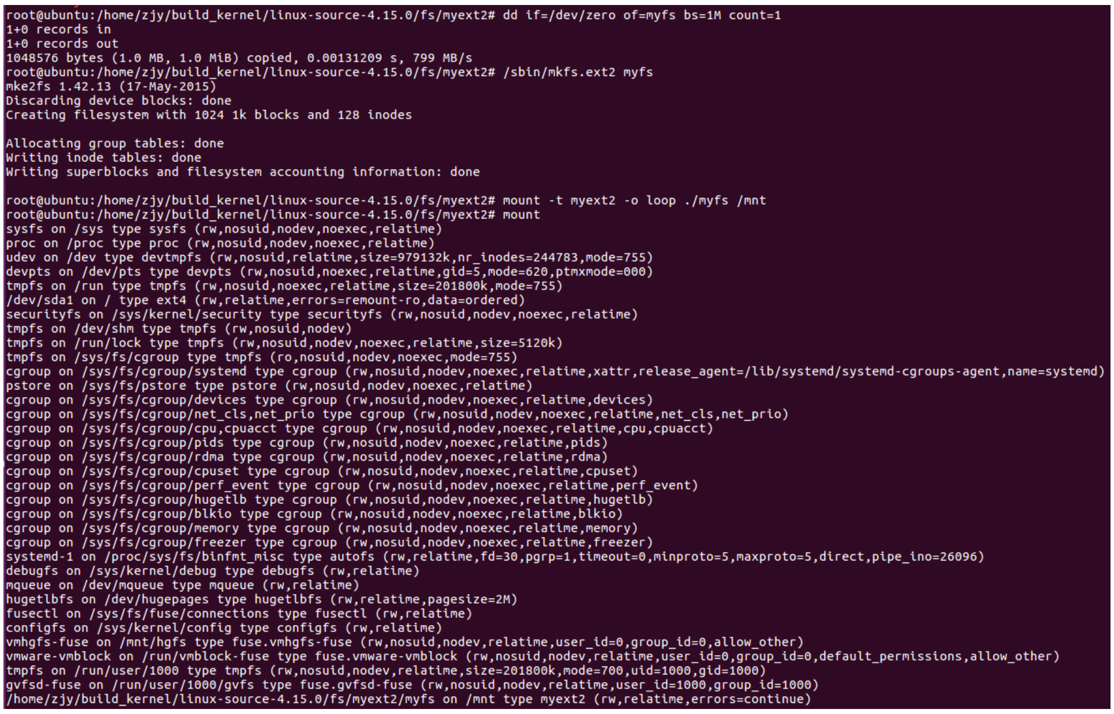
#mount

……

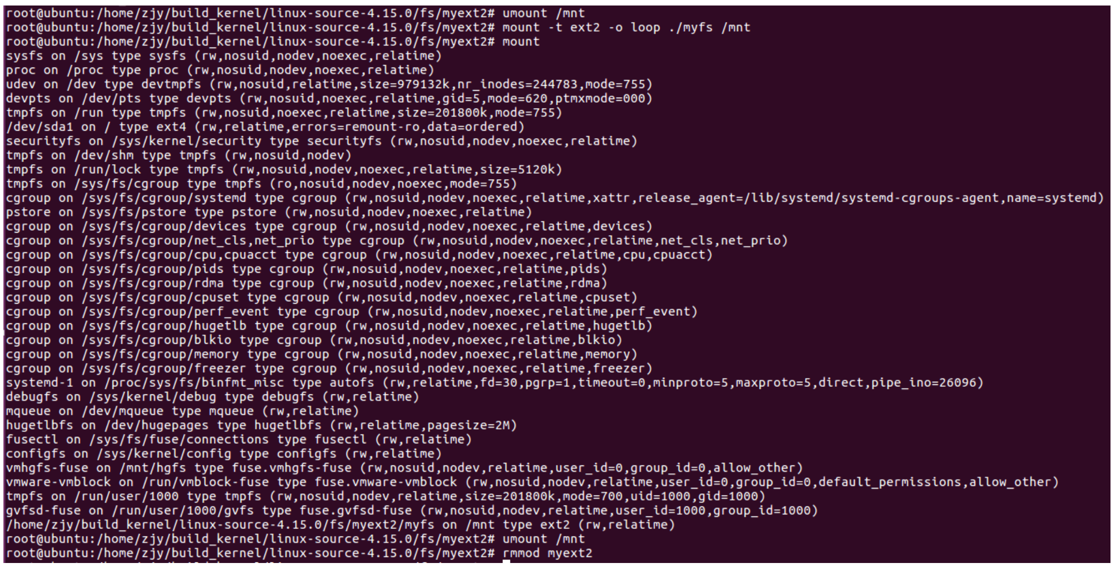
…… on /mnt type ext2 (rw)

#umount /mnt

#rmmod myext2　/*卸载模块*/



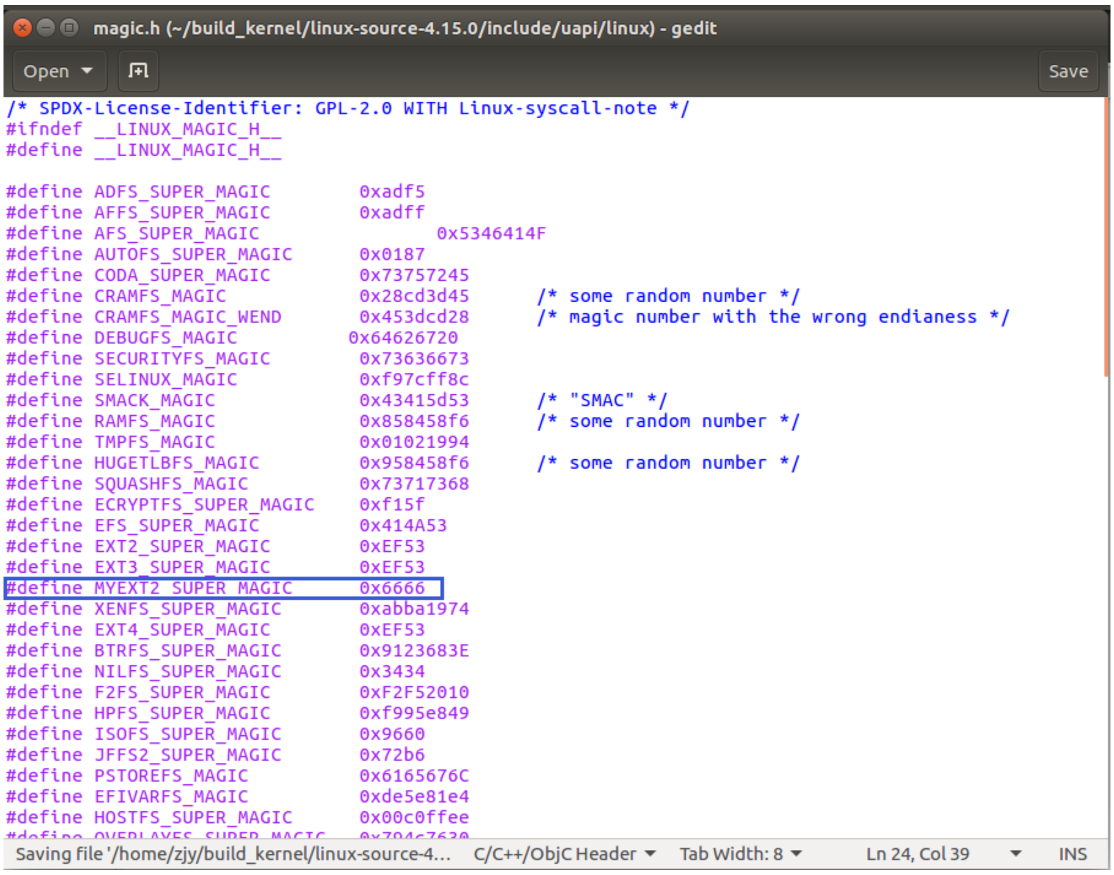
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00131209 s, 799 MB/s
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# /sbin/mkfs.ext2 myfs
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

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

root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=979132k,nr_inodes=244783,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=201800k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=30,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=26096)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmhgfs-fuse on /mnt/hgfs type fuse.vmhgfs-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201800k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myfs on /mnt type myext2 (rw,relatime,errors=continue)
```



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# umount /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t ext2 -o loop ./myfs /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=979132k,nr_inodes=244783,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=201800k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=30,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=26096)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmhgfs-fuse on /mnt/hgfs type fuse.vmhgfs-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201800k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myfs on /mnt type ext2 (rw,relatime)
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# umount /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# rmmod myext2
```

2.　修改 myext2 的 magic number

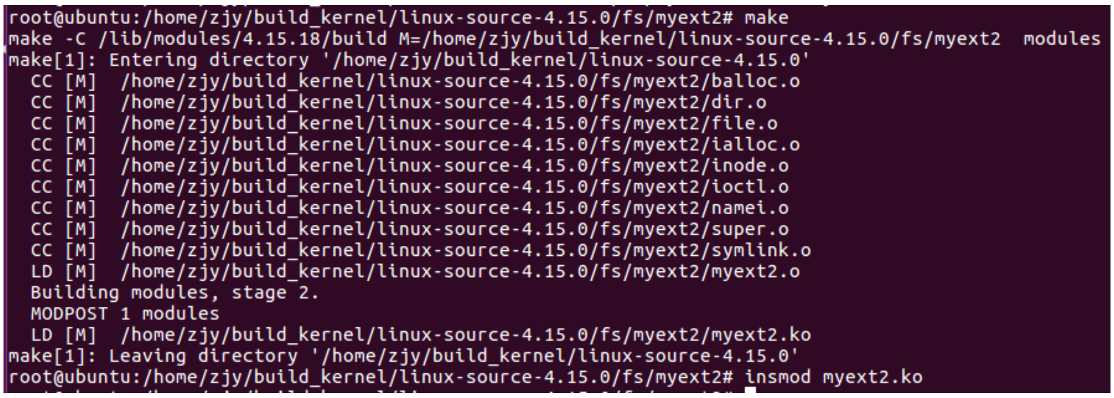
找到 myext2 的 magic number，并将其改为 0x6666



magic.h (~/build_kernel/linux-source-4.15.0/include/uapi/linux) - gedit
Open
Save

```
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

#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
#define MYEXT2_SUPER_MAGIC      0x6666
#define XENFS_SUPER_MAGIC       0xabba1974
#define EXT4_SUPER_MAGIC        0xEF53
#define BTRFS_SUPER_MAGIC       0x9123683E
#define NILFS_SUPER_MAGIC       0x3434
#define F2FS_SUPER_MAGIC        0xF2F52010
#define HPFS_SUPER_MAGIC        0xf995e849
#define ISOFS_SUPER_MAGIC       0x9660
#define JFFS2_SUPER_MAGIC       0x72b6
#define PSTOREFS_MAGIC          0x6165676C
#define EFIVARFS_MAGIC          0xde5e81e4
#define HOSTFS_SUPER_MAGIC      0x00c0ffee
```

再用 make 重新编译内核模块，使用命令 insmod 安装编译好的 myext2.ko 内核模块。



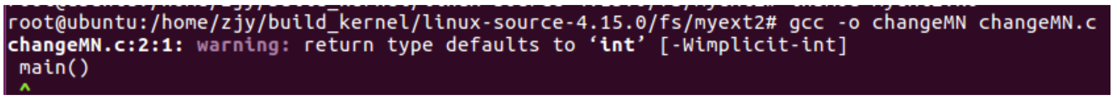
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# make
make -C /lib/modules/4.15.18/build M=/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2  modules
make[1]: Entering directory '/home/zjy/build_kernel/linux-source-4.15.0'
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/balloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/dir.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ialloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/inode.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ioctl.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/super.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/symlink.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/zjy/build_kernel/linux-source-4.15.0'
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# insmod myext2.ko
```

编写 changeMN.c:

```
#include <stdio.h>
main()
{
        int ret;
        FILE *fp_read;
        FILE *fp_write;
        unsigned char buf[2048];

        fp_read=fopen("./myfs","rb");
```

```
if(fp_read == NULL)
{
        printf("open myfs failed!\n");
        return 1;
}

fp_write=fopen("./fs.new","wb");

if(fp_write==NULL)
{
        printf("open fs.new failed!\n");
        return 2;
}

ret=fread(buf,sizeof(unsigned char),2048,fp_read);

printf("previous magic number is 0x%x%x\n",buf[0x438],buf[0x439]);

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

fwrite(buf,sizeof(unsigned char),2048,fp_write);

printf("current magic number is 0x%x%x\n",buf[0x438],buf[0x439]);

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

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

fclose(fp_read);
fclose(fp_write);
```

```
    return 0;
}
```



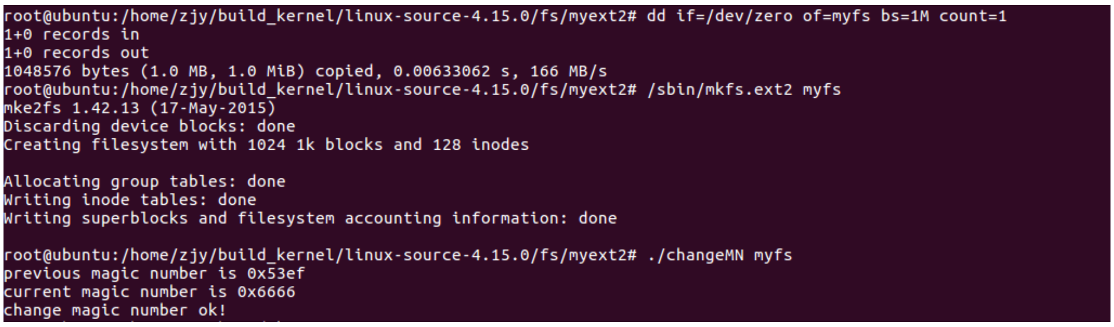
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# gcc -o changeMN changeMN.c
changeMN.c:2:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main()
 ^
```

下面我们开始测试:

#dd if=/dev/zero of=myfs bs=1M count=1

#/sbin/mkfs.ext2 myfs

#./changeMN myfs



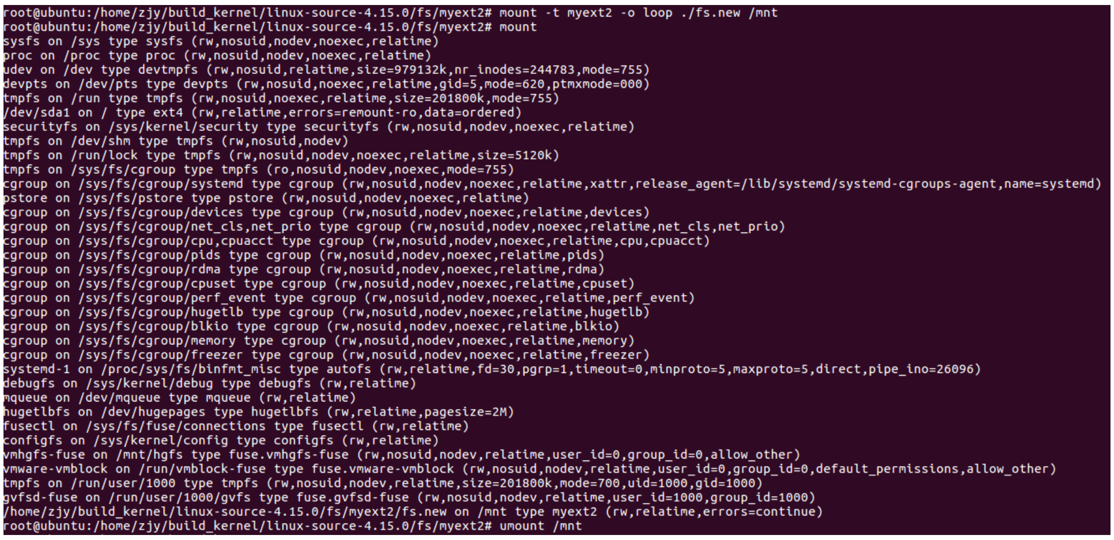
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00633062 s, 166 MB/s
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# /sbin/mkfs.ext2 myfs
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

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

root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
```

#mount -t myext2 -o loop ./fs.new /mnt

#mount

...... on /mnt type myext2 (rw)

#sudo umount /mnt

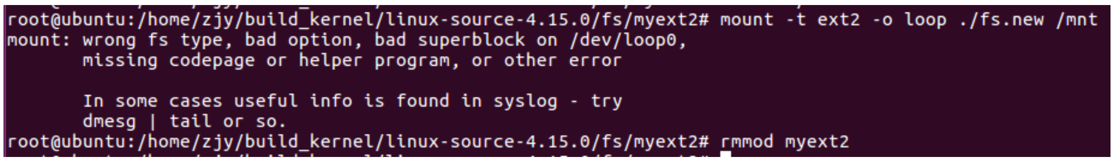


```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=979132k,nr_inodes=244783,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=201800k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=30,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=26096)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmhgfs-fuse on /mnt/hgfs type fuse.vmhgfs-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201800k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# umount /mnt
```

# mount -t ext2 -o loop ./fs.new /mnt

mount: wrong fs type, bad option, bad superblock on /dev/loop0, ...

# rmmod myext2



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t ext2 -o loop ./fs.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.
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# rmmod myext2
```

3.　修改文件系统操作

对于 mknod 函数，我们在 myext2 中作如下修改：

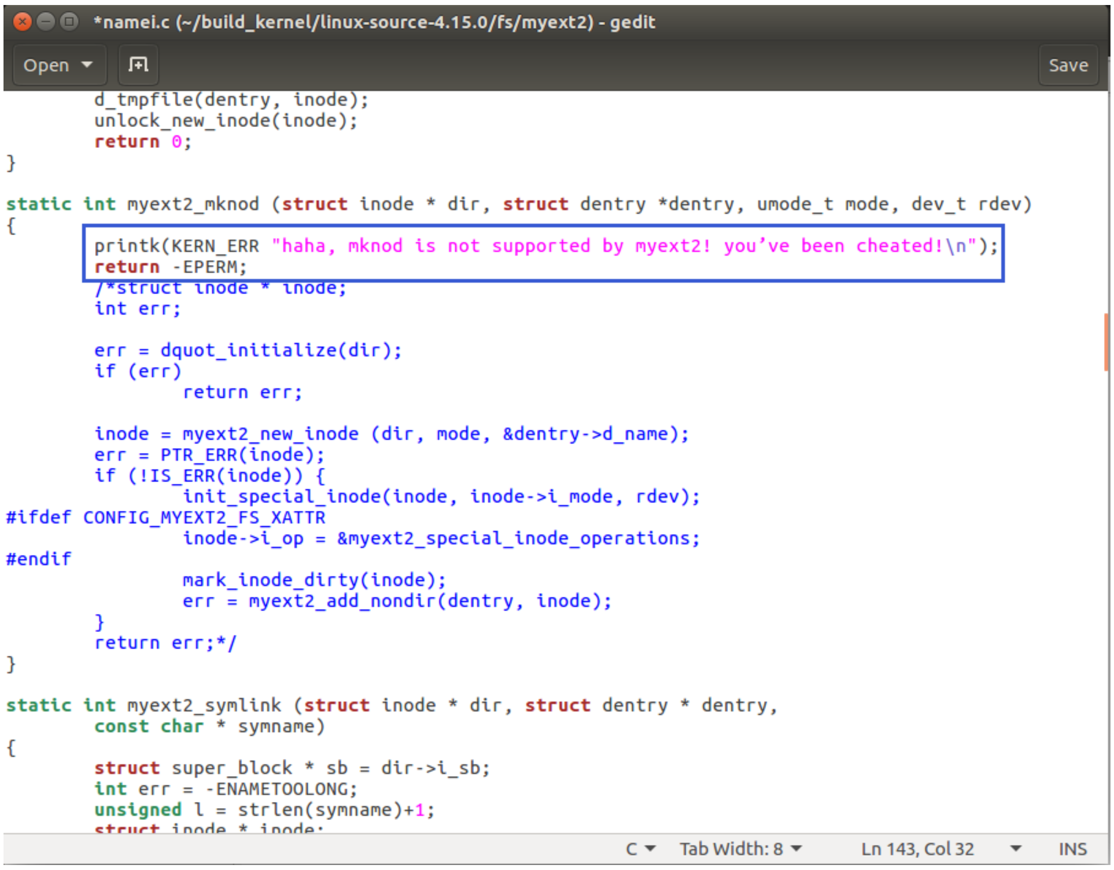
fs/myext2/namei.c

```
static int myext2_mknod (struct inode * dir, struct dentry *dentry, int mode, int rdev)
{
      printk(KERN_ERR “haha, mknod is not supported by myext2! you’ve been cheated!\n”);
      return -EPERM;
   /*
   …..
    把其它代码注释
   */
}
```

添加的程序中：

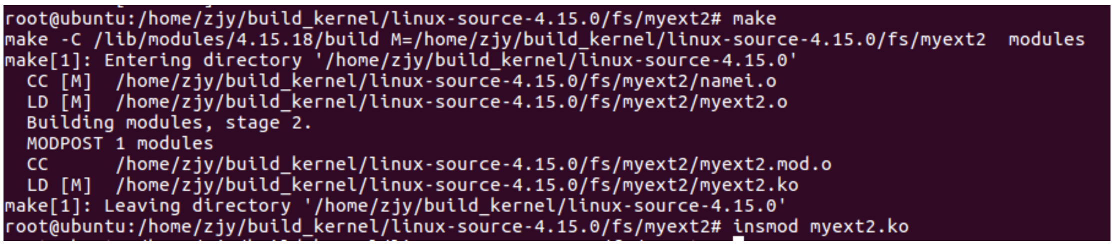
第一行 打印信息，说明 mknod 操作不被支持。

第二行 将错误号为 EPERM 的结果返回给 shell，即告诉 shell，在 myext2 文件系统中，maknod 不被支持。


*namei.c (~/build_kernel/linux-source-4.15.0/fs/myext2) - gedit
Open
Save
C
Tab Width: 8
Ln 143, Col 32
INS

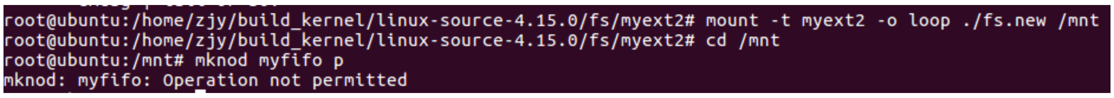
```
        d_tmpfile(dentry, inode);
        unlock_new_inode(inode);
        return 0;
}

static int myext2_mknod (struct inode * dir, struct dentry *dentry, umode_t mode, dev_t rdev)
{
        printk(KERN_ERR "haha, mknod is not supported by myext2! you've been cheated!\n");
        return -EPERM;
        /*struct inode * inode;
        int err;

        err = dquot_initialize(dir);
        if (err)
                return err;

        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;*/
}

static int myext2_symlink (struct inode * dir, struct dentry * dentry,
        const char * symname)
{
        struct super_block * sb = dir->i_sb;
        int err = -ENAMETOOLONG;
        unsigned l = strlen(symname)+1;
        struct inode * inode;
```

修改完毕，再用 make 重新编译内核模块，使用命令 insmod 安装编译好的 myext2.ko 内核模块。我们在 shell 下执行如下测试程序：



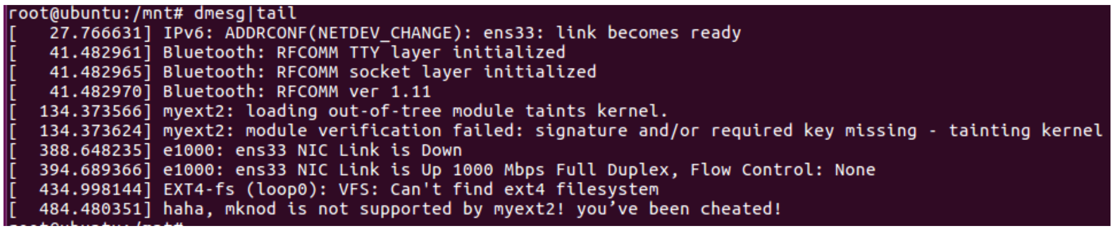
```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# make
make -C /lib/modules/4.15.18/build M=/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2  modules
make[1]: Entering directory '/home/zjy/build_kernel/linux-source-4.15.0'
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.mod.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/zjy/build_kernel/linux-source-4.15.0'
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# insmod myext2.ko
```

```
#mount –t myext2 –o loop ./fs.new /mnt
#cd /mnt
#mknod myfifo p
mknod: `myfifo': Operation not permitted
#
```



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# cd /mnt
root@ubuntu:/mnt# mknod myfifo p
mknod: myfifo: Operation not permitted
```

第一行命令：将 fs.new mount 到/mnt 目录下。
第二行命令：进入/mnt 目录，也就是进入 fs.new 这个 myext2 文件系统。
第三行命令：执行创建一个名为 myfifo 的命名管道的命令。
第四、五行是执行结果：第四行是我们添加的 myext2_mknod 函数的 printk 的结果；第五行是返回错误号 EPERM 结果给 shell，shell 捕捉到这个错误后打出的出错信息。需要注意的是，如果你是在图形界面下使用虚拟控制台，printk 打印出来的信息不一定能在你的终端上显示出来，但是可以通过命令 dmesg|tail 来观察。

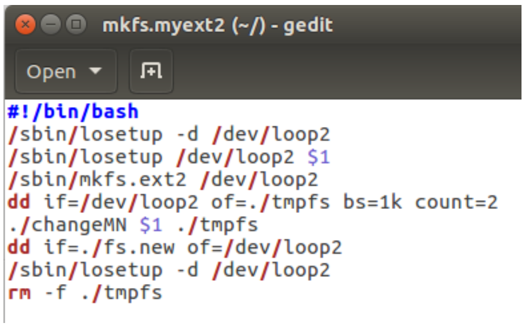


```
root@ubuntu:/mnt# dmesg|tail
[   27.766631] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[   41.482961] Bluetooth: RFCOMM TTY layer initialized
[   41.482965] Bluetooth: RFCOMM socket layer initialized
[   41.482970] Bluetooth: RFCOMM ver 1.11
[  134.373566] myext2: loading out-of-tree module taints kernel.
[  134.373624] myext2: module verification failed: signature and/or required key missing - tainting kernel
[  388.648235] e1000: ens33 NIC Link is Down
[  394.689366] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[  434.998144] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[  484.480351] haha, mknod is not supported by myext2! you've been cheated!
```

4. 添加文件系统创建工具

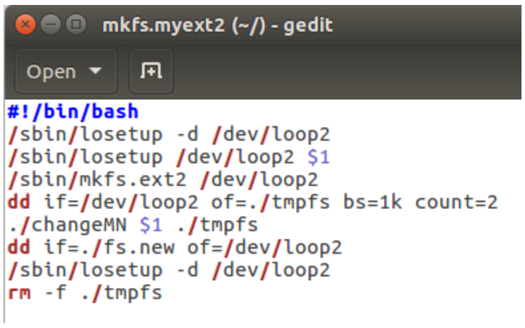
我们在主目录下编辑如下的程序: ~/mkfs.myext2

```
#!/bin/bash
/sbin/losetup -d /dev/loop2
/sbin/losetup /dev/loop2 $1
/sbin/mkfs.ext2 /dev/loop2
dd if=/dev/loop2 of=./tmpfs bs=1k count=2
./changeMN $1 ./tmpfs
dd if=./fs.new of=/dev/loop2
/sbin/losetup -d /dev/loop2
rm -f ./tmpfs
```

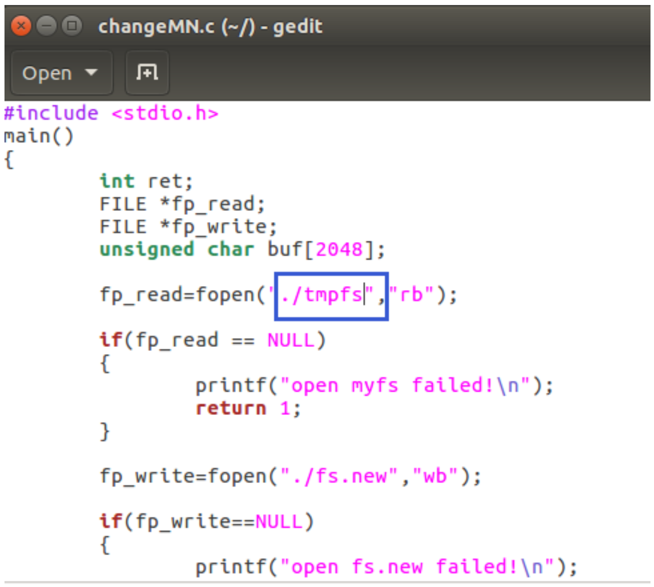


```
#!/bin/bash
/sbin/losetup -d /dev/loop2
/sbin/losetup /dev/loop2 $1
/sbin/mkfs.ext2 /dev/loop2
dd if=/dev/loop2 of=./tmpfs bs=1k count=2
./changeMN $1 ./tmpfs
dd if=./fs.new of=/dev/loop2
/sbin/losetup -d /dev/loop2
rm -f ./tmpfs
```

我们发现 mkfs.myext2 脚本中的 changeMN 程序功能，与 4.2 节的 changeMN 功能不一样，下面修改 changeMN.c 程序，以适合本节 mkfs.myext2 和下面测试的需要。



```
#!/bin/bash
/sbin/losetup -d /dev/loop2
/sbin/losetup /dev/loop2 $1
/sbin/mkfs.ext2 /dev/loop2
dd if=/dev/loop2 of=./tmpfs bs=1k count=2
./changeMN $1 ./tmpfs
dd if=./fs.new of=/dev/loop2
/sbin/losetup -d /dev/loop2
rm -f ./tmpfs
```



```
#include <stdio.h>
main()
{
        int ret;
        FILE *fp_read;
        FILE *fp_write;
        unsigned char buf[2048];

        fp_read=fopen("./tmpfs","rb");

        if(fp_read == NULL)
        {
                printf("open myfs failed!\n");
                return 1;
        }

        fp_write=fopen("./fs.new","wb");

        if(fp_write==NULL)
        {
                printf("open fs.new failed!\n");
```

编辑完了之后，做如下测试。

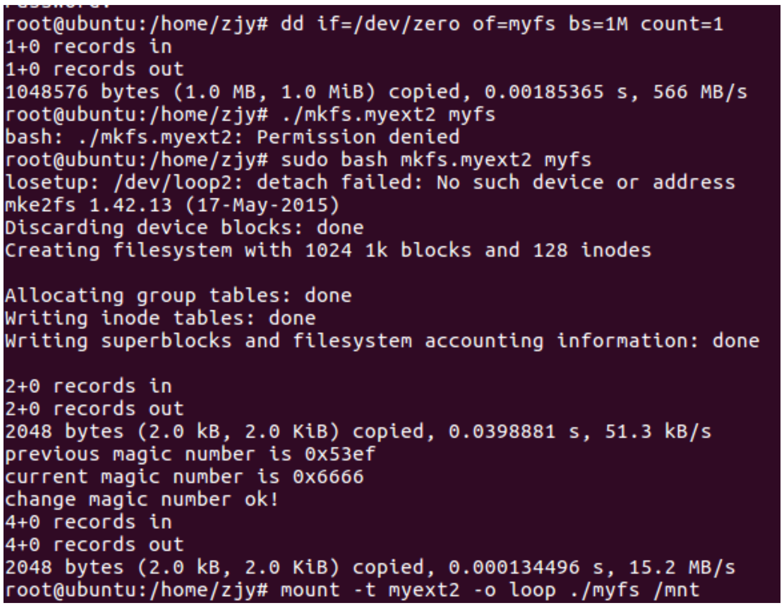
# dd if=/dev/zero of=myfs bs=1M count=1

# ./mkfs.myext2 myfs (或 sudo bash mkfs.myext2 myfs)

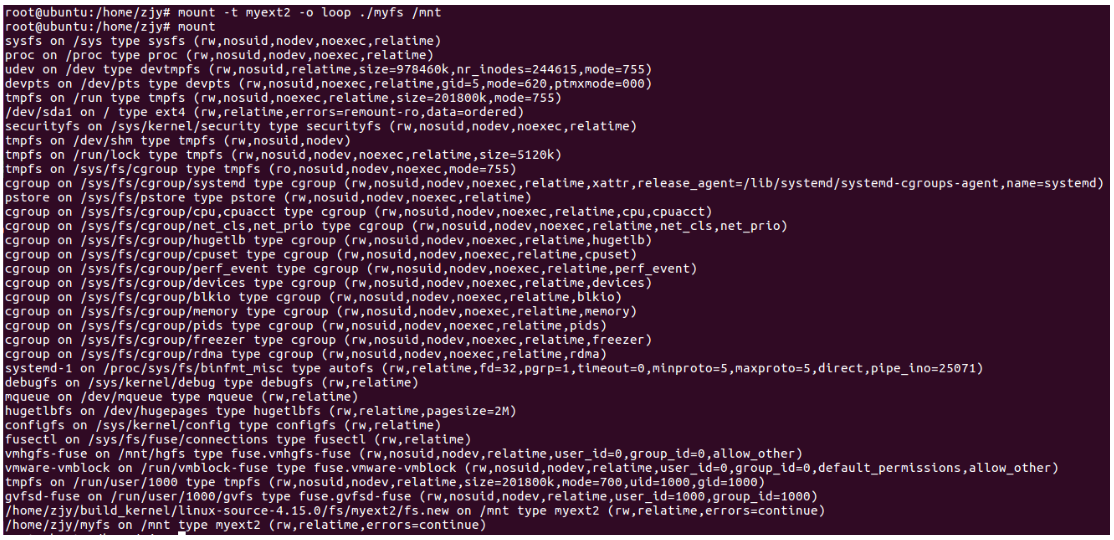
#sudo mount -t myext2 -o loop ./myfs /mnt

# mount

/dev/loop on /mnt myext2 (rw)



```
root@ubuntu:/home/zjy# dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00185365 s, 566 MB/s
root@ubuntu:/home/zjy# ./mkfs.myext2 myfs
bash: ./mkfs.myext2: Permission denied
root@ubuntu:/home/zjy# sudo bash mkfs.myext2 myfs
losetup: /dev/loop2: detach failed: No such device or address
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

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

2+0 records in
2+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.0398881 s, 51.3 kB/s
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
4+0 records in
4+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.000134496 s, 15.2 MB/s
root@ubuntu:/home/zjy# mount -t myext2 -o loop ./myfs /mnt
```

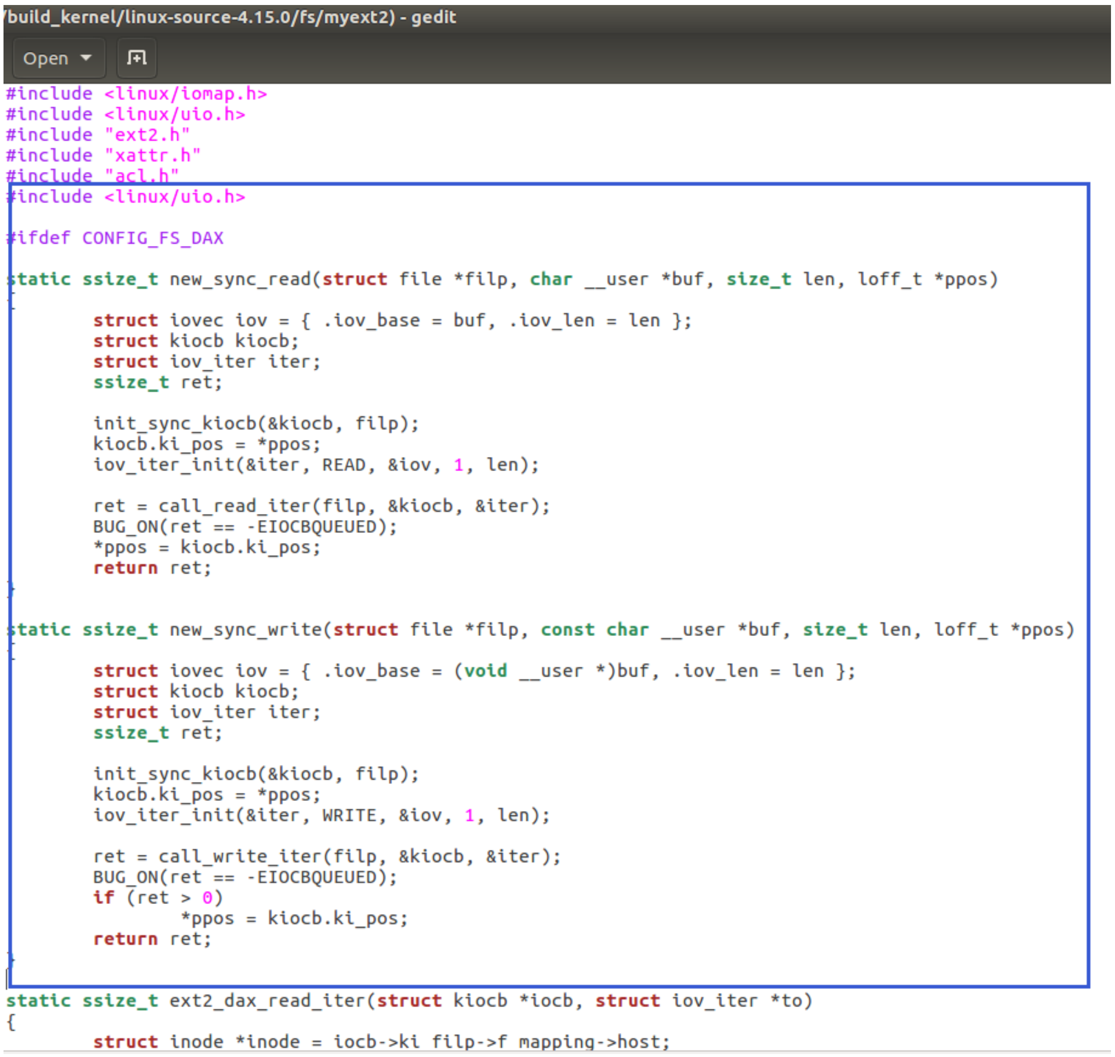


```
root@ubuntu:/home/zjy# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:/home/zjy# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=978460k,nr_inodes=244615,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=201800k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=32,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=25071)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
configfs on /sys/kernel/config type configfs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
vmhgfs-fuse on /mnt/hgfs type fuse.vmhgfs-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201800k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
/home/zjy/myfs on /mnt type myext2 (rw,relatime,errors=continue)
```

5.　修改加密文件系统的 read 和 write 操作

在内核模块 myext2.ko 中修改 file.c 的代码，添加两个函数 new_sync_read_crypt 和 new_sync_read_crypt，将这两个函数指针赋给 myext2_file_operations 结构中的 read 和 write 操作。在 new_sync_write_crypt 中增加对用户传入数据 buf 的加密，在 new_sync_read_crypt 中增加解密。可以使用 DES 等加密和解密算法。

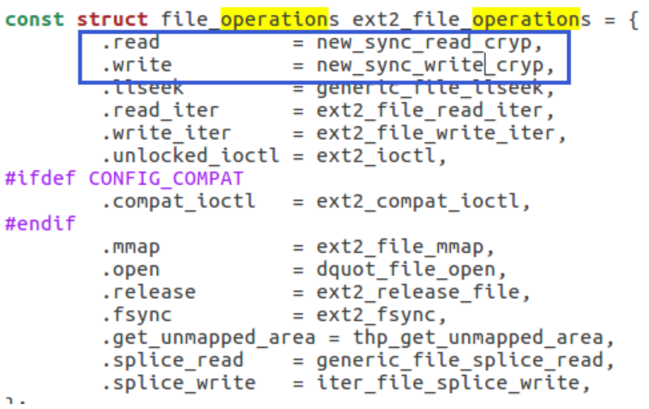
首先把 fs/read_write.c 中的 new_sync_read 和 new_sync_write 两个函数复制到 file.c 中，再添加两个新函数。



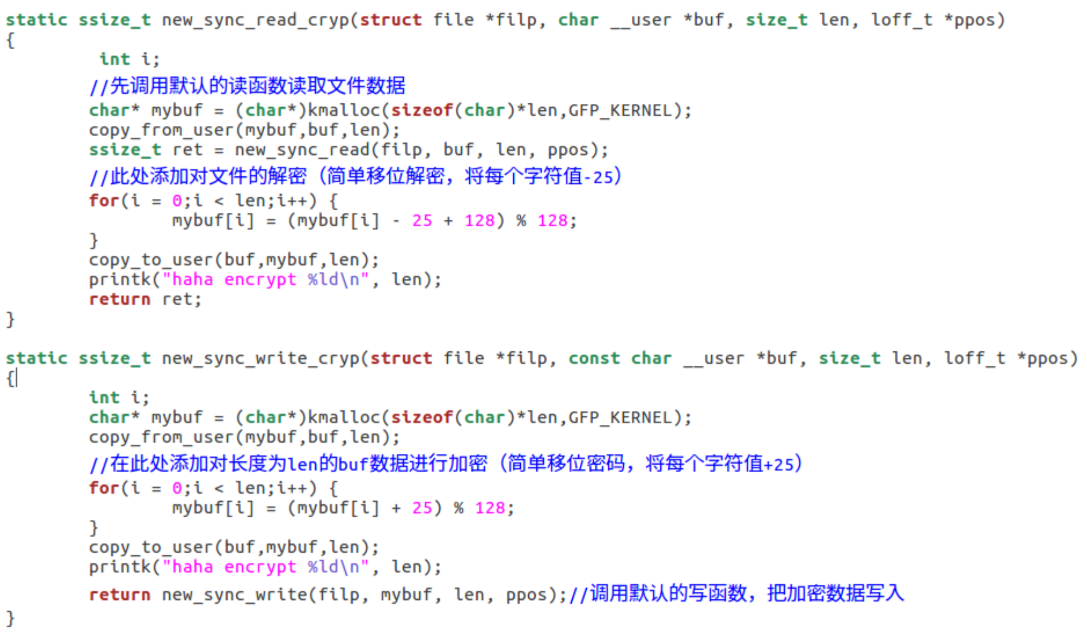
/build_kernel/linux-source-4.15.0/fs/myext2) - gedit

Open

```
#include <linux/iomap.h>
#include <linux/uio.h>
#include "ext2.h"
#include "xattr.h"
#include "acl.h"
#include <linux/uio.h>

#ifdef CONFIG_FS_DAX

static ssize_t new_sync_read(struct file *filp, char __user *buf, size_t len, loff_t *ppos)
{
	struct iovec iov = { .iov_base = buf, .iov_len = len };
	struct kiocb kiocb;
	struct iov_iter iter;
	ssize_t ret;

	init_sync_kiocb(&kiocb, filp);
	kiocb.ki_pos = *ppos;
	iov_iter_init(&iter, READ, &iov, 1, len);

	ret = call_read_iter(filp, &kiocb, &iter);
	BUG_ON(ret == -EIOCBQUEUED);
	*ppos = kiocb.ki_pos;
	return ret;
}


static ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t len, loff_t *ppos)
{
	struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len };
	struct kiocb kiocb;
	struct iov_iter iter;
	ssize_t ret;

	init_sync_kiocb(&kiocb, filp);
	kiocb.ki_pos = *ppos;
	iov_iter_init(&iter, WRITE, &iov, 1, len);

	ret = call_write_iter(filp, &kiocb, &iter);
	BUG_ON(ret == -EIOCBQUEUED);
	if (ret > 0)
		*ppos = kiocb.ki_pos;
	return ret;
}

static ssize_t ext2_dax_read_iter(struct kiocb *iocb, struct iov_iter *to)
{
	struct inode *inode = iocb->ki_filp->f_mapping->host;
```

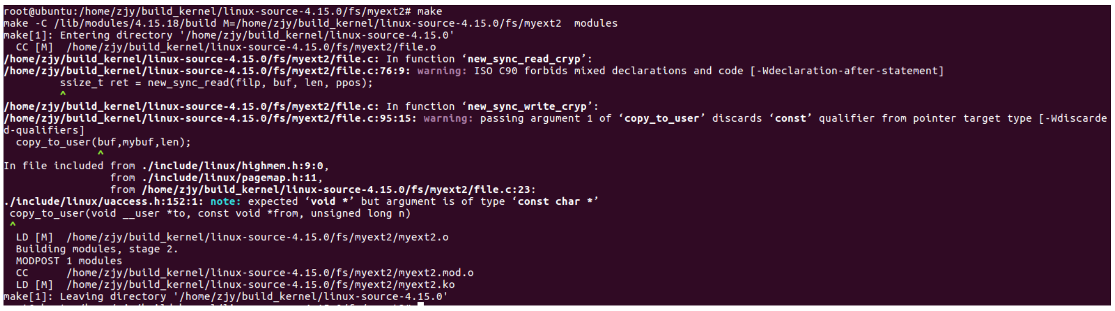


```
const struct file_operations ext2_file_operations = {
	.read		= new_sync_read_cryp,
	.write		= new_sync_write_cryp,
	.llseek		= generic_file_llseek,
	.read_iter	= ext2_file_read_iter,
	.write_iter	= ext2_file_write_iter,
	.unlocked_ioctl = ext2_ioctl,
#ifdef CONFIG_COMPAT
	.compat_ioctl	= ext2_compat_ioctl,
#endif
	.mmap		= ext2_file_mmap,
	.open		= dquot_file_open,
	.release	= ext2_release_file,
	.fsync		= ext2_fsync,
	.get_unmapped_area = thp_get_unmapped_area,
	.splice_read	= generic_file_splice_read,
	.splice_write	= iter_file_splice_write,
};
```

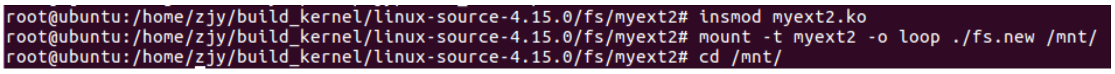


```
static ssize_t new_sync_read_cryp(struct file *filp, char __user *buf, size_t len, loff_t *ppos)
{
         int i;
        //先调用默认的读函数读取文件数据
        char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
        copy_from_user(mybuf,buf,len);
        ssize_t ret = new_sync_read(filp, buf, len, ppos);
        //此处添加对文件的解密（简单移位解密，将每个字符值-25）
        for(i = 0;i < len;i++) {
                mybuf[i] = (mybuf[i] - 25 + 128) % 128;
        }
        copy_to_user(buf,mybuf,len);
        printk("haha encrypt %ld\n", len);
        return ret;
}

static ssize_t new_sync_write_cryp(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);
        //在此处添加对长度为len的buf数据进行加密（简单移位密码，将每个字符值+25）
        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 new_sync_write(filp, mybuf, len, ppos);//调用默认的写函数，把加密数据写入
}
```

上述修改完成后，再用 make 重新编译 myext2 模块，使用命令 insmod 安装编译好的 myext2.ko 内核模块。重新加载 myext2 内核模块，创建一个 myext2 文件系统，并尝试往文件系统中写入一个字符串文件。

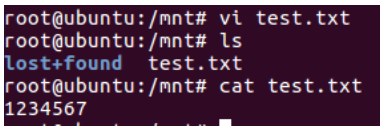


```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# make
make -C /lib/modules/4.15.18/build M=/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2  modules
make[1]: Entering directory '/home/zjy/build_kernel/linux-source-4.15.0'
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.o
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c: In function ‘new_sync_read_cryp’:
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c:76:9: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
         ssize_t ret = new_sync_read(filp, buf, len, ppos);
         ^
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c: In function ‘new_sync_write_cryp’:
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c:95:15: warning: passing argument 1 of ‘copy_to_user’ discards ‘const’ qualifier from pointer target type [-Wdiscarde
d-qualifiers]
  copy_to_user(buf,mybuf,len);
               ^
In file included from ./include/linux/highmem.h:9:0,
                 from ./include/linux/pagemap.h:11,
                 from /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c:23:
./include/linux/uaccess.h:152:1: note: expected ‘void *’ but argument is of type ‘const char *’
 copy_to_user(void __user *to, const void *from, unsigned long n)
 ^
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.mod.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/zjy/build_kernel/linux-source-4.15.0'
```



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# insmod myext2.ko
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# mount -t myext2 -o loop ./fs.new /mnt/
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# cd /mnt/
```

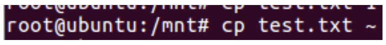
新建文件 test.txt 并写入字符串“1234567”，再查看 test.txt 文件内容：cat test.txt 。



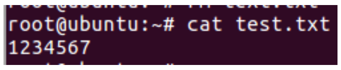
```
root@ubuntu:/mnt# vi test.txt
root@ubuntu:/mnt# ls
lost+found  test.txt
root@ubuntu:/mnt# cat test.txt
1234567
```

把 test.txt 文件复制到主目录下：cp test.txt ~ 。

在主目录下打开 test.txt 文件，查看 test.txt 文件内容的结果：

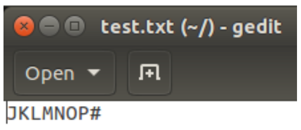


```
root@ubuntu:/mnt# cp test.txt ~
```



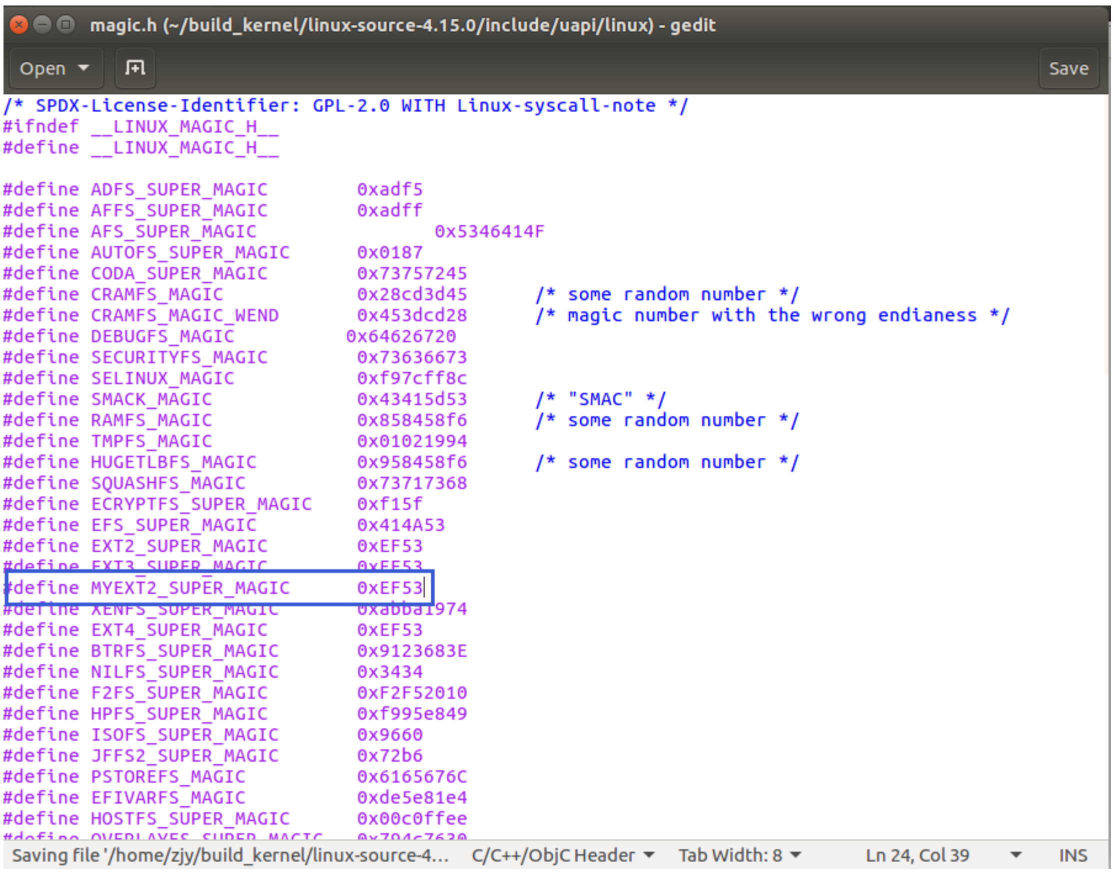
```
root@ubuntu:~# cat test.txt
1234567
```

使用文件管理器的复制，再查看结果：



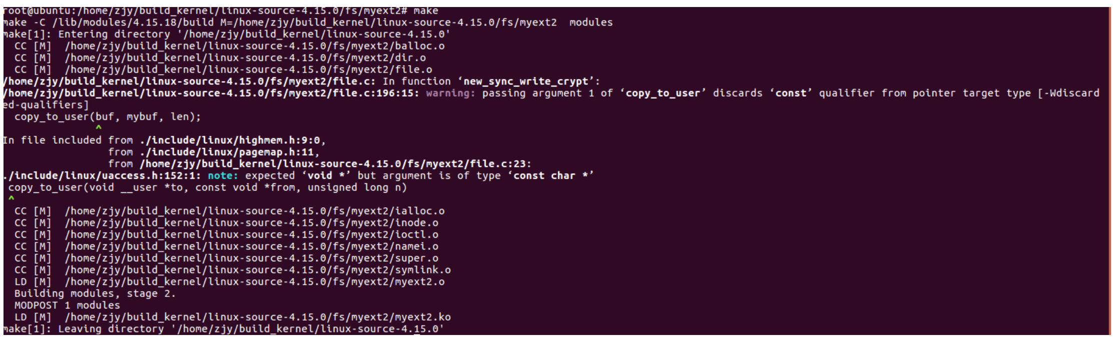
```
JKLMNOP#
```

我们把之前的 magic number 改回 0xEF53。


magic.h (~/build_kernel/linux-source-4.15.0/include/uapi/linux) - gedit
Open
Save

```
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

#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
#define MYEXT2_SUPER_MAGIC      0xEF53
#define XENFS_SUPER_MAGIC       0xabba1974
#define EXT4_SUPER_MAGIC        0xEF53
#define BTRFS_SUPER_MAGIC       0x9123683E
#define NILFS_SUPER_MAGIC       0x3434
#define F2FS_SUPER_MAGIC        0xF2F52010
#define HPFS_SUPER_MAGIC        0xf995e849
#define ISOFS_SUPER_MAGIC       0x9660
#define JFFS2_SUPER_MAGIC       0x72b6
#define PSTOREFS_MAGIC          0x6165676C
#define EFIVARFS_MAGIC          0xde5e81e4
#define HOSTFS_SUPER_MAGIC      0x00c0ffee
#define OVERLAYFS_SUPER_MAGIC   0x794c7630
```

重新编译 myext2 模块，安装 myext2.ko 后，



```
root@ubuntu:/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2# make
make -C /lib/modules/4.15.18/build M=/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2  modules
make[1]: Entering directory '/home/zjy/build_kernel/linux-source-4.15.0'
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/balloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/dir.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.o
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c: In function ‘new_sync_write_crypt’:
/home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c:196:15: warning: passing argument 1 of ‘copy_to_user’ discards ‘const’ qualifier from pointer target type [-Wdiscard
ed-qualifiers]
  copy_to_user(buf, mybuf, len);
               ^
In file included from ./include/linux/highmem.h:9:0,
                 from ./include/linux/pagemap.h:11,
                 from /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/file.c:23:
./include/linux/uaccess.h:152:1: note: expected ‘void *’ but argument is of type ‘const char *’
 copy_to_user(void __user *to, const void *from, unsigned long n)
 ^
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ialloc.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/inode.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/ioctl.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/super.o
  CC [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/symlink.o
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/zjy/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/zjy/build_kernel/linux-source-4.15.0'
```

执行下面命令：

dd if=/dev/zero of=myfs bs=1M count=1

/sbin/mkfs.ext2 myfs

mount -t myext2 -o loop ./myfs /mnt

cd /mnt

echo “1234567”   >   test.txt

cat test.txt

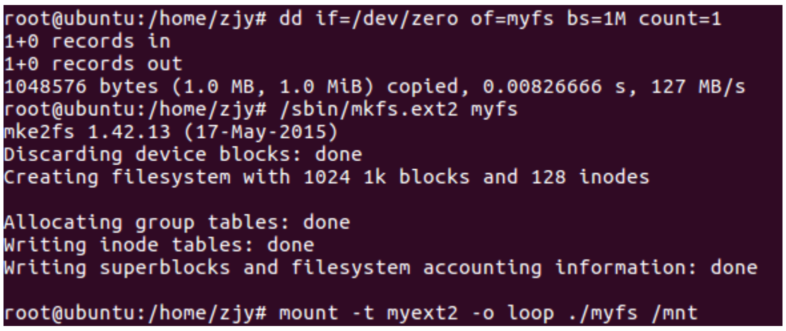
cd

umount /mnt

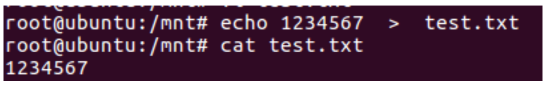
mount -t ext2 -o loop ./myfs /mnt

cd /mnt

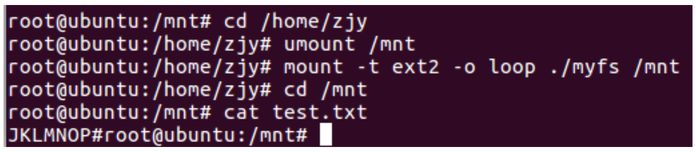
cat test.txt



```
root@ubuntu:/home/zjy# dd if=/dev/zero of=myfs bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00826666 s, 127 MB/s
root@ubuntu:/home/zjy# /sbin/mkfs.ext2 myfs
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

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

root@ubuntu:/home/zjy# mount -t myext2 -o loop ./myfs /mnt
```



```
root@ubuntu:/mnt# echo 1234567  >  test.txt
root@ubuntu:/mnt# cat test.txt
1234567
```



```
root@ubuntu:/mnt# cd /home/zjy
root@ubuntu:/home/zjy# umount /mnt
root@ubuntu:/home/zjy# mount -t ext2 -o loop ./myfs /mnt
root@ubuntu:/home/zjy# cd /mnt
root@ubuntu:/mnt# cat test.txt
JKLMNOP#root@ubuntu:/mnt#
```

结果分析：

Shell 的 cp 指令复制文件是不加密的。而利用文件系统管理器拷贝会将文件加密的。即使使用 ext2 文件系统的 magic number，在 myext2 文件系统中创建的文件都是加密文件

源程序：

changeMN.c

```
#include <stdio.h>
main()
{
	int ret;
	FILE *fp_read;
	FILE *fp_write;
	unsigned char buf[2048];

	fp_read=fopen("./tmpfs","rb");

	if(fp_read == NULL)
	{
		printf("open myfs failed!\n");
		return 1;
	}

	fp_write=fopen("./fs.new","wb");

	if(fp_write==NULL)
```

```
    {
        printf("open fs.new failed!\n");
        return 2;
    }

    ret=fread(buf,sizeof(unsigned char),2048,fp_read);

    printf("previous magic number is 0x%x%x\n",buf[0x438],buf[0x439]);

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

    fwrite(buf,sizeof(unsigned char),2048,fp_write);

    printf("current magic number is 0x%x%x\n",buf[0x438],buf[0x439]);

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

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

    fclose(fp_read);
    fclose(fp_write);

    return 0;
}
```

new_sync_read_crypt

```
static 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 = new_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 encrypt %ld\n", len);
return ret;
}
```

new_sync_write_crypt

```
static ssize_t new_sync_write_crypt(struct file *filp, const char __user *bu f, 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\n", len);
        return new_sync_write(filp, buf, len, ppos);
}
```

三、讨论、心得（20 分）

1. 这个实验做了三遍，结果发现第一遍做的就是正确的，但是后面几遍实验让我加深了对实验的理解。之前没有弄明白每个指令应该在哪一个父目录下运行，就有些步骤做错了。
2. 我感觉直接从外面复制到命令行的指令会有字符的问题。一模一样的指令，有的可以运行，有的就会报格式错误。
3. 我在 myext2 执行脚本之后，文件夹内的文件就全变成只读文件了，又因为后面的实验要进行修改操作，在命令行中用 vi 编辑比较麻烦，因此 chmod 666 改了一下权限。
4. Shell 中的 cp 将首先读取文件中的数据，然后写到新位置的文件中去，因此当我们从文件系统复制文件时，myext2 将被挂载在/mnt，而其他的文件系统在 ubuntu 上正常运行。我们先解密数据，然后写回，磁盘中的数据存储就不会加密。而利用文

件系统管理器拷贝的副本使用 mmap 而不是读取，直接映射数据，因此是加密的。