

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： 沈子衿 学号： 3160104734
电子邮件地址： zijinshen@zju.edu.cn
实验日期： 2018 年 12 月 27 日

一、实验环境

主机配置：

- 主机型号：Lenovo ThinkPad T480
- 内存：16GB
- 处理器：i7-8550U
- 操作系统：Windows 10 家庭中文版

虚拟机配置：

- 虚拟机环境：Vmware Workstation Pro 14
- Ubuntu 版本：16.04
- Linux 内核版本：3.18.24
- 编译文件模块使用的素材内核版本：3.18.24

二、实验内容和结果及分析

1. 实验设计思路

本实验的内容是要添加一个类似于 ext2 的自定义文件系统 myext2。myext2 文件系统的描述如下：

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

我们的工作，就是要通过修改内核代码、编译并插入对应的内核模块在 Linux 上实现这个自定义的 myext2 文件系统，并对 myext2 的稳定性和一些基本操作进行严格的测试。

2. 实验步骤及截图

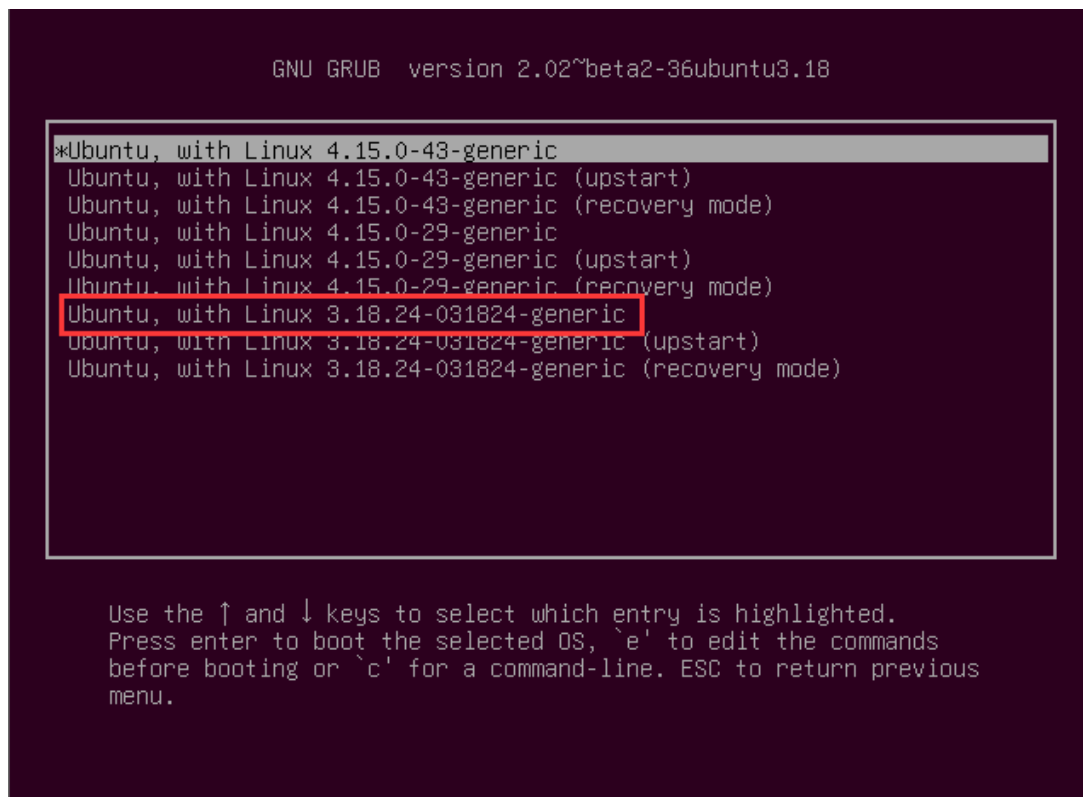
本次实验，我们使用 3.18.24 的内核和配置了 3.18.24 内核的 Ubuntu16.04 发行版进行实验。在实验过程中，由于我曾试图使用 4.15 的内核（最终失败了），且部分 3.18.24 下的操作尝试了多次才成功，也没有来得及截图，因此接下来的部分截图可能截取自对 4.15 的操作。

作。不过，这不会影响我对实验过程的叙述。

- 环境配置

首先，从 <https://kernel.ubuntu.com/~kernel-ppa/mainline/v3.18.24-vivid/> 网站下载 Ubuntu 发行版内核（3.18.24 版本）对应文件（包括 image 和 header）并在 Ubuntu16.04 环境下双击安装，安装完毕后重启切换内核：

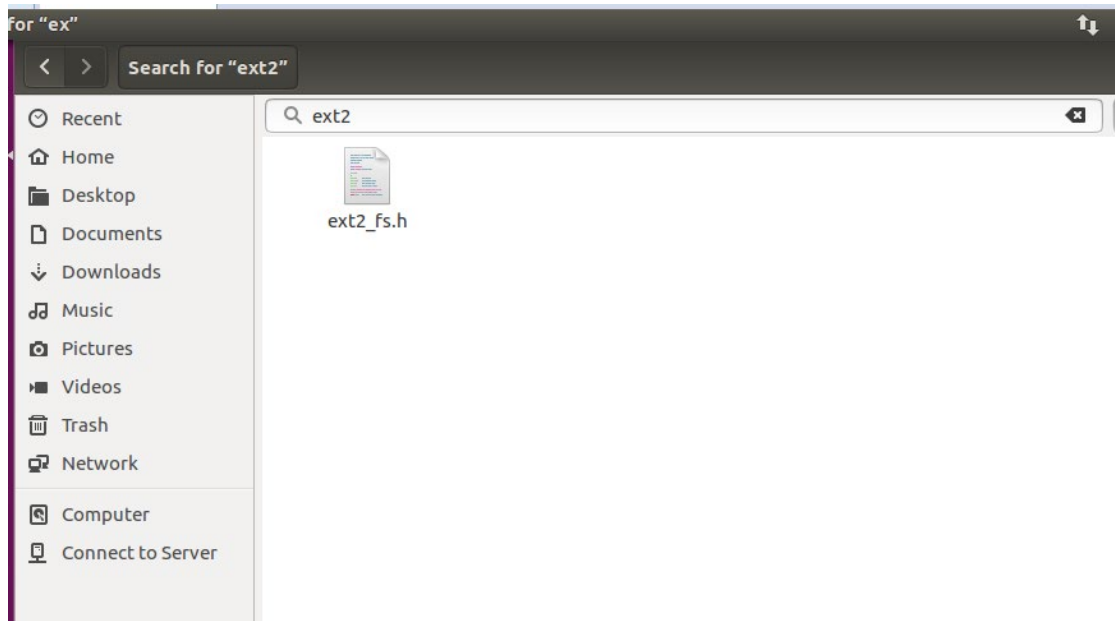
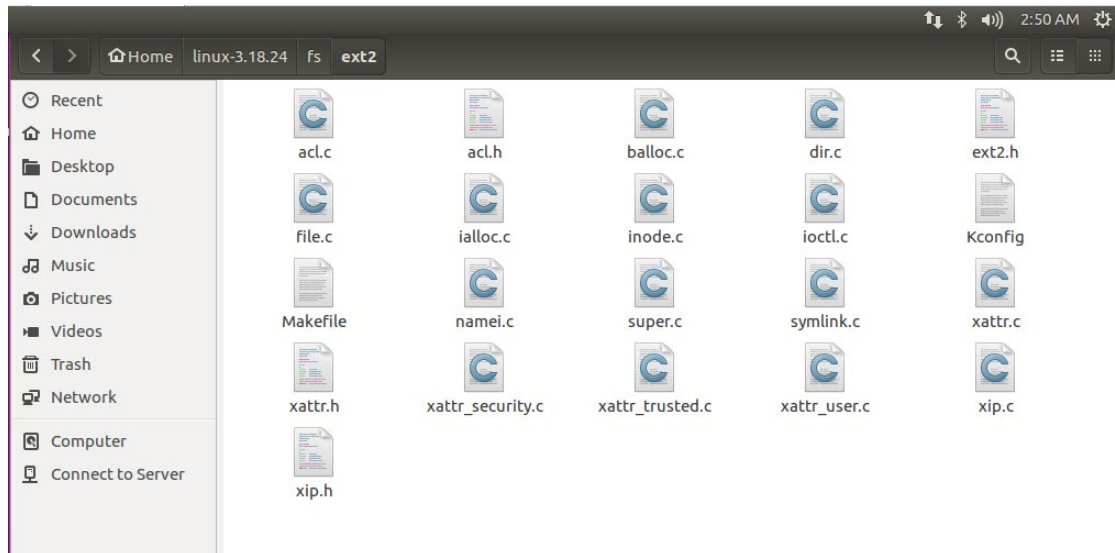
	linux-headers-3.18.24-031824-generic-lpae 3.18.24-031824.201511031331 armhf.deb	2015-11-03 19:11 733K
	linux-headers-3.18.24-031824-generic 3.18.24-031824.201511031331 amd64.deb	2015-11-03 18:44 703K
	linux-headers-3.18.24-031824-generic 3.18.24-031824.201511031331 armhf.deb	2015-11-03 19:09 737K
	linux-headers-3.18.24-031824-generic 3.18.24-031824.201511031331 i386.deb	2015-11-03 18:57 689K
	linux-headers-3.18.24-031824-generic 3.18.24-031824.201511031331 ppc64el.deb	2015-11-03 19:17 712K
	linux-headers-3.18.24-031824-lowlatency 3.18.24-031824.201511031331 amd64.deb	2015-11-03 18:45 703K
	linux-headers-3.18.24-031824-lowlatency 3.18.24-031824.201511031331 i386.deb	2015-11-03 18:58 689K
	linux-headers-3.18.24-031824 3.18.24-031824.201511031331 all.deb	2015-11-03 18:33 8.8M
	linux-image-3.18.24-031824-generic-lpae 3.18.24-031824.201511031331 armhf.deb	2015-11-03 19:11 51M
	linux-image-3.18.24-031824-generic 3.18.24-031824.201511031331 amd64.deb	2015-11-03 18:44 52M
	linux-image-3.18.24-031824-generic 3.18.24-031824.201511031331 armhf.deb	2015-11-03 19:09 52M
	linux-image-3.18.24-031824-generic 3.18.24-031824.201511031331 i386.deb	2015-11-03 18:56 51M
	linux-image-3.18.24-031824-generic 3.18.24-031824.201511031331 ppc64el.deb	2015-11-03 19:17 50M
	linux-image-3.18.24-031824-lowlatency 3.18.24-031824.201511031331 amd64.deb	2015-11-03 18:45 52M
	linux-image-3.18.24-031824-lowlatency 3.18.24-031824.201511031331 i386.deb	2015-11-03 18:58 51M



成功切换内核，可以正式开始实验了。

- 添加一个类似 ext2 的文件系统 myext2

首先，按照实验要求对素材内核~linux-3.18.24/目录下一 ext2 文件系统源代码新建“myext2”文件系统源代码，并将其中的 ext2.h 修改为 myext2.h；然后，对本机内核/lib/modules/\$(uname -r)/build /include/路径下的代码进行修改，以 ext2 相关源文件为模板建立 myext2 相关源文件：



实现上述操作的命令如下：

```
rikka@ubuntu: /lib/modules/4.15.0-29-generic/build/include/asm-generic/bitops
rikka@ubuntu:~$ sudo cd ~/linux-3.18.24/
[sudo] password for rikka:
sudo: cd: command not found
rikka@ubuntu:~$ cd ~/linux-3.18.24/
rikka@ubuntu:~/linux-3.18.24$ cd fs
rikka@ubuntu:~/linux-3.18.24/fs$ sudo cp -R ext2 myext2
rikka@ubuntu:~/linux-3.18.24/fs$ cd ~/linux-3.18.24/fs/myext2/
rikka@ubuntu:~/linux-3.18.24/fs/myext2$ mv ext2.h myext2.h
mv: cannot move 'ext2.h' to 'myext2.h': Permission denied
rikka@ubuntu:~/linux-3.18.24/fs/myext2$ sudo mv ext2.h myext2.h
rikka@ubuntu:~/linux-3.18.24/fs/myext2$ cd /lib/modules/$(uname -r)/build/include/linux
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/linux$ sudo cp ext2_fs.h myext2_fs.h
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/linux$ cd ..
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include$ cd asm-generic
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/asm-generic$ cd bitops/
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/asm-generic/bitops$ cp ext2-atomic.h
myext2-atomic.h
cp: cannot create regular file 'myext2-atomic.h': Permission denied
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/asm-generic/bitops$ sudo cp ext2-ato
mic.h myext2-atomic.h
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/asm-generic/bitops$ sudo cp ext2-ato
mic-setbit.h myext2-atomic-setbit.h
rikka@ubuntu:/lib/modules/4.15.0-29-generic/build/include/asm-generic/bitops$
```

这之后，对复制过来的 myext2 源代码中的字符进行替换，即将所有 ext2 替换为 myext2（包括大、小写），脚本编辑如下：

```
substitute.sh (~/.linux-3.18.24/fs/myext2) - gedit
#!/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
```

这里我没有按照实验要求，直接 copy 了代码，结果出现了 windows 和类 unix 系统换行符不兼容的问题：

```

rikka@ubuntu: ~/linux-3.18.24/fs/myext2
#!/bin/bash^M
^M
SCRIPT=substitute.sh^M
^M
for f in *^M
do ^M
    if [ $f = $SCRIPT ]^M
    then^M
        echo "skip $f"^M
        continue^M
    fi^M
^M
    echo -n "substitute ext2 to myext2 in $f..."^M
    cat $f | sed 's/ext2/myext2/g' > ${f}_tmp^M
    mv ${f}_tmp $f^M
    echo "done"^M
^M
    echo -n "substitute EXT2 to MYEXT2 in $f..."^M
    cat $f | sed 's/EXT2/MYEXT2/g' > ${f}_tmp^M
    mv ${f}_tmp $f^M
    echo "done"^M
^M
done^M
"substitute.sh" 23 lines, 422 characters

```

这一问题通过 dos2unix 得到了解决：

```

rikka@ubuntu:~/linux-3.18.24/fs/myext2$ sudo dos2unix substitute.sh
dos2unix: converting file substitute.sh to Unix format ...
rikka@ubuntu:~/linux-3.18.24/fs/myext2$

```

最终正确的脚本执行结果如下：

```

rikka@ubuntu:~/linux-3.18.24/fs/myext2$ sudo 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

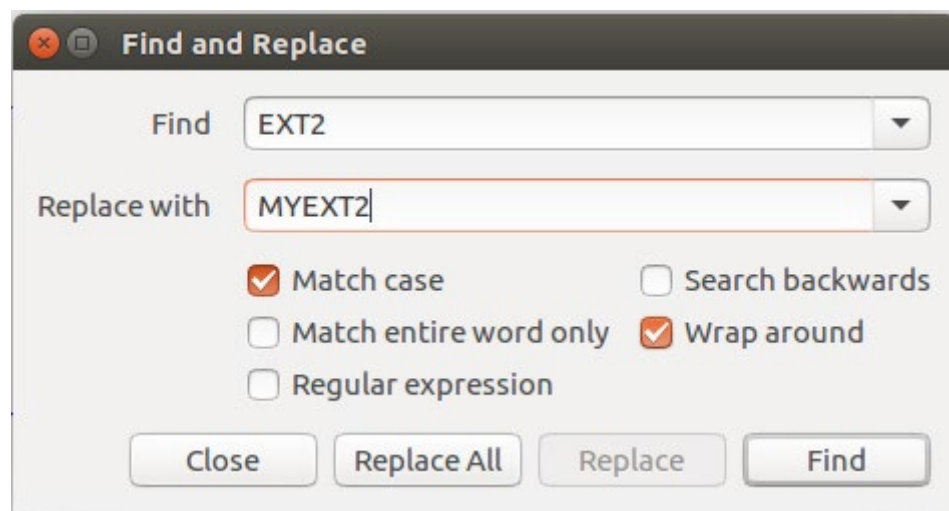
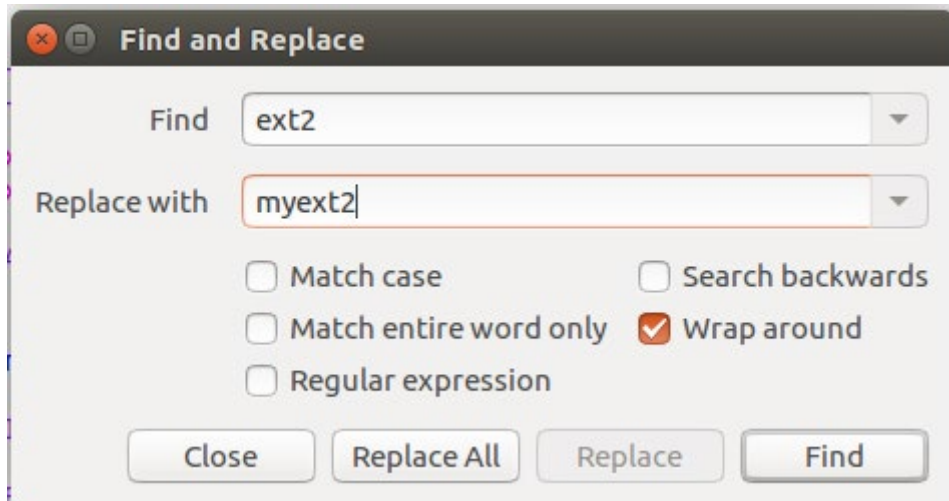
```

然后，使用管理员模式运行文件管理器,进入系统内核根目录将有关文件中的 ext2 进行替换，并添加包含：

```
rikka@ubuntu: ~  
rikka@ubuntu:~$ sudo nautilus  
[sudo] password for rikka:  
  
(nautilus:3844): Gtk-WARNING **: Failed to register client: GDBus.Error:org.free  
desktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not pro  
vided by any .service files  
  
** (nautilus:3844): CRITICAL **: Another desktop manager in use; desktop window  
won't be created  
  
(nautilus:3844): IBUS-WARNING **: The owner of /home/rikka/.config/ibus/bus is n  
ot root!  
Nautilus-Shares-Message: Called "net usershare info" but it failed: Failed to exe  
cute child process "net" (No such file or directory)  
Nautilus-Shares-Message: Called "net usershare info" but it failed: Failed to exe  
cute child process "net" (No such file or directory)  
  
(gedit:3888): IBUS-WARNING **: The owner of /home/rikka/.config/ibus/bus is not  
root!
```

此处以修改 myext2_fs.h 为例：

```
#ifndef _LINUX_EXT2_FS_H  
#define _LINUX_EXT2_FS_H  
  
#include <linux/types.h>  
#include <linux/magic.h>  
  
#define EXT2_NAME_LEN 255  
  
/*  
 * Maximal count of links to a file  
 */  
#define EXT2_LINK_MAX 32000  
  
#define EXT2_SB_MAGIC_OFFSET 0x38  
#define EXT2_SB_BLOCKS_OFFSET 0x04  
#define EXT2_SB_BSIZE_OFFSET 0x18  
  
static inline u64 ext2_image_size(void *ext2_sb)  
{  
    __u8 *p = ext2_sb;  
    if ((__le16 *) (p + EXT2_SB_MAGIC_OFFSET) != cpu_to_le16  
(EXT2_SUPER_MAGIC))  
        return 0;  
    return (u64) le32_to_cpup((__le32 *) (p + EXT2_SB_BLOCKS_OFFSET)) <<  
        le32_to_cpup((__le32 *) (p + EXT2_SB_BSIZE_OFFSET));  
}  
  
#endif /* _LINUX_EXT2_FS_H */
```

替换内容后的 my_ext2 如图所示：

```

/* 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

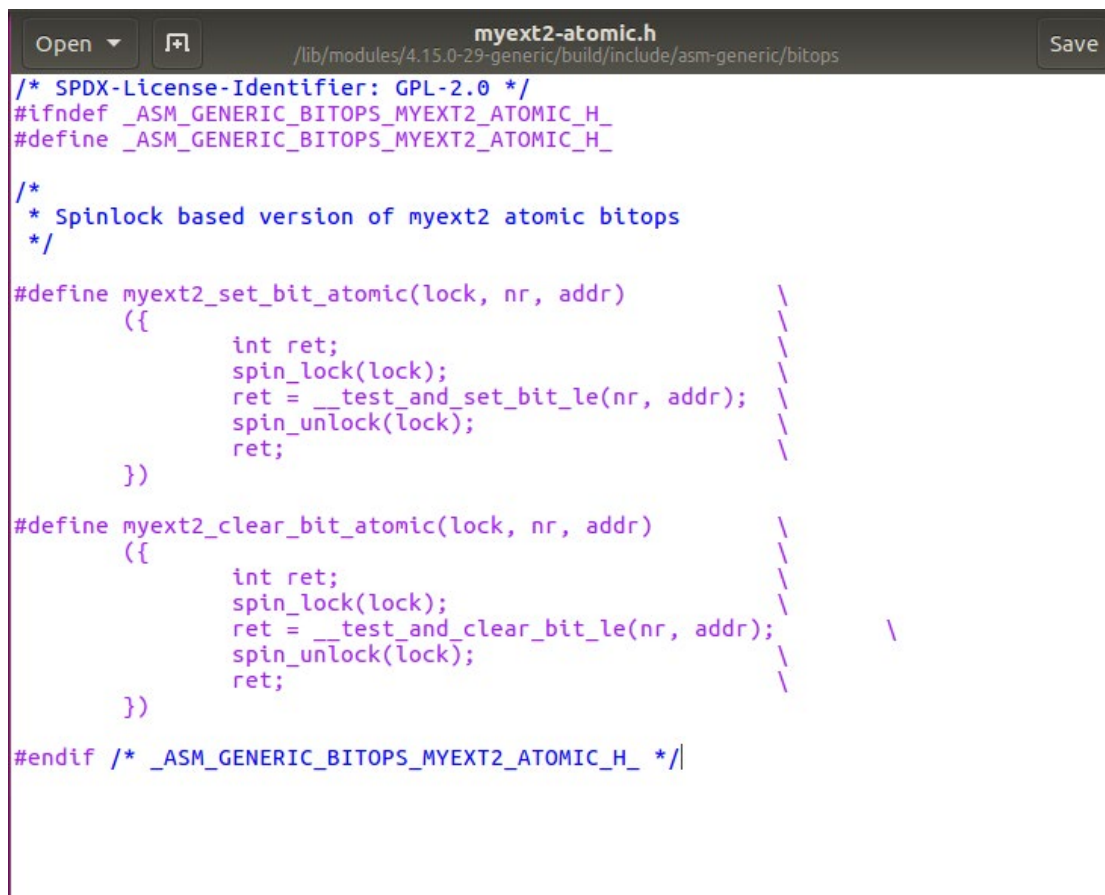
#define MYEXT2_SB_MAGIC_OFFSET 0x38
#define MYEXT2_SB_BLOCKS_OFFSET 0x04
#define MYEXT2_SB_BSIZE_OFFSET 0x18

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 */

```

替换后的 myext2-atomic.h 如图所示：



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

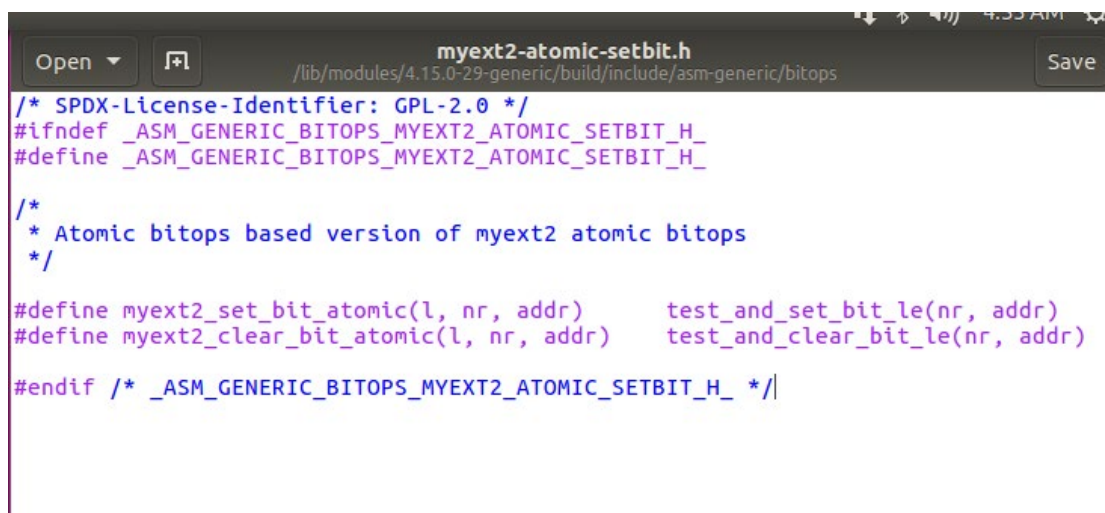
/*
 * Spinlock based version of myext2 atomic bitops
 */

#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_ */
```

替换后的 myext2-atomic-setbit.h 如图所示：



```
/* 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_ */
```

添加包含后的 bitops.h（两个）和 magic.h 如图所示：

Open bitops.h Save

/lib/modules/4.15.0-29-generic/build/include/asm-generic

```
/* 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>

/*modified by zijin*/
#include <asm-generic/bitops/myext2-atomic.h>

#endif /* __ASM_GENERIC_BITOPS_H */
```

```
bitops.h (/lib/modules/4.15.0-29-generic/build/arch/x86/include/asm) - gedit
bitops.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

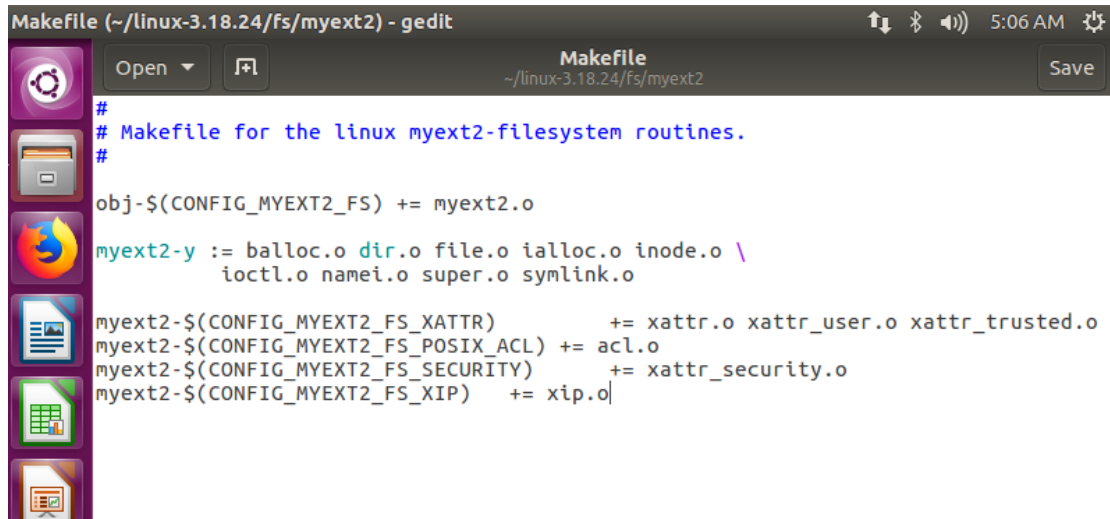
#include <linux/compiler.h>
#include <asm/alternative.h>
#include <asm/rmwcc.h>
#include <asm/barrier.h>
/*modified by zijin*/
#include <asm-generic/bitops/myext2-atomic-setbit.h>

#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
```

```
magic.h (/lib/modules/4.15.0-29-generic/build/include/uapi/linux) - gedit
magic.h
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

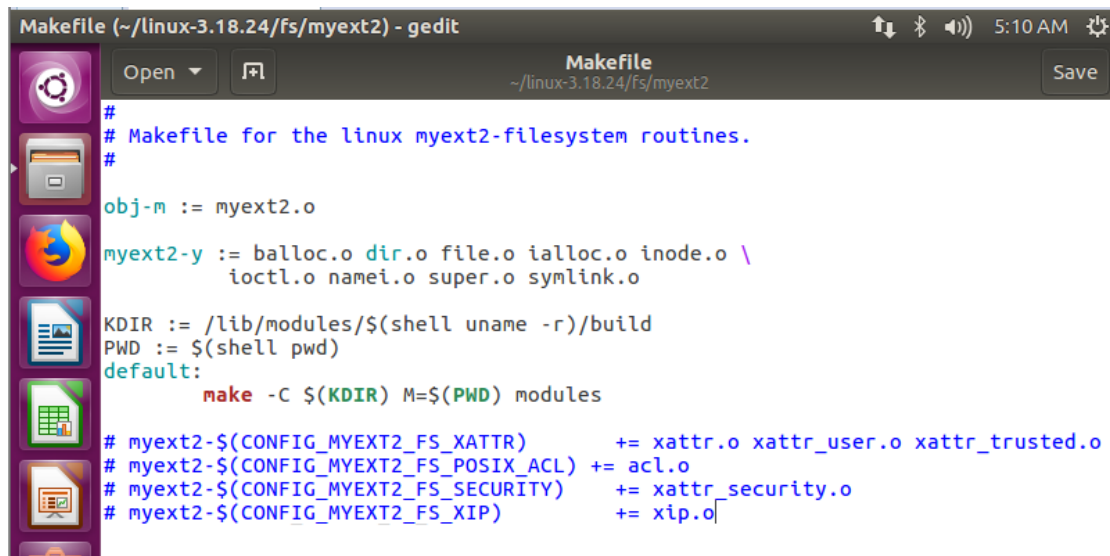
/*modified by zijin*/
#define MYEXT2_SUPER_MAGIC 0xEF53
#define ADFS_SUPER_MAGIC 0xadf5
#define AFFS_SUPER_MAGIC 0xadff
#define AFS_SUPER_MAGIC 0x5346414F
#define AUTofs_SUPER_MAGIC 0x0187
#define CODA_SUPER_MAGIC 0x73757245
#define CRAMFS_MAGIC 0x28cd3d45 /* some random number */
#define CRAMFS_MAGIC_WEND 0x453dcd28 /* magic number with the wrong
endianess */
#define DEBUGFS_MAGIC 0x64626720
#define SECURITYFS_MAGIC 0x73636673
#define SELINUX_MAGIC 0xf97cfff8c
#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
```

源代码的修改工作到此结束。接下来修改 myext2/Makefile 文件，修改前的 Makefile 文件如下：



```
#  
# Makefile for the linux myext2-filesystem routines.  
#  
obj-$(CONFIG_MYEXT2_FS) += myext2.o  
  
myext2-y := balloc.o dir.o file.o ialloc.o inode.o \  
          ioctl.o namei.o super.o symlink.o  
  
myext2-$(CONFIG_MYEXT2_FS_XATTR)      += xattr.o xattr_user.o xattr_trusted.o  
myext2-$(CONFIG_MYEXT2_FS_POSIX_ACL) += acl.o  
myext2-$(CONFIG_MYEXT2_FS_SECURITY)  += xattr_security.o  
myext2-$(CONFIG_MYEXT2_FS_XIP)      += xip.o
```

修改后的 makefile 如下:

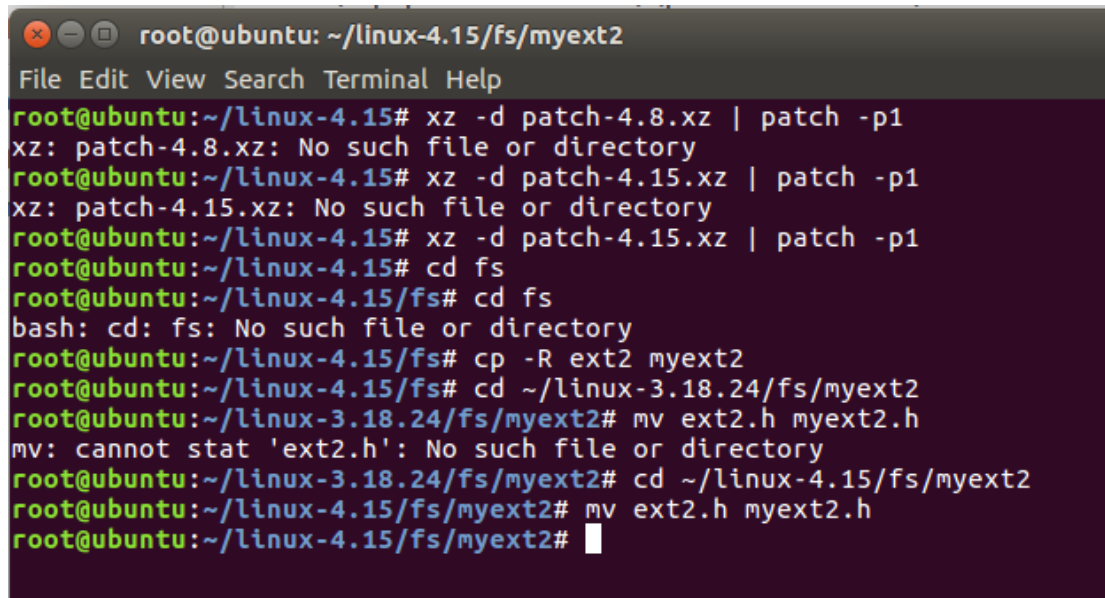


```
#  
# Makefile for the linux myext2-filesystem routines.  
#  
obj-m := myext2.o  
  
myext2-y := balloc.o dir.o file.o ialloc.o inode.o \  
          ioctl.o namei.o super.o symlink.o  
  
KDIR := /lib/modules/$(shell uname -r)/build  
PWD := $(shell pwd)  
default:  
    make -C $(KDIR) M=$(PWD) modules  
  
# myext2-$(CONFIG_MYEXT2_FS_XATTR)      += xattr.o xattr_user.o xattr_trusted.o  
# myext2-$(CONFIG_MYEXT2_FS_POSIX_ACL) += acl.o  
# myext2-$(CONFIG_MYEXT2_FS_SECURITY)  += xattr_security.o  
# myext2-$(CONFIG_MYEXT2_FS_XIP)      += xip.o
```

Makefile 编辑完成后, 执行 make, 可以看出执行成功 (注: 这里我试图使用 4.15 内核重新进行实验, 因此将之前的步骤重复一遍之后, 在 4.15 的环境下进行了 make):


```
root@ubuntu:~/linux-4.15/fs/myext2# make
make -C /lib/modules/4.15.0-29-generic/build M=/home/rikka/linux-4.15/fs/myext2
modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-29-generic'
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev,
libelf-devel or elfutils-libelf-devel"
CC [M] /home/rikka/linux-4.15/fs/myext2/balloc.o
CC [M] /home/rikka/linux-4.15/fs/myext2/dir.o
CC [M] /home/rikka/linux-4.15/fs/myext2/file.o
CC [M] /home/rikka/linux-4.15/fs/myext2/ialloc.o
CC [M] /home/rikka/linux-4.15/fs/myext2/inode.o
CC [M] /home/rikka/linux-4.15/fs/myext2/ioctl.o
CC [M] /home/rikka/linux-4.15/fs/myext2/namei.o
CC [M] /home/rikka/linux-4.15/fs/myext2/super.o
CC [M] /home/rikka/linux-4.15/fs/myext2/symlink.o
LD [M] /home/rikka/linux-4.15/fs/myext2/mymyext2.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/rikka/linux-4.15/fs/myext2/mymyext2.mod.o
LD [M] /home/rikka/linux-4.15/fs/myext2/mymyext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
```

(注：当时转移到 4.15 后执行的部分命令如下：)



```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
root@ubuntu:~/linux-4.15# xz -d patch-4.8.xz | patch -p1
xz: patch-4.8.xz: No such file or directory
root@ubuntu:~/linux-4.15# xz -d patch-4.15.xz | patch -p1
xz: patch-4.15.xz: No such file or directory
root@ubuntu:~/linux-4.15# xz -d patch-4.15.xz | patch -p1
root@ubuntu:~/linux-4.15# cd fs
root@ubuntu:~/linux-4.15/fs# cd fs
bash: cd: fs: No such file or directory
root@ubuntu:~/linux-4.15/fs# cp -R ext2 myext2
root@ubuntu:~/linux-4.15/fs# cd ~/linux-3.18.24/fs/myext2
root@ubuntu:~/linux-3.18.24/fs/myext2# mv ext2.h myext2.h
mv: cannot stat 'ext2.h': No such file or directory
root@ubuntu:~/linux-3.18.24/fs/myext2# cd ~/linux-4.15/fs/myext2
root@ubuntu:~/linux-4.15/fs/myext2# mv ext2.h myext2.h
root@ubuntu:~/linux-4.15/fs/myext2#
```

(转移到 4.15 后重新执行的 substitute.sh：)

```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
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
```

内核编译完毕后，插入 myext2.ko 模块：

```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2#
```

执行 `cat /proc/filesystems |grep myext2` 命令，发现文件系统加载成功：

```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2# cat /proc/filesystem | grep myext2
cat: /proc/filesystem: No such file or directory
root@ubuntu:~/linux-4.15/fs/myext2# cat /proc/filesystems | grep myext2
myext2
root@ubuntu:~/linux-4.15/fs/myext2#
```

此时，设置当前目录为主目录：

```
myext2
root@ubuntu:~/linux-4.15/fs/myext2# cd .
root@ubuntu:~/linux-4.15/fs/myext2# cd ~
root@ubuntu:~#
```

挂载文件系统:

```
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2# cat /proc/filesystem | grep myext2
cat: /proc/filesystem: No such file or directory
root@ubuntu:~/linux-4.15/fs/myext2# cat /proc/filesystems | grep myext2
myext2
root@ubuntu:~/linux-4.15/fs/myext2# cd .
root@ubuntu:~/linux-4.15/fs/myext2# cd ~
root@ubuntu:~# 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, 0.00114659 s, 915 MB/s
root@ubuntu:~# mount -t myext2 -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.
root@ubuntu:~# /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:~# mount -t myext2 -o loop ./myfs /mnt
```

测试 mount:

```
root@ubuntu:~# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:~# 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=4042504k,nr_inodes=1010626,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=814444k,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/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
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/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
```



```

systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,time
out=0,minproto=5,maxproto=5,direct,pipe_ino=20976)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_i
d=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=814444k,mode=7
00,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,u
ser_id=0,group_id=0)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
/home/rikka/myfs on /mnt type myext2 (rw,relatime,errors=continue)

```

在第二张图中，我们可以看到 myext2 文件系统已经被成功挂载。

本部分实验结束前，卸载该文件系统和 myext2 模块：

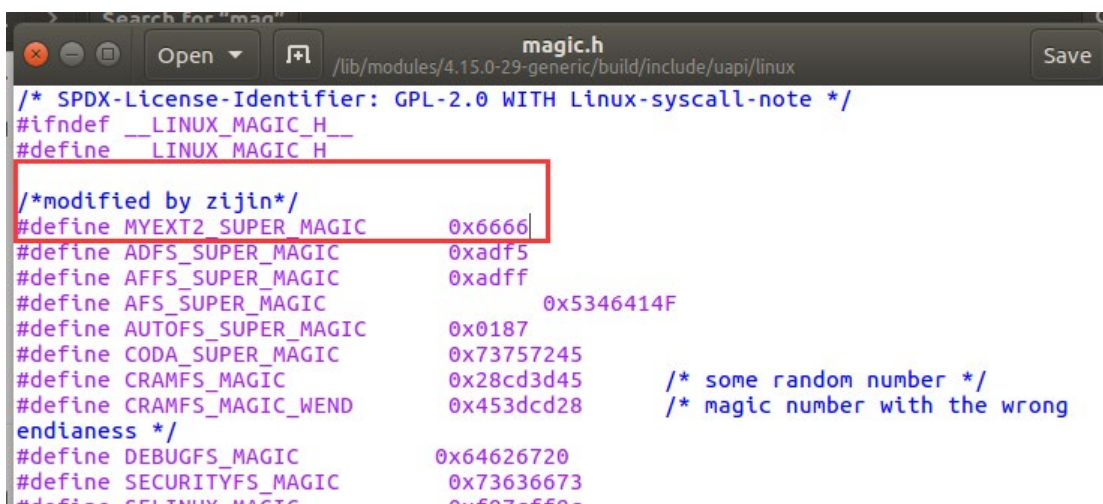
```

root@ubuntu:~# umount /mnt
No command 'umount' found, did you mean:
  Command 'umount' from package 'mount' (main)
umount: command not found
root@ubuntu:~# umountr /mnt
umountr: command not found
root@ubuntu:~# umount /mnt
root@ubuntu:~# rmmod myext2
root@ubuntu:~#

```

- 修改 myext2 的 magic number

在上面实验的基础上，找到 myext2 的 magic number（注：magic.h 很明显是指本机内核中的，而非素材内核中的 magic.h），并将其改为 0x6666：



```

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

/*modified by zijin*/
#define MYEXT2_SUPER_MAGIC 0x6666
#define ADFS_SUPER_MAGIC 0xadf5
#define AFS_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

```

修改后，重新编译内核模块并插入模块：

```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
root@ubuntu:~/linux-4.15/fs/myext2# make
make -C /lib/modules/4.15.0-29-generic/build M=/home/rikka/linux-4.15/fs/myext2
modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-29-generic'
makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev,
libelf-devel or elfutils-libelf-devel"
CC [M] /home/rikka/linux-4.15/fs/myext2/balloc.o
CC [M] /home/rikka/linux-4.15/fs/myext2/dir.o
CC [M] /home/rikka/linux-4.15/fs/myext2/file.o
CC [M] /home/rikka/linux-4.15/fs/myext2/ialloc.o
CC [M] /home/rikka/linux-4.15/fs/myext2/inode.o
CC [M] /home/rikka/linux-4.15/fs/myext2/ioctl.o
CC [M] /home/rikka/linux-4.15/fs/myext2/namei.o
CC [M] /home/rikka/linux-4.15/fs/myext2/super.o
CC [M] /home/rikka/linux-4.15/fs/myext2/symlink.o
LD [M] /home/rikka/linux-4.15/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
LD [M] /home/rikka/linux-4.15/fs/myext2/myext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
root@ubuntu:~/linux-4.15/fs/myext2#
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2#
```

从“学在浙大”下载 ChangeMN.c 文件:

```
ChangeMN.c
Save
#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\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\n",buf[0x438],buf[0x439]);

    while(ret -- 2048)
}
```

挂载文件系统并测试 changeMN 文件：

```
root@ubuntu: ~
File Edit View Search Terminal Help
root@ubuntu:~# 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, 0.00234513 s, 447 MB/s
root@ubuntu:~# /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:~# gcc -o changeMN ChangeMN.c
ChangeMN.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
root@ubuntu:~# ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
root@ubuntu:~#
```

这里发现了两个问题：第一，changeMN 后面的参数是没有必要的，因为源代码中根本没有接收命令行参数的设定，打开 ./myfs 也是固定地写在代码中的（理论上，应当接收一个文件名参数，这样无论 of 的文件名是什么，程序都可以运行。考虑到这一瑕疵并没有影响实验进程，因此并没有勘误）；第二，程序输出的 previous magic number 为 ‘53ef’，并不是我们此前定义的 ‘ef53’。考察 ChangeMN.c 源代码，发现：

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

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

此处源代码默认 ef53 这两个字节在文件中是顺序存储的。但是我们知道，部分操作系统在组织内存的时候存在“小端模式”——即，数据的高字节保存在内存的高地址中，而数据的低字节保存在内存的低地址中。本例中读取的文件为二进制文件，ef53 在内存中实际上以 ‘53ef’ 的方式存储。而 buf 是顺序读取的，才会出现这种颠倒的情况。

了解以上问题的原因后，继续进行实验。执行 mount 命令：

```

root@ubuntu:~# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:~# 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=4042504k,nr_inodes=1010626,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=814444k,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/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
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/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpu)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=20976)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=814444k,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
/home/rikka/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:~#

```

发现 myext2 被成功挂载。

这之后卸载/mnt，执行命令 `sudo mount -t ext2 -o loop ./fs.new /mnt` 重新挂载基于 ext2 文件系统的块存储/mnt，然后再测试 mount：

```

root@ubuntu:~# 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:~# rmmod myext2
root@ubuntu:~#

```

发现报错。该错误是两个文件系统 magic number 不匹配导致的。

本部分实验结束前，执行 `rmmod myext2` 命令移除 `myext2` 模块。

- 修改文件系统操作

对于 `mkmod` 函数，在 `myext2` 中作如下修改：

```

static int myext2_mknod (struct inode * dir, struct dentry *dentry, umode_t
mode, dev_t rdev)
{
    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_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;
}
*/
}

```


修改完成后，重新 make 并插入模块：

```
root@ubuntu: ~/linux-4.15/fs/myext2
File Edit View Search Terminal Help
root@ubuntu:~/linux-4.15/fs/myext2# make
make -C /lib/modules/4.15.0-29-generic/build M=/home/rikka/linux-4.15/fs/myext2
modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-29-generic'
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev,
libelf-devel or elfutils-libelf-devel"
  CC [M]  /home/rikka/linux-4.15/fs/myext2/balloc.o
  CC [M]  /home/rikka/linux-4.15/fs/myext2/namei.o
  LD [M]  /home/rikka/linux-4.15/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
  CC      /home/rikka/linux-4.15/fs/myext2/myext2.mod.o
  LD [M]  /home/rikka/linux-4.15/fs/myext2/myext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-29-generic'
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2#
```

插入模块后进行测试：

```
root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2# cd ~
root@ubuntu:~# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# mknod myfifo p
mknod: myfifo: Operation not permitted
root@ubuntu:/mnt#
```

可以看到，mknod 操作不被允许，我们的裁剪取得了预期的效果。

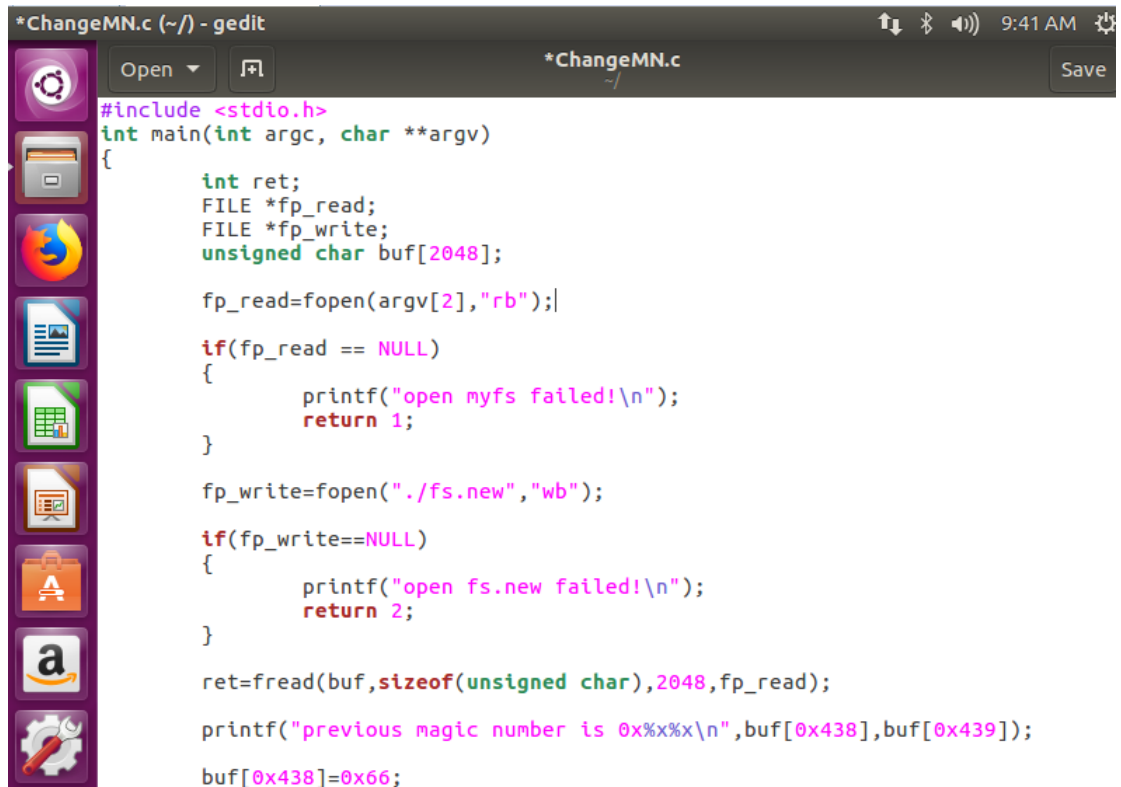
- 添加文件系统创建工具

首先，编辑 mkfs.myext2 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
```

修改 ChangeMN.c 的源代码，将 main 设置为可以接受命令行参数的模式，第二个参数作为 fopen 的文件名：



```
*ChangeMN.c (~/) - gedit
*ChangeMN.c
~/

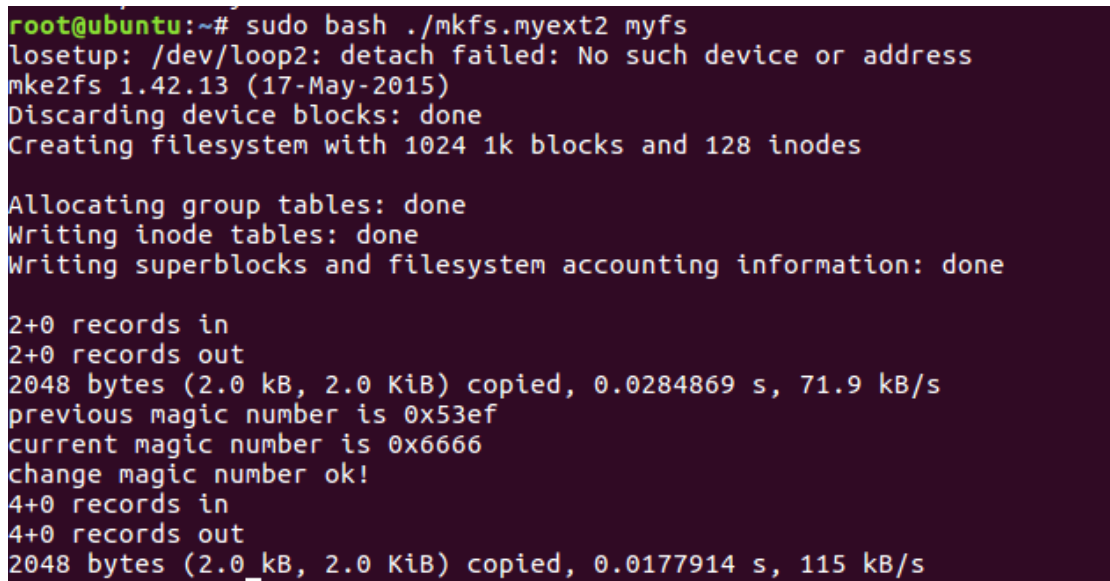
#include <stdio.h>
int main(int argc, char **argv)
{
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];

    fp_read=fopen(argv[2],"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;
```

插入模块，进行测试：



```
root@ubuntu:~# 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, 0.0284869 s, 71.9 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, 0.0177914 s, 115 kB/s
```

执行 mount:


```

for more details see mount(8).
root@ubuntu:~# sudo mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:~# 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=4042472k,nr_inodes=
1010618,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=6
0,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=814444k,mode=
55)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,no
exec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=51
0k)
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,r
latime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=sys
temd)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=24,pg
rp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=19290)
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)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relati
me,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=81444
4k,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,r
elatime,user_id=0,group_id=0)
/home/rikka/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:~#

```

发现 make 成功了，但没有出现/dev/loop。分析 shell 脚本：

```
~/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 ↵
```

这里与教材上不一样的，以本实验指导为准。 ↵

可看到/dev/loop2 文件夹的相关操作，推测所谓/dev/loop 当为/dev/loop2，实验参考出现了错漏。而所谓sudo mount -t myext2 -o loop ./myfs /mnt 命令，使用的是/home/rikka 下的 myfs，并不是/dev/loop2 下的 myfs，故命令应当为 sudo mount -t myext2 -o loop /dev/loop2/myfs

/mnt, 实验参考的表述依然是存在问题的。

- 修改加密文件系统的 read 和 write 操作

编写两个加密函数如下。要点：涉及内核态指针访问时，须将数据拷贝到用户态下编辑，再写回内核态，否则会出现内存访问错误：

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

ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len,
loff_t *ppos)
{
    char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
    int i;
    ssize_t ret = new_sync_read(filp, mybuf, len, ppos);
    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;
}
```

配置结构体如下：

```
const struct file_operations myext2_file_operations = {
    .llseek      = generic_file_llseek,
    .read        = new_sync_read_crypt,
    .write       = new_sync_write_crypt,
    .read_iter   = generic_file_read_iter,
    .write_iter  = generic_file_write_iter,
    .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 = iter_file_splice_write,
};
```

修改后执行 make，并配置文件系统（注：此时我处理 4.15 内核遇到了困难，尝试多次无法解决之后，换用 3.18.24 内核）

```
root@ubuntu: ~
File Edit View Search Terminal Help
root@ubuntu:~# ./mkfs.myext2 myfs
bash: ./mkfs.myext2: Permission denied
root@ubuntu:~# sudo ./mkfs.myext2 myfs
sudo: ./mkfs.myext2: command not found
root@ubuntu:~# 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, 0.028397 s, 72.1 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, 0.0199269 s, 103 kB/s
root@ubuntu:~# mount -t myext2 -o loop ./fs.new /mnt/

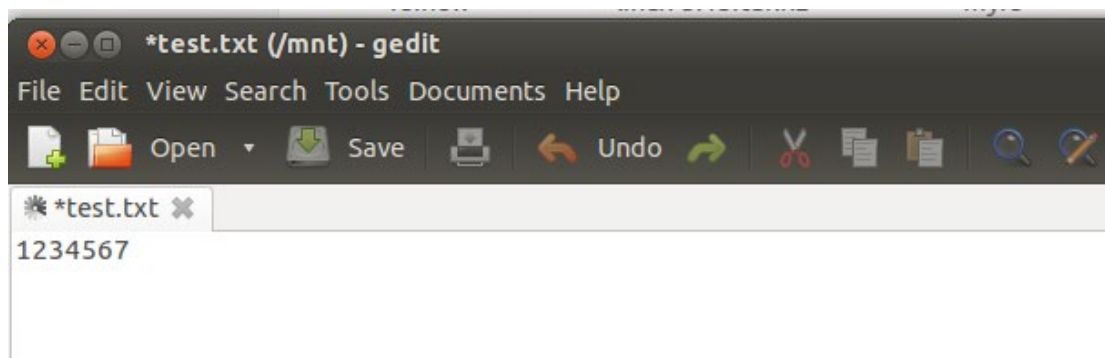
2+0 records in
2+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.028397 s, 72.1 kB/s
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
4+0 records in
4+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.0199269 s, 103 kB/s
root@ubuntu:~# mount -t myext2 -o 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:~# mount -t myext2 -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:~# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:~# cd /mnt
```

在 mnt 下新建文件并编辑文件:

```
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# cat test.txt
cat: test.txt: No such file or directory
root@ubuntu:/mnt# touch test.txt
```



在 mnt 下查看文件后，在使用 cp 命令 copy 到主目录下查看：

```
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# nano test.txt
root@ubuntu:/mnt# cat test.txt
1234567
root@ubuntu:/mnt# cp test.txt ~
root@ubuntu:/mnt# cat ~/test.txt
1234567
root@ubuntu:/mnt#
```

发现文件处于未加密状态（或者说，被成功解密）。但在使用文件管理器复制查看时：



显示出的是已经加密的文本。

这时我们把魔数改回来,再重新编译：

```
root@ubuntu:~# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# echo "1234567" > test2.txt
root@ubuntu:/mnt# cat test2.txt
1234567
root@ubuntu:/mnt# cd
root@ubuntu:~# umount /mnt
root@ubuntu:~# mount -t ext2 -o loop ./myfs /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# cat test2.txt
JKLMNOP#root@ubuntu:/mnt#
```

可以看出，此时即使使用 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。

3. 测试程序运行结果截图

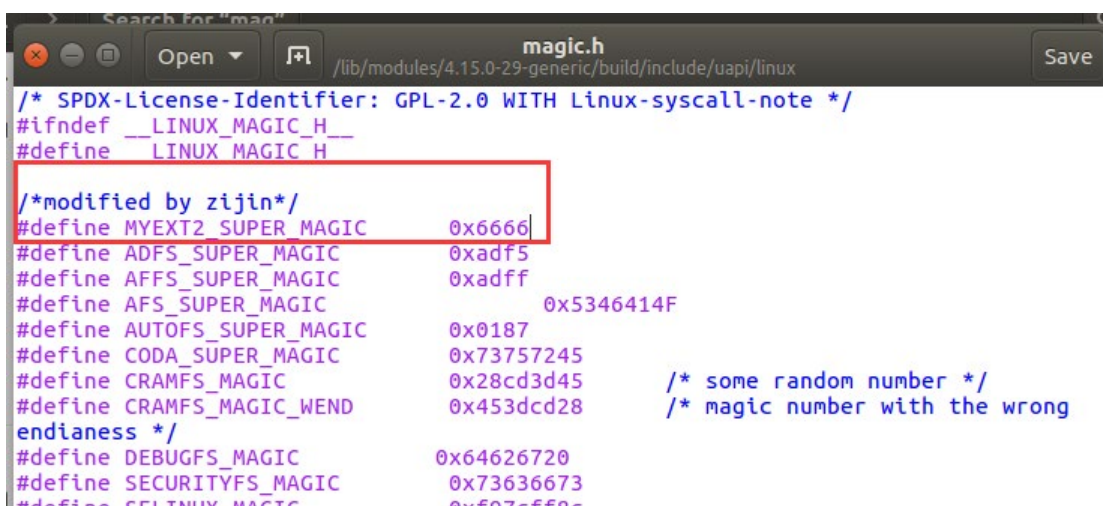
- 第一部分

```

systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,time
out=0,minproto=5,maxproto=5,direct,pipe_ino=20976)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_i
d=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=814444k,mode=7
00,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,u
ser_id=0,group_id=0)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
/home/rikka/myfs on /mnt type myext2 (rw,relatime,errors=continue)

```

● 第二部分

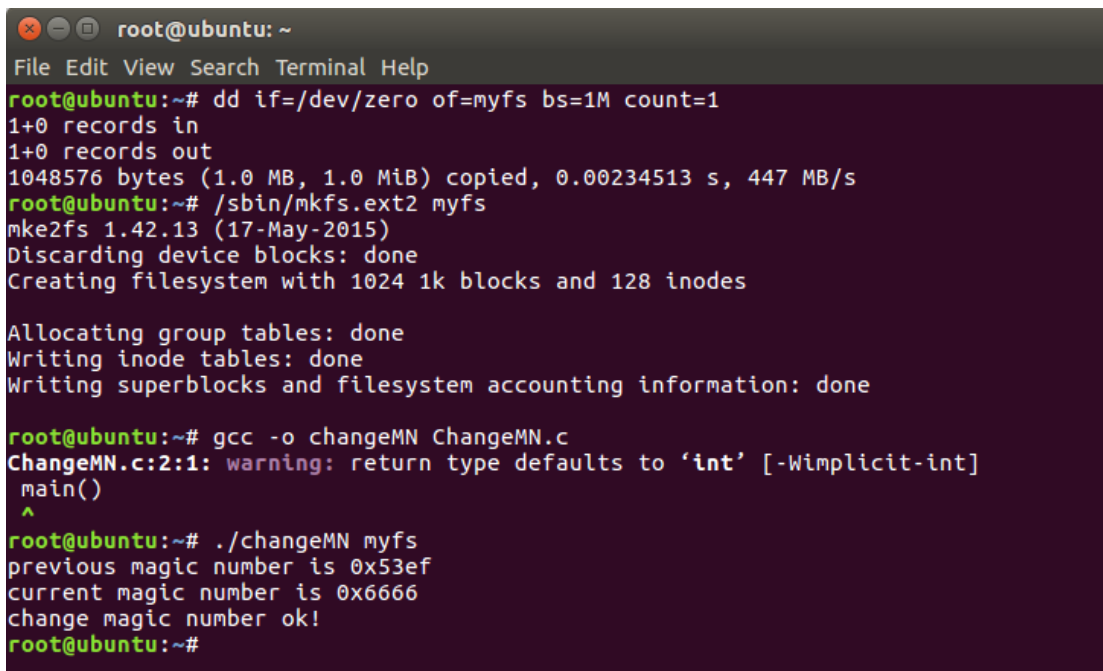


```

Search for "mag"
magic.h
/lib/modules/4.15.0-29-generic/build/include/uapi/linux
Save
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

/*modified by zijin*/
#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
endianness */
#define DEBUGFS_MAGIC          0x64626720
#define SECURITYFS_MAGIC       0x73636673
#define SELINUX_MAGIC          0x60706566

```



```

root@ubuntu: ~
File Edit View Search Terminal Help
root@ubuntu:~# 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, 0.00234513 s, 447 MB/s
root@ubuntu:~# /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:~# gcc -o changeMN ChangeMN.c
ChangeMN.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
root@ubuntu:~# ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
root@ubuntu:~#

```

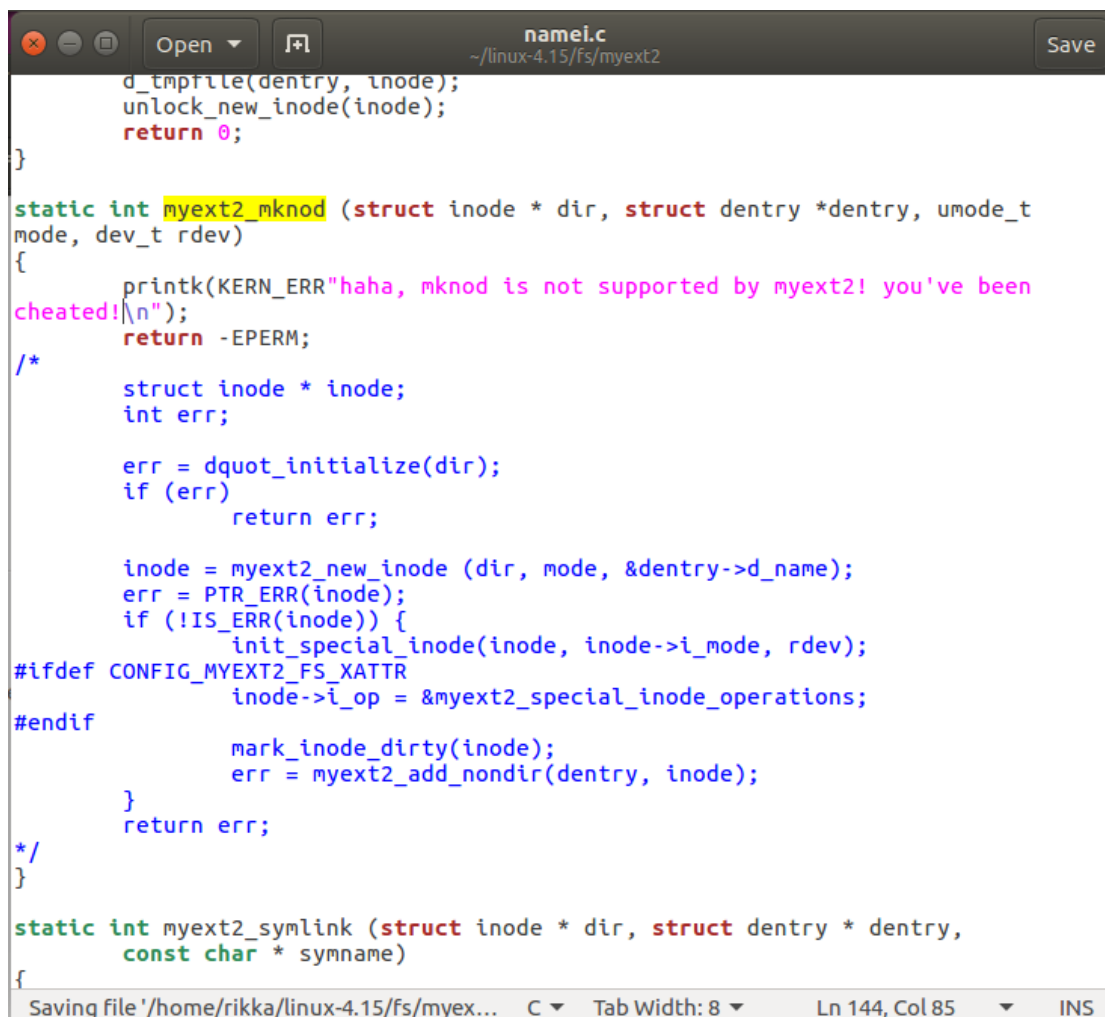


```

cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugepages)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=20976)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=814444k,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
/home/rikka/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:~#

```

● 第三部分



```

namei.c
~/linux-4.15/fs/myext2
Save

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

static int myext2_mknode (struct inode * dir, struct dentry *dentry, umode_t
mode, dev_t rdev)
{
    printk(KERN_ERR"haha, mknode 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)
{

```

Saving file '/home/rikka/linux-4.15/fs/myex... C Tab Width: 8 Ln 144, Col 85 INS

```

root@ubuntu:~/linux-4.15/fs/myext2# insmod myext2.ko
root@ubuntu:~/linux-4.15/fs/myext2# cd ~
root@ubuntu:~# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# mknod myfifo p
mknod: myfifo: Operation not permitted
root@ubuntu:/mnt#

```

● 第四部分

```

#!/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

```

```

*ChangeMN.c (~/) - gedit
#include <stdio.h>
int main(int argc, char **argv)
{
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];

    fp_read=fopen(argv[2],"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;
}

```



```

root@ubuntu:~# 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, 0.0284869 s, 71.9 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, 0.0177914 s, 115 kB/s

ime,pids)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=24,pg
rp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=19290)
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)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relati
me,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=81444
4k,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)
gvfsd-fuse on /home/rikka/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,r
elatime,user_id=0,group_id=0)
/home/rikka/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:~#

```

- 第五部分

```

root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# nano test.txt
root@ubuntu:/mnt# cat test.txt
1234567
root@ubuntu:/mnt# cp test.txt ~
root@ubuntu:/mnt# cat ~/test.txt
1234567
root@ubuntu:/mnt#

```



```

test.txt
~/
JKLMNOP#

```

```

root@ubuntu:~# mount -t myext2 -o loop ./myfs /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# echo "1234567" > test2.txt
root@ubuntu:/mnt# cat test2.txt
1234567
root@ubuntu:/mnt# cd
root@ubuntu:~# umount /mnt
root@ubuntu:~# mount -t ext2 -o loop ./myfs /mnt
root@ubuntu:~# cd /mnt
root@ubuntu:/mnt# cat test2.txt
JKLMNOP#root@ubuntu:/mnt#

```

4. 结果分析

- 为什么使用 `cp` 命令 `copy` 到主目录的文件就没有被加密，而使用文件管理器图形界面 `copy` 到主目录的文件就被加密了呢？

答：因为 `cp` 命令是通过先 `read` 再 `write` 实现的，在 `myext2` 调用 `myext2` 的 `read` 函数读——在 `ext2` 文件系统下调用 `ext2` 的 `write` 函数写并不会影响内容。而文件管理器的复制是直接将在 `/mnt/test.txt` 的二进制数据写入 `~/test.txt`，期间没有调用 `myext2` 的读函数，因此此时的 `test.txt` 是未被解密的状态。

- 为什么即使使用 `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呢？

答：一开始我们在 `myext2` 文件系统的 `/mnt` 目录下创建的 `test.txt`，是基于 `myext2` 的 `write` 函数写入的，因此其数据是加密的。但是，由于 `myext2` 的 `magic number` 和 `ext2` 一致，使用 `./myfs` 将 `/mnt` 重新 `mount` 为 `ext2` 文件系统的命令并不会报错，而是成功执行（`magic number` 不一致时命令会出错）。`/mnt` 变为 `ext2` 格式后，里面的 `test.txt` 也会被当作 `ext2` 文件系统的文件处理，读出来的自然也是未解密的数据了。

5. 源程序

[修改后的 `file.c`]

```

1.  /*
2.   *  linux/fs/myext2/file.c
3.   *
4.   *  Copyright (C) 1992, 1993, 1994, 1995
5.   *  Remy Card (card@masi.ibp.fr)
6.   *  Laboratoire MASI - Institut Blaise Pascal
7.   *  Universite Pierre et Marie Curie (Paris VI)
8.   *
9.   *  from
10.  *

```

```

11. * linux/fs/minix/file.c
12. *
13. * Copyright (C) 1991, 1992 Linus Torvalds
14. *
15. * myext2 fs regular file handling primitives
16. *
17. * 64-bit file support on 64-bit platforms by Jakub Jelinek
18. * (jj@sunsite.ms.mff.cuni.cza)
19. */
20.
21. #include <linux/time.h>
22. #include <linux/pagemap.h>
23. #include <linux/quotaops.h>
24. #include "myext2.h"
25. #include "xattr.h"
26. #include "acl.h"
27.
28. /*
29. * Called when filp is released. This happens when all file descriptors
30. * for a single struct file are closed. Note that different open() calls
31. * for the same file yield different struct file structures.
32. */
33. ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t len, loff_t *ppos)
34. {
35.     char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
36.     int i;
37.     copy_from_user(mybuf,buf,len);
38.     for(i = 0; i < len; i++)
39.         mybuf[i] = (mybuf[i] + 25) % 128;
40.     printk("haha encrypt %ld\n",len);
41.     return new_sync_write(filp, mybuf, len, ppos);
42. }
43.
44. ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len, loff_t *ppos)
45. {
46.     char* mybuf = (char*)kmalloc(sizeof(char)*len,GFP_KERNEL);
47.     int i;
48.     ssize_t ret = new_sync_read(filp, mybuf, len, ppos);
49.     for(i = 0; i < len; i++)
50.         mybuf[i] = (mybuf[i] - 25 + 128) % 128;
51.     copy_to_user(buf,mybuf,len);
52.     printk("haha encrypt %ld\n",len);

```

```

53.     return ret;
54. }
55. static int myext2_release_file (struct inode * inode, struct file * filp)
56. {
57.     if (filp->f_mode & FMODE_WRITE) {
58.         mutex_lock(&MYEXT2_I(inode)->truncate_mutex);
59.         myext2_discard_reservation(inode);
60.         mutex_unlock(&MYEXT2_I(inode)->truncate_mutex);
61.     }
62.     return 0;
63. }
64.
65. int myext2_fsync(struct file *file, loff_t start, loff_t end, int datasync)
66. {
67.     int ret;
68.     struct super_block *sb = file->f_mapping->host->i_sb;
69.     struct address_space *mapping = sb->s_bdev->bd_inode->i_mapping;
70.
71.     ret = generic_file_fsync(file, start, end, datasync);
72.     if (ret == -EIO || test_and_clear_bit(AS_EIO, &mapping->flags)) {
73.         /* We don't really know where the IO error happened... */
74.         myext2_error(sb, __func__,
75.                     "detected IO error when writing metadata buffers");
76.         ret = -EIO;
77.     }
78.     return ret;
79. }
80.
81. /*
82.  * We have mostly NULL's here: the current defaults are ok for
83.  * the myext2 filesystem.
84.  */
85. const struct file_operations myext2_file_operations = {
86.     .llseek      = generic_file_llseek,
87.     .read        = new_sync_read_crypt,
88.     .write       = new_sync_write_crypt,
89.     .read_iter   = generic_file_read_iter,
90.     .write_iter  = generic_file_write_iter,
91.     .unlocked_ioctl = myext2_ioctl,
92. #ifdef CONFIG_COMPAT
93.     .compat_ioctl = myext2_compat_ioctl,
94. #endif
95.     .mmap        = generic_file_mmap,
96.     .open        = dquot_file_open,

```

```

97.     .release      = myext2_release_file,
98.     .fsync        = myext2_fsync,
99.     .splice_read   = generic_file_splice_read,
100.    .splice_write   = iter_file_splice_write,
101.};
102.
103.#ifdef CONFIG_MYEXT2_FS_XIP
104.const struct file_operations myext2_xip_file_operations = {
105.    .llseek          = generic_file_llseek,
106.    .read             = xip_file_read,
107.    .write            = xip_file_write,
108.    .unlocked_ioctl   = myext2_ioctl,
109.#ifdef CONFIG_COMPAT
110.    .compat_ioctl     = myext2_compat_ioctl,
111.#endif
112.    .mmap             = xip_file_mmap,
113.    .open             = dquot_file_open,
114.    .release          = myext2_release_file,
115.    .fsync            = myext2_fsync,
116.};
117.#endif
118.
119.const struct inode_operations myext2_file_inode_operations = {
120.#ifdef CONFIG_MYEXT2_FS_XATTR
121.    .setxattr         = generic_setxattr,
122.    .getxattr         = generic_getxattr,
123.    .listxattr        = myext2_listxattr,
124.    .removexattr      = generic_removexattr,
125.#endif
126.    .setattr          = myext2_setattr,
127.    .get_acl          = myext2_get_acl,
128.    .set_acl          = myext2_set_acl,
129.    .fiemap           = myext2_fiemap,
130.};

```

三、讨论、心得（20 分）

本次实验相较于上次实验花费的时间较少，但还是遇到了不少挑战。

第一个难点，是本机内核和内核源码的兼容性问题。一开始，我错误地理解了题意，使用的是 4.15 系统内核搭配 3.18.24 内核源码，很明显，这是不可行的，因为不能使用 4.15 的 make 工具编译 3.18.24 的内核源码；这之后，我尝试换用 4.15、3.7 内核源码，但又在测试加密文件系统阶段止步不前（保存文件时出现‘Bad Address’错误）。最终，我意识到可以通过在官网下载安装包的方式更新系统内核，于是我为 16.04 发行版安装了 3.18.24 内核，成功地完成了实验。不过我相信使用 4.15 内核也可以很好地完成实验，只是我没有找对方

法罢了。当然，也有可能是实验过程中犯了一些不易察觉的错误。

另一个难点是加密函数的编写。第一次尝试时，我受到 `char* mybuf = buf` 这行代码的影响，忘记了处于用户态的函数/程序代码没有权限操作内核栈中的数据，直接对 `mybuf/buf` 进行加/减操作，导致写文件时编辑器进程因内存访问错误被强制杀死（killed）。通过查阅资料和 ppt，我认识到了自己的错误，严格按照用户态和内核态的编程规范重写了安全的加密函数，解决了内存访问错误的问题。

总而言之，本次实验同课程内容的结合较为紧密，既有必要的实验指导又不乏启发性的重难点考查，丰富了我的知识储备，锻炼了我的能力，帮助我温习了书本概念，为今后的学习打下了良好的基础。

不过实验指导里面历史遗留问题真的很多……老师可以考虑修订一下。