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：添加一个加密系统

学生姓名：杨樾人 学号：3160104875

电子邮件地址：yangyueren@outlook.com

实验日期：2018 年 1 月 6 日

一、实验环境

填写您的计算机配置，操作系统环境，Linux 版本

阿里云服务器

1GHz

Ubuntu16.04

Linux-3.18.24

二、实验内容和结果及分析

实验设计思路

添加一个类似于 ext2，但对磁盘上的数据块进行加密的文件系统 myext2。实验主要内容：

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

实验步骤及截图

三、 实验步骤

提示：下面的操作步骤以 3.18.24 版本的内核为例，其它版本内核可能会有所区别。下面的操作用户需要 root 权限

2.1 添加一个类似 ext2 的文件系统 myext2

要添加一个类似 ext2 的文件系统 myext2，首先是确定实现 ext2 文件系统的内核源码是由哪些文件组成。Linux 源代码结构很清楚地 myext 告诉我们：fs/ext2 目录下的所有文件是属于 ext2 文件系统的。再检查一下这些文件所包含的头文件，可以初步总结出来 Linux 源

代码中属于 ext2 文件系统的有：

```
fs/ext2/acl.c
fs/ext2/acl.h
fs/ext2/balloc.c
fs/ext2/bitmap.c
fs/ext2/dir.c
fs/ext2/ext2.h
fs/ext2/file.c
……
include/linux/ext2_fs.h
```

接下来开始添加 myext2 文件系统的源代码到 Linux 源代码。把 ext2 部分的源代码克隆到 myext2 去，即复制一份以上所列的 ext2 源代码文件给 myext2 用。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 shell 下，执行如下操作：

```
#cd ~/linux-3.18.24   /* 内核源代码目录，假设内核源代码解压在主目录的 Linux-3.18.24 子目录中*/
#cd fs
#cp  -R ext2 myext2
#cd ~/linux-3.18.24/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
```

这样就完成了克隆文件系统工作的第一步——源代码复制。对于克隆文件系统来说，这样当然还远远不够，因为文件里面的数据结构名、函数名、以及相关的一些宏等内容还没有根据 myext2 改掉，连编译都通不过。

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

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”。

特别提示:

- 不要拷贝 word 文档中的 substitute.sh 脚本，在 Linux 环境下重新输入一遍，substitute.sh 脚本程序只能运行一次。ubuntu 环境: sudo bash substitute.sh。
- 先删除 fs/myext2 目录下的 *.o 文件，再运行脚本程序。
- 在下面的替换或修改内核代码时可以使用 gedit 编辑器，要注意大小写。

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”

```c
#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 */
```

```c
#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_ */

```

```
#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>

```
#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>

```
#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 /include/uapi/linux/magic.h 文件中添加：

#define MYEXT2_SUPER_MAGIC 0xEF53

```
#define EFS_SUPER_MAGIC      0x414A53
#define EXT2_SUPER_MAGIC     0xEF53
#define MYEXT2_SUPER_MAGIC   0xEF53      /* MYEXT2 */
#define EXT3_SUPER_MAGIC     0xEF53
#define XENFS_SUPER_MAGIC    0xabba1974
#define EXT4_SUPER_MAGIC     0xEF53
```

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/Makefile 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
```

编译内核模块的命令是 make，在 myext2 目录下执行命令：

#make

```
        make -C $(KDIR) M=$(PWD) modules                    [8/118]

root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# make
make -C /lib/modules/3.18.24/build M=/root/linux-3.18.24/fs/myext2
  modules
make[1]: Entering directory '/root/linux-3.18.24'
  CC [M]  /root/linux-3.18.24/fs/myext2/balloc.o
  CC [M]  /root/linux-3.18.24/fs/myext2/dir.o
  CC [M]  /root/linux-3.18.24/fs/myext2/file.o
  CC [M]  /root/linux-3.18.24/fs/myext2/ialloc.o
  CC [M]  /root/linux-3.18.24/fs/myext2/inode.o
  CC [M]  /root/linux-3.18.24/fs/myext2/ioctl.o
  CC [M]  /root/linux-3.18.24/fs/myext2/namei.o
  CC [M]  /root/linux-3.18.24/fs/myext2/super.o
  CC [M]  /root/linux-3.18.24/fs/myext2/symlink.o
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/linux-3.18.24/fs/myext2/myext2.mod.o
```

编译好模块后，使用 insmod 命令加载模块：

#insmod myext2.ko

```
  CC [M]  /root/linux-3.18.24/fs/myext2/super.o
  CC [M]  /root/linux-3.18.24/fs/myext2/symlink.o
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/linux-3.18.24/fs/myext2/myext2.mod.o
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.ko
make[1]: Leaving directory '/root/linux-3.18.24'
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# insmod mye
xt2.ko
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# cat /proc/
/filesystems | grep myext2
        myext2
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2#
```

查看一下 myext2 文件系统是否加载成功：

#cat /proc/filesystem |grep 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*/
```

```
  CC      /root/linux-3.18.24/fs/myext2/myext2.mod.o       [75/203]
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.ko
make[1]: Leaving directory '/root/linux-3.18.24'
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# insmod my$
xt2.ko
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# cat /proc$
/filesystems | grep myext2
        myext2
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# vim testm$
ext2.sh
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# chmod 777
testmyext2.sh
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# ./testmye$
t2.sh
        myext2
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00154835 s, 677 MB/s
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

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=908684k,nr_ino
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mo
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,m
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=order
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
```

```
des=227171,mode=755)                                                [0/203]
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mo
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,m
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=order
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relat
ime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,n
odev,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/perf_event type cgroup (rw,nosuid,nodev,n
oexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noex
ec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexe
c,relatime,cpuset)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec
,relatime,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noex
ec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=
34,pgrp=1,timeout=0,minproto=5,maxproto=5,direct)
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)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=204
908k,mode=700)
/root/linux-3.18.24/fs/myext2/myfs on /mnt type ext2 (rw,relatime)
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# cat Makefi
le
```

2.2 修改 myext2 的 magic number

在上面做的基础上。找到 myext2 的 magic number，并将其改为 0x6666：

3.18.24 内核版本，这个值在 include/uapi/linux/magic.h 文件中。

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

```
#define EFS_SUPER_MAGIC      0x414A53
#define EXT2_SUPER_MAGIC     0xEF53
#define MYEXT2_SUPER_MAGIC   0x6666      /* MYEXT2 */
#define EXT3_SUPER_MAGIC     0xEF53
#define XENFS_SUPER_MAGIC    0xabba1974
#define EXT4_SUPER_MAGIC     0xEF53
```

改动完成之后，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。

```
iZ2ze4u33ixw0vp6rh4fceZ# make
make -C /lib/modules/3.18.24/build M=/root/linux-3.18.24/fs/myext2
  modules
make[1]: Entering directory '/root/linux-3.18.24'
  CC [M]  /root/linux-3.18.24/fs/myext2/balloc.o
  CC [M]  /root/linux-3.18.24/fs/myext2/dir.o
  CC [M]  /root/linux-3.18.24/fs/myext2/file.o
  CC [M]  /root/linux-3.18.24/fs/myext2/ialloc.o
  CC [M]  /root/linux-3.18.24/fs/myext2/inode.o
  CC [M]  /root/linux-3.18.24/fs/myext2/ioctl.o
  CC [M]  /root/linux-3.18.24/fs/myext2/namei.o
  CC [M]  /root/linux-3.18.24/fs/myext2/super.o
  CC [M]  /root/linux-3.18.24/fs/myext2/symlink.o
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.ko
make[1]: Leaving directory '/root/linux-3.18.24'
iZ2ze4u33ixw0vp6rh4fceZ#
```

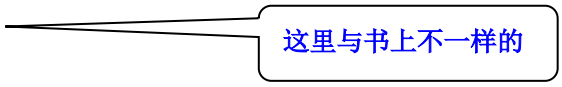
在我们测试这个部分之前，我们需要写个小程序 changeMN.c，来修改我们创建的 myfs 文件系统的 magic number。因为它必须和内核中记录 myext2 文件系统的 magic number 匹配，myfs 文件系统才能被正确地 mount。

changeMN.c 程序可以在课程网站中下载。这个程序经过编译后产生的可执行程序名字为 changeMN。

下面我们开始测试：

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

#/sbin/mkfs.ext2 myfs
#./changeMN myfs
#mount -t myext2 -o loop ./fs.new /mnt
#mount



…… on /mnt type myext2 (rw)
#sudo umount /mnt
sudo mount -t ext2 -o loop ./fs.new /mnt
mount: wrong fs type, bad option, bad superblock on /dev/loop0, …

rmmod myext2

```
iZ2ze4u33ixw0vp6rh4fceZ# umount /mnt
umount: /mnt: not mounted
iZ2ze4u33ixw0vp6rh4fceZ# 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.
```

```
iZ2ze4u33ixw0vp6rh4fceZ# vim testMy2.sh
iZ2ze4u33ixw0vp6rh4fceZ# ./testMy2.sh
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00222523 s, 471 MB/s
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

previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
mount: unknown filesystem type 'myext2'
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=908684k,nr_in$
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,m$
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,$
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=orde$
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nod$
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,si$
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=7$
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relat
ime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,n
odev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,
noexec,relatime,cpu,cpuacct)
```

2.3 修改文件系统操作

myext2 只是一个实验性质的文件系统，我们希望它只要能支持简单的文件操作即可。

因此在完成了 myext2 的总体框架以后，我们来修改掉 myext2 支持的一些操作，来加深对操作系统对文件系统的操作的理解。下面以裁减 myext2 的 mknod 操作为例，了解这个过程的实现流程。

Linux 将所有的对块设备、字符设备和命名管道的操作，都看成对文件的操作。mknod 操作是用来产生那些块设备、字符设备和命名管道所对应的节点文件。在 ext2 文件系统中它的实现函数如下：

```
fs/ext2/namei.c, line 144
144 static int ext2_mknod (struct inode * dir, struct dentry *dentry, int mode, dev_t
rdev)
145 {
146         struct inode * inode;
147         int err;
148
149         if (!new_valid_dev(rdev))
150                 return -EINVAL;
151
152         inode = ext2_new_inode (dir, mode);
153         err = PTR_ERR(inode);
154         if (!IS_ERR(inode)) {
155                 init_special_inode(inode, inode->i_mode, rdev);
156 #ifdef CONFIG_EXT2_FS_XATTR
157                 inode->i_op = &ext2_special_inode_operations;
158 #endif
159                 mark_inode_dirty(inode);
160                 err = ext2_add_nondir(dentry, inode);
161         }
162         return err;
163 }
```

它定义在结构 ext2_dir_inode_operations 中：

```
fs/ext2/namei.c, line 400
392 struct inode_operations ext2_dir_inode_operations = {
393         .create         = ext2_create,
394         .lookup         = ext2_lookup,
395         .link           = ext2_link,
396         .unlink         = ext2_unlink,
397         .symlink        = ext2_symlink,
398         .mkdir          = ext2_mkdir,
399         .rmdir          = ext2_rmdir,
400         .mknod          = ext2_mknod,
401         .rename         = ext2_rename,
402 #ifdef CONFIG_EXT2_FS_XATTR
```

```
403         .setxattr       = generic_setxattr,
404         .getxattr       = generic_getxattr,
405         .listxattr      = ext2_listxattr,
406         .removexattr    = generic_removexattr,
407 #endif
408         .setattr        = ext2_setattr,
409         .permission     = ext2_permission,
410 };
```

当然，从 ext2 克隆过去的 myext2 的 myext2_mknod，以及 myext2_dir_inode_operations 和上面的程序是一样的。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
   */
}
```

```
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 be
en cheated!\n");
    return -EPERM;
}
```

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。

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：

```
#mount –t myext2 –o loop ./fs.new /mnt
#cd /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。

```
iZ2ze4u33ixw0vp6rh4fceZ# ./testMy3.sh
mknod: myfifo: Operation not permitted
[    5.348439] audit: type=1400 audit(1546413874.277:6): apparmor=
"STATUS" operation="profile_load" name="/usr/sbin/ntpd" pid=393 co
mm="apparmor_parser"
[    5.418161] audit: type=1400 audit(1546413874.345:7): apparmor=
"STATUS" operation="profile_load" name="/usr/sbin/tcpdump" pid=403
 comm="apparmor_parser"
[    6.515096] random: nonblocking pool is initialized
[83858.229288] myext2: module verification failed: signature and/o
r  required key missing - tainting kernel
[84108.337152] EXT4-fs (loop0): mounting ext2 file system using th
e ext4 subsystem
[84108.337296] EXT4-fs (loop0): mounted filesystem without journal
. Opts: (null)
[84794.584695] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[84824.603890] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[84997.910905] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[85756.423599] haha, mknod is not supported by myext2! you've been
 cheated!
iZ2ze4u33ixw0vp6rh4fceZ#
```

可见，我们的裁减工作取得了预期的效果。

2.4. 添加文件系统创建工具

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

其实，前面的第一小节《添加一个和类似 ext2 的文件系统 myext2》和第二小节《修改 myext2 的 magic number》在测试实验结果的时候，已经陆陆续续地讲到了如何创建 myext2 文件系统。下面工作的主要目的就是将这些内容总结一下，制作出一个更快捷方便的 myext2 文件系统的创建工具：mkfs.myext2（名称上与 mkfs.ext2 保持一致）。

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

这里与教材上不一样的，以本实验指导为准。

第一行 表明是 shell 程序。
第三行 如果有程序用了/dev/loop2 了，就将它释放。
第四行 用 losetup 将第一个参数代表的文件装到/dev/loop2 上
第五行 用 mkfs.ext2 格式化/dev/loop2。也就是用 ext2 文件系统格式格式化我们的文件系统。
第六行 将文件系统的头 2K 字节的内容取出来，复制到 tmpfs 文件里面。
第七行 调用程序 changeMN 读取 tmpfs，复制到 fs.new，并且将 fs.new 的 magic number 改成 0x6666
第八行 再将 2K 字节的内容写回去。
第九行 把我们的文件系统从 loop2 中卸下来。
第十行 将临时文件删除。

我们发现 mkfs.myext2 脚本中的 changeMN 程序功能，与 2.2 节的 changeMN 功能不一样，请修改 changeMN.c 程序，以适合本节 mkfs.myext2 和下面测试的需要。

```c
main()
{
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];

    fp_read=fopen("./tmpfs","rb");  /* Change: from "./myfs" to "./tmpfs" */

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

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)
```

```
iZ2ze4u33ixw0vp6rh4fceZ# 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=908684k,nr_ino
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mo
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,m
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=order
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relat
ime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,n
odev,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/perf_event type cgroup (rw,nosuid,nodev,n
oexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noex
ec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexe
c,relatime,cpuset)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec
,relatime,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noex
ec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=
```

```
iZ2ze4u33ixw0vp6rh4fceZ# ./test4.sh
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00228137 s, 460 MB/s
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, 0.000997875 s, 2.1 MB/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, 0.00262184 s, 781 kB/s
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=908684k,nr_in$
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,m$
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,$
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=orde$
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
```

2.5 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_write_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。（//以 3.18 内核为例）

对 new_sync_ write _cryp 函数，可以做如下修改：

ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t len, loff_t *ppos)

{

char* mybuf = buf;

//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）

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

}

对 new_sync_read_cryp 函数，可以做如下修改：

ssize_t new_sync_read_crypt(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

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）

printk("haha encrypt %ld\n", len);

return ret;

}

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

static ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len, loff_t *ppo
{
        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 decrypt %ld\n", len);
        return ret;
}
```

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。

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

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@iZ2ze4u33ixw0vp6rh4fceZ:/mnt# cat test.txt
1234567
```

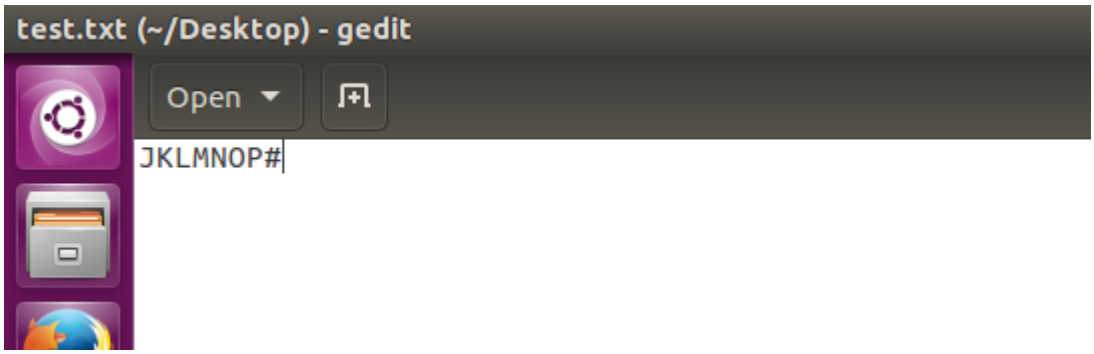
把 test.txt 文件复制到主目录下：cp test.txt ~ 。

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

```
root@iZ2ze4u33ixw0vp6rh4fceZ:~# cat test.txt
JKLMOP
```

使用文件管理器的复制，再查看结果？

在 Ubuntu 虚拟机下进行查看：



我们把之前的 magic number 改回 0xEF53。重新编译 myext2 模块，安装 myext2.ko 后，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

查看实验结果，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。

```
root@iZ2ze4u33ixw0vp6rh4fceZ:#~ ./finalTest.sh
1+0 records in
1+0 records out
1048576 bytes(1.0 MB, 1.0 MiB) copied, 0.00157132 s, 667 MB/s
make2fs 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

1234567
JKLMNOP#root@iZ2ze4u33ixw0vp6rh4fceZ:~#
```

至此，文件系统部分的实验已经全部完成了。通过本实验，你对 Linux 整个文件系统的运作流程，如何添加一个文件系统，以及如何修改 Linux 对文件系统的操作，有了比较深的了解。在本实验的基础上，你完全可以发挥自己的创造性，构造出自己的文件系统，然后将它添加到 Linux 中。

测试程序运行结果截图

p1

```
  CC      /root/linux-3.18.24/fs/myext2/myext2.mod.o      [75/203]
  LD [M]  /root/linux-3.18.24/fs/myext2/myext2.ko
make[1]: Leaving directory '/root/linux-3.18.24'
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# insmod my$
xt2.ko
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# cat /proc$
/filesystems | grep myext2
        myext2
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# vim testm$
ext2.sh
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# chmod 777
testmyext2.sh
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# ./testmye$
t2.sh
        myext2
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00154835 s, 677 MB/s
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

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=908684k,nr_ino
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mo
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,m
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=order
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
```

```
des=227171,mode=755)                                                [0/203]
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mo
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,m
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=order
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relat
ime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,n
odev,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/perf_event type cgroup (rw,nosuid,nodev,n
oexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noex
ec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexe
c,relatime,cpuset)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec
,relatime,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noex
ec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=
34,pgrp=1,timeout=0,minproto=5,maxproto=5,direct)
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)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=204
908k,mode=700)
/root/linux-3.18.24/fs/myext2/myfs on /mnt type ext2 (rw,relatime)
root@iZ2ze4u33ixw0vp6rh4fceZ:~/linux-3.18.24/fs/myext2# cat Makefi
le
```

p2

```
iZ2ze4u33ixw0vp6rh4fceZ# vim testMy2.sh
iZ2ze4u33ixw0vp6rh4fceZ# ./testMy2.sh
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00222523 s, 471 MB/s
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

previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
mount: unknown filesystem type 'myext2'
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=908684k,nr_in$
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,m$
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,$
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=orde$
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nod$
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,si$
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=7$
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relat
ime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,n
odev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,
noexec,relatime,cpu,cpuacct)
```

```
iZ2ze4u33ixw0vp6rh4fceZ# ./testMy3.sh
mknod: myfifo: Operation not permitted
[    5.348439] audit: type=1400 audit(1546413874.277:6): apparmor=
"STATUS" operation="profile_load" name="/usr/sbin/ntpd" pid=393 co
mm="apparmor_parser"
[    5.418161] audit: type=1400 audit(1546413874.345:7): apparmor=
"STATUS" operation="profile_load" name="/usr/sbin/tcpdump" pid=403
 comm="apparmor_parser"
[    6.515096] random: nonblocking pool is initialized
[83858.229288] myext2: module verification failed: signature and/o
r  required key missing - tainting kernel
[84108.337152] EXT4-fs (loop0): mounting ext2 file system using th
e ext4 subsystem
[84108.337296] EXT4-fs (loop0): mounted filesystem without journal
. Opts: (null)
[84794.584695] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[84824.603890] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[84997.910905] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[85756.423599] haha, mknod is not supported by myext2! you've been
 cheated!
iZ2ze4u33ixw0vp6rh4fceZ#
```

p4

```
iZ2ze4u33ixw0vp6rh4fceZ# ./test4.sh
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00228137 s, 460 MB/s
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, 0.000997875 s, 2.1 MB/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, 0.00262184 s, 781 kB/s
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=908684k,nr_in$
des=227171,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,m$
de=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204908k,$
ode=755)
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=orde$
ed)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,node
v,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,siz
e=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=75
5)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noex
ec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent
```

```
root@iZ2ze4u33ixw0vp6rh4fceZ:#~ ./finalTest.sh
1+0 records in
1+0 records out
1048576 bytes(1.0 MB, 1.0 MiB) copied, 0.00157132 s, 667 MB/s
make2fs 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

1234567
JKLMNOP#root@iZ2ze4u33ixw0vp6rh4fceZ:~#
```

结果分析

从结果分析，myext2 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，对 1234567 进行加密，在用 ext2 文件格式读取后，结果为 JKLMNOP，实现了对数据的加密读写。

源程序

源程序见上文。

四、讨论、心得（20 分）

在这里写：实验过程中遇到的问题及解决的方法，您做本实验体会

在做实验的时候，由于第二次没有修改 changeMN.c 中的 fp_read=fopen("./myfs","rb");导致没有找到文件，所以出错了，在更改此处错误后，正确的执行并完成了工作。

在第五步的时候，按照老师给的代码进行读写的加密解密操作，发现并不能成功的对文件内容进行加密解密，在网上重新找了一份代码，第一次用户态的指针失效，发生了错误的内存访问，第二次经过修改后才得以实验成功。

myext2 是 ext2 的定制版本，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。

在做这个实验的时候，对 Linux 的虚拟文件系统查阅了相关资料。Linux 内核支持装载不同的文件系统类型，不同的文件系统有各自管理文件的方式，所以引入了 vfs，来进行简化。