

# 浙江大学

## 本科实验报告

课程名称:	操作系统
实验名称:	添加一个加密文件系统
姓 名:	彭子帆
学 院:	计算机科学与技术学院
系:	软件工程
专 业:	软件工程
学 号:	3170105860
指导教师:	彭子帆

2020 年 1 月 12 日

## 浙江大学实验报告

### 一、 实验目的

1. 文件系统是操作系统中最直观的部分，因为用户可以通过文件直接地和操作系统交互，操作系统也必须为用户提供数据计算、数据存储的功能。本实验通过添加一个文件系统，进一步理解 Linux 中的文件系统原理及其实现。
2. 深入理解操作系统文件系统原理，学习理解 Linux 的 VFS 文件系统管理技术，学习理解 Linux 的 ext2 文件系统实现技术，设计和实现 myext2 加密文件系统。

### 二、 实验内容

1. 添加一个类似 ext2 的文件系统 myext2。
2. 修改 myext2 的 magic number。
3. 添加文件系统创建工具。
4. 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。

### 三、 主要仪器设备

Linux 操作系统发行版本为 ubuntu 18.04.3 LTS，主机操作系统为 Windows 10，虚拟机软件为 VMware workstation 15.0 windows。内核版本为 5.0.0 的内核

### 四、 操作方法与实验步骤

#### 4.1 准备 myext 文件系统源文件

首先准备 myext2 文件系统需要的源文件，所需要进行的主要操作步骤是拷贝/fs/ext2 路径下的所有文件，都更改为 myext2 的文件名和变量名，同时还需要拷贝/lib/modules 路径下的三个跟 ext2 相关头文件，更改为 myext2 的文件名，之后再在需要包含新的头文件的路径之下添加对新的头文件的引用，并增加相应的宏定义。这一系列操作的截图以及注解如下方的截图所示，这里只列举出具有代表性的过程，全部截图详见附加文件：

```

sfofgalaxy@ubuntu:/usr/src/linux-headers-5.0.0-37-generic$ ls
arch      Documentation  include  Kconfig  mm        scripts  ubuntu
block     drivers        init     kernel   Module.symvers security  usr
certs     firmware      ipc      lib       net       sound    virt
crypto    fs             Kbuild   Makefile  samples   tools
sfofgalaxy@ubuntu:/usr/src/linux-headers-5.0.0-37-generic$ cd fs
sfofgalaxy@ubuntu:/usr/src/linux-headers-5.0.0-37-generic/fs$ ls
9p         configfs  f2fs      jffs2      notify     reiserfs
adfs       cramfs    fat        jfs         ntfs        romfs
affs       crypto    freevxfs  Kconfig     ocfs2       squashfs
afs        debugfs   fscache   Kconfig.binfmt omfs         sysfs
aufs       devpts    fuse      kernfs      openpromfs  sysv
autofs     dlm       gfs2      lockd       orangefs    tracefs
befs       ecryptfs  hfs       Makefile    overlayfs   ubifs
bfs        efivarfs  hfsplus   minix       proc         udf
btrfs     efs       hostfs    nfs         pstore       ufs
cachefiles exofs     hpfs      nfs_common  qnx4         xfs
ceph       exportfs  hugetlbfs nfsd         qnx6
cifs       ext2      isofs     nilfs2      quota
sfofgalaxy@ubuntu:/usr/src/linux-headers-5.0.0-37-generic/fs$ sudo cp -R ext2 myext2

sfofgalaxy@ubuntu:/usr/src$ cd linux-source-5.0.0/
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0$ cd ..
sfofgalaxy@ubuntu:/usr/src$ cd linux-source-5.0.0/
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0$ cd fs
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0/fs$ sudo cp -R ext2 myext2/
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0/fs$ cd myext2/
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2$ ls
acl.c      dir.c      ialloc.c  Kconfig   super.c   xattr.h           xattr_user.c
acl.h      ext2.h     inode.c   Makefile  symlink.c xattr_security.c
ballocc.c file.c     ioctl.c   namei.c   xattr.c   xattr_trusted.c
sfofgalaxy@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2$

sfofgalaxy@ubuntu:/lib/modules/5.0.0-37-generic/build/include/linux$ ls m
ls: cannot access 'm': No such file or directory
sfofgalaxy@ubuntu:/lib/modules/5.0.0-37-generic/build/include/linux$ cd /lib/modules/5.0.0-37-generic/build/in
clude/ init/
sfofgalaxy@ubuntu:/lib/modules/5.0.0-37-generic/build/include/linux$ cd /lib/modules/5.0.0-37-generic/build/include/asm-generic/bitops/
sfofgalaxy@ubuntu:/lib/modules/5.0.0-37-generic/build/include/asm-generic/bitops$ cp ext2-atomic.h myext2-atomic.h
cp: cannot create regular file 'myext2-atomic.h': Permission denied
sfofgalaxy@ubuntu:/lib/modules/5.0.0-37-generic/build/include/asm-generic/bitops$ sudo cp ext2-atomic.h myext2-atomic.h

```

图 1 准备源代码以及拷贝文件

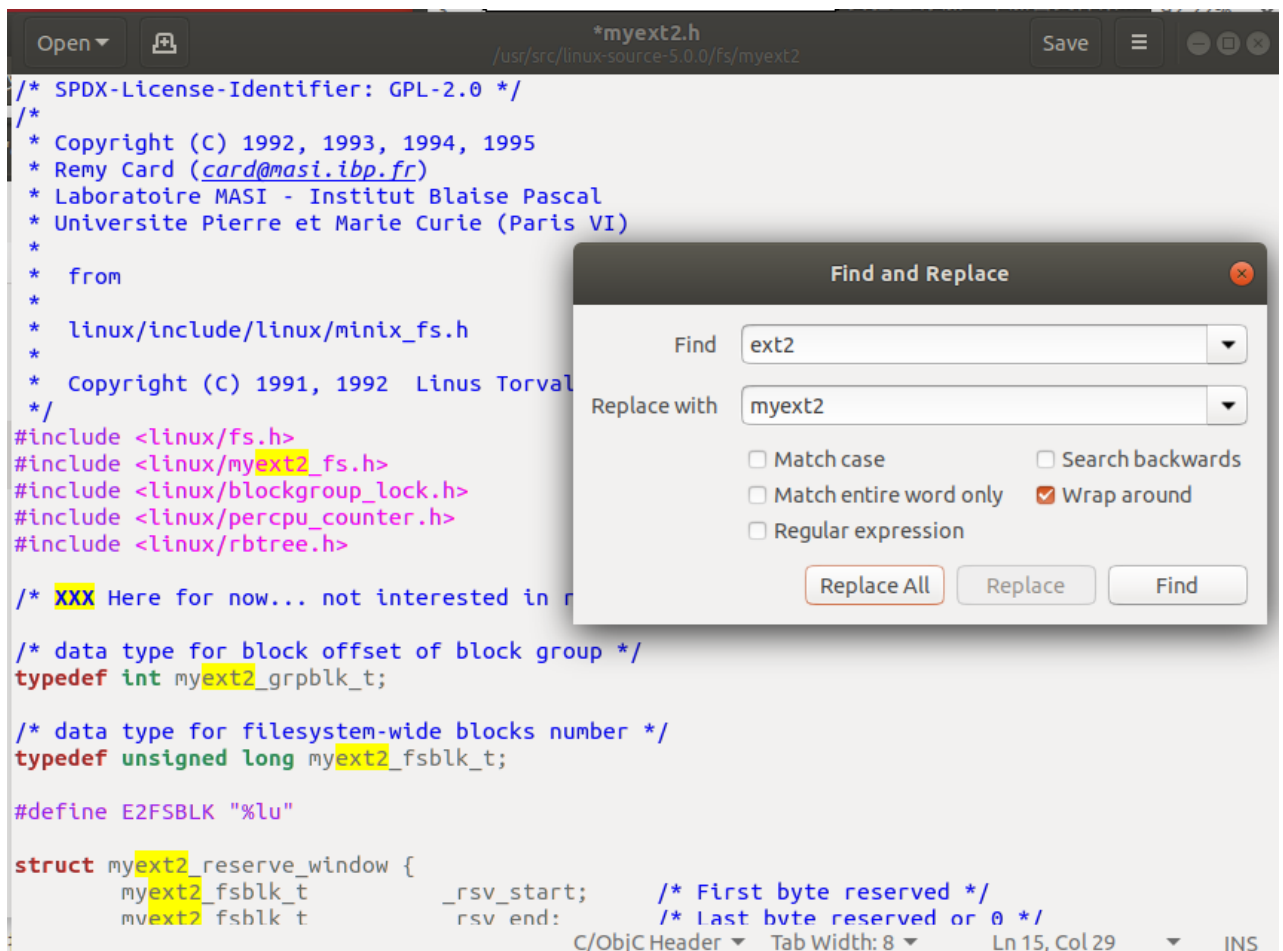


图 2 变更变量名

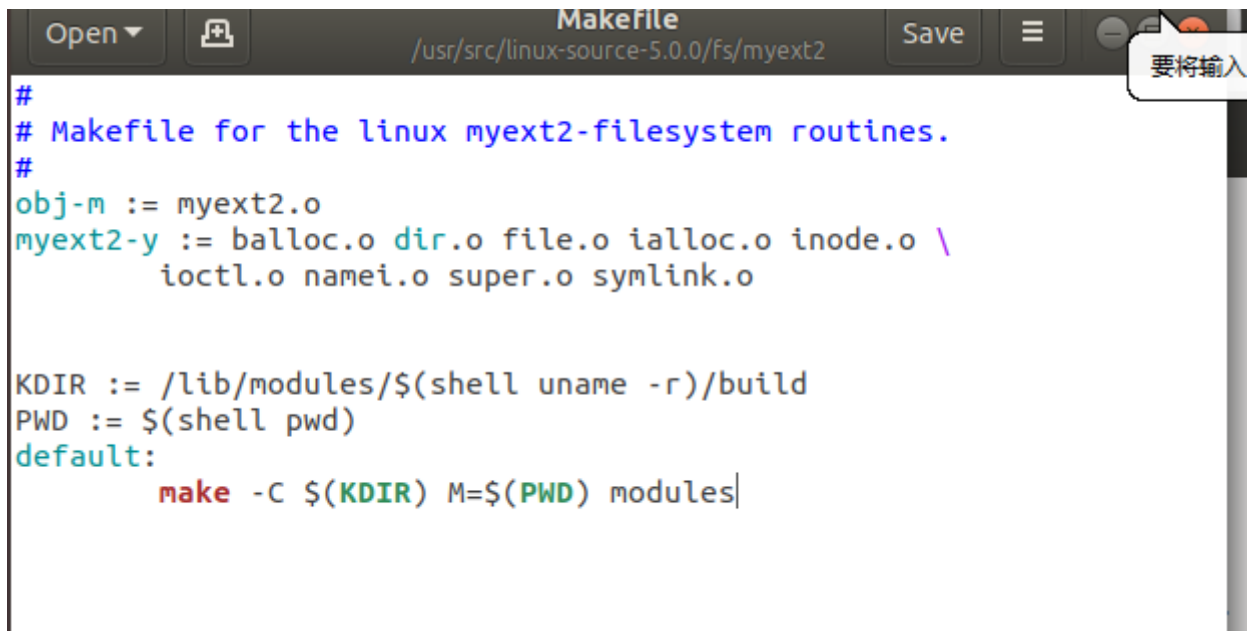


图 3 Makefile

```

#define EFS_SUPER_MAGIC          0x414A53
#define EXT2_SUPER_MAGIC         0xEF53
#define EXT3_SUPER_MAGIC         0xEF53
#define XENFS_SUPER_MAGIC        0xabba1974
#define EXT4_SUPER_MAGIC         0xEF53
#define BTRFS_SUPER_MAGIC        0x9123683E
#define NILFS_SUPER_MAGIC        0x3434
#define F2FS_SUPER_MAGIC         0xF2F52010
#define HPFS_SUPER_MAGIC         0xf995e849
#define ISOFS_SUPER_MAGIC        0x9660
#define JFFS2_SUPER_MAGIC        0x72b6
#define XFS_SUPER_MAGIC          0x58465342      /* "XFSB" */
#define PSTOREFS_MAGIC           0x6165676C
#define EFIVARFS_MAGIC           0xde5e81e4
#define HOSTFS_SUPER_MAGIC       0x00c0ffee
#define OVERLAYFS_SUPER_MAGIC    0x794c7630
#define MYEXT2_SUPER_MAGIC       0xEF53

#define MINIX_SUPER_MAGIC        0x137F          /* minix v1 f
14 char names */
#define MINIX_SUPER_MAGIC2      0x138F          /* minix v1 f
30 char names */
#define MINIX2_SUPER_MAGIC      0x2468          /* minix v2 f
14 char names */

```

图 4 添加 myext2 的 magic number

```

root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# bash substitute.sh
substitute ext2 to myext2 in acl.c...done
substitute EXT2 to MYEXT2 in acl.c...done
substitute ext2 to myext2 in acl.h...done
substitute EXT2 to MYEXT2 in acl.h...done
substitute ext2 to myext2 in balloc.c...done
substitute EXT2 to MYEXT2 in balloc.c...done
substitute ext2 to myext2 in dir.c...done
substitute EXT2 to MYEXT2 in dir.c...done
substitute ext2 to myext2 in file.c...done
substitute EXT2 to MYEXT2 in file.c...done
substitute ext2 to myext2 in ialloc.c...done
substitute EXT2 to MYEXT2 in ialloc.c...done
substitute ext2 to myext2 in inode.c...done
substitute EXT2 to MYEXT2 in inode.c...done
substitute ext2 to myext2 in ioctl.c...done
substitute EXT2 to MYEXT2 in ioctl.c...done
substitute ext2 to myext2 in Kconfig...done
substitute EXT2 to MYEXT2 in Kconfig...done
substitute ext2 to myext2 in Makefile...done

```

图 5 替换 fs/myext2 下的变量名

用 make 将 myext2 文件系统编译为内核模块进行挂在测试

```

root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# make
make -C /lib/modules/5.0.0-37-generic/build M=/usr/src/linux-source-5.0.0/fs/mye
xt2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.0.0-37-generic'
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/balloc.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/dir.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/file.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/ialloc.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/inode.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/ioctl.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/namei.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/super.o
  CC [M]  /usr/src/linux-source-5.0.0/fs/myext2/symlink.o
  LD [M]  /usr/src/linux-source-5.0.0/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
  CC      /usr/src/linux-source-5.0.0/fs/myext2/myext2.mod.o
  LD [M]  /usr/src/linux-source-5.0.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.0.0-37-generic'
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# insmod myext2.ko
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cat /proc/filesystems |grep m
yext2
    myext2
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2#

```

图 6 内核模块编译

```

sfofgalaxy@ubuntu: ~
File Edit View Search Terminal Help
sfofgalaxy@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, 0.00187511 s, 559 MB/s
sfofgalaxy@ubuntu:~$/sbin/mkfs.ext2 myfs
bash: /sbin/mkfs.ext2: No such file or directory
sfofgalaxy@ubuntu:~$ /sbin/mkfs.ext2 myfs
mke2fs 1.44.1 (24-Mar-2018)
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

sfofgalaxy@ubuntu:~$ mount -t myext2 -o loop ./myfs/mnt
mount: only root can use "--options" option
sfofgalaxy@ubuntu:~$ sudo mount -t myext2 -o loop ./myfs/mnt
[sudo] password for sfofgalaxy:
mount: ./myfs/mnt: can't find in /etc/fstab.
sfofgalaxy@ubuntu:~$ sudo mount -t myext2 -o loop ./myfs /mnt
sfofgalaxy@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=982548k,nr_inodes=245637,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=201336k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
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/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)

```



```

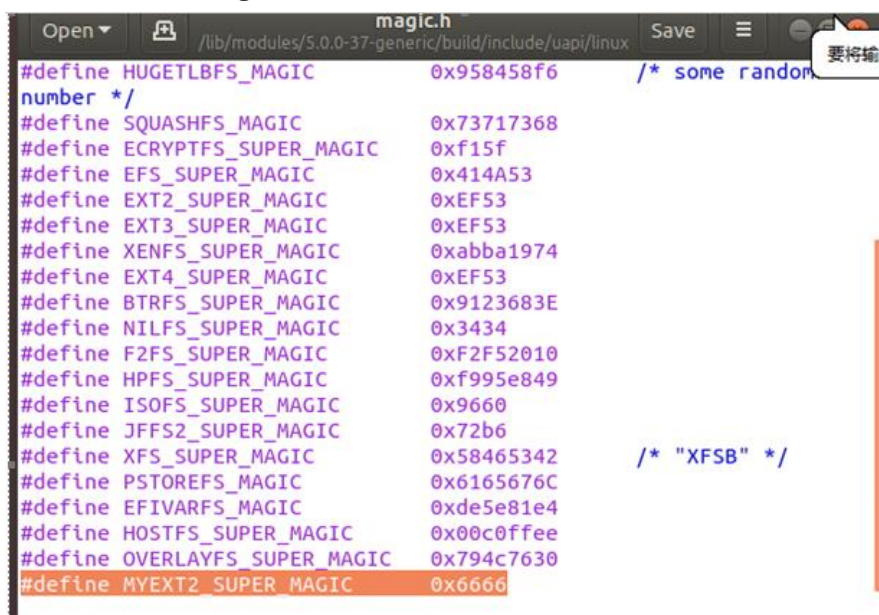
/home/sfofgalaxy/myfs on /mnt type myext2 (rw,relatime,errors=continue)
sfofgalaxy@ubuntu:~$ umount /mnt
umount: /mnt: umount failed: Operation not permitted.
sfofgalaxy@ubuntu:~$ sudo umount /mnt
sudo: umount: command not found
sfofgalaxy@ubuntu:~$ sudo umount /mnt
sfofgalaxy@ubuntu:~$ mount -t ext2 -o loop ./myfs /mnt
mount: only root can use "--options" option
sfofgalaxy@ubuntu:~$ sudo mount -t ext2 -o loop ./myfs /mnt
sfofgalaxy@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=982548k,nr_inodes=245637,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=201336k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
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/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/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/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/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)

gvfsd-fuse on /run/user/121/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=121,group_id=125)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201332k,mode=000,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)
/dev/sr0 on /media/sfofgalaxy/CDROM type iso9660 (ro,nosuid,nodev,relatime,nojoliet,check=s,map=n,blocksize=2048,uid=1000,gid=1000,dmode=500,fmode=400,uhelper=disk2)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=201332k,mode=700)
/home/sfofgalaxy/myfs on /mnt type ext2 (rw,relatime)
sfofgalaxy@ubuntu:~$ sudo umount /mnt
sfofgalaxy@ubuntu:~$ sudo rmdir myext2

```

图 7 挂载以及查看挂载结果

#### 4.2 修改 myext2 文件系统的 magic number, 进行挂载测试



```

Open ▾ /lib/modules/5.0.0-37-generic/build/include/uapi/linux Save
#define HUGETLBFS_MAGIC 0x958458f6 /* some random number */
#define SQUASHFS_MAGIC 0x73717368
#define ECRYPTFS_SUPER_MAGIC 0xf15f
#define EFS_SUPER_MAGIC 0x414a53
#define EXT2_SUPER_MAGIC 0xef53
#define EXT3_SUPER_MAGIC 0xef53
#define XENFS_SUPER_MAGIC 0xabba1974
#define EXT4_SUPER_MAGIC 0xef53
#define BTRFS_SUPER_MAGIC 0x9123683e
#define NILFS_SUPER_MAGIC 0x3434
#define F2FS_SUPER_MAGIC 0xf2f52010
#define HPFS_SUPER_MAGIC 0xf995e849
#define ISOFS_SUPER_MAGIC 0x9660
#define JFFS2_SUPER_MAGIC 0x72b6
#define XFS_SUPER_MAGIC 0x58465342 /* "XFSB" */
#define PSTOREFS_MAGIC 0x6165676c
#define EFIVARFS_MAGIC 0xde5e81e4
#define HOSTFS_SUPER_MAGIC 0x00c0ffee
#define OVERLAYFS_SUPER_MAGIC 0x794c7630
#define MYEXT2_SUPER_MAGIC 0x6666

```

图 1 修改 magic number

```

root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cat /proc/filesystems |grep m
yext2
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# make
make -C /lib/modules/5.0.0-37-generic/build M=/usr/src/linux-source-5.0.0/fs/mye
xt2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.0.0-37-generic'
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/balloc.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/dir.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/file.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/ialloc.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/inode.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/ioctl.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/namei.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/super.o
CC [M] /usr/src/linux-source-5.0.0/fs/myext2/symlink.o
LD [M] /usr/src/linux-source-5.0.0/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
LD [M] /usr/src/linux-source-5.0.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.0.0-37-generic'
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2#

```

图 2 重新编译结果

接下来我们通过 `dd` 命令建立文件系统的基本文件。由于我们这里只能使用 `mkfs.ext2` 创建工具，因此生成的文件系统是 `ext2`，想要将其与 `myext2` 关联，需要修改其 `magic number`。这里我们编辑编译 `changeMN.c` 文件，并将其作用在刚建立的文件之上：



```

Open changeMN.c Save
#include <stdio.h>
int main(void) {
    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;
    }
}

```



```
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mv /home/sfofgalaxy/Downloads
/changeMN .
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# dd if=/dev/zero of=myfs bs=1M
count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00324028 s, 324 MB/s
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# /sbin/mkfs.ext2 myfs
mke2fs 1.44.1 (24-Mar-2018)
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:/usr/src/linux-source-5.0.0/fs/myext2# ./changeMN
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount -t myext2 -o loop ./fs.
new /mnt
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount
```

图3 修改 MN 过程

接下来我们就可以进行挂载测试，可以发现只有 magic number 与 magic.h 中所定义好的 magic number 互相匹配的文件与文件系统才能匹配挂载，由于运行了 changeMN，文件系统只能与 myext2 关联而不能与 ext2 关联，结果如下：

```
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mv /home/sfofgalaxy/Downloads
/changeMN .
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# dd if=/dev/zero of=myfs bs=1M
count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00324028 s, 324 MB/s
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# /sbin/mkfs.ext2 myfs
mke2fs 1.44.1 (24-Mar-2018)
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:/usr/src/linux-source-5.0.0/fs/myext2# ./changeMN
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount -t myext2 -o loop ./fs.
new /mnt
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount
```

图4 挂载 myext2 成功

```

,dmode=500,fmode=400,uhelper=udisks2)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=20
1332k,mode=700)
/usr/src/linux-source-5.0.0/fs/myext2/fs.new on /mnt type myext2
(rw,relatime,errors=continue)
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# umount /mnt
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount -t ext2
-o loop ./fs.new /mnt
mount: /mnt: wrong fs type, bad option, bad superblock on /dev/lo
op16, missing codepage or helper program, or other error.
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# rmmod myext2
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2#

```

图 5 挂载 ext2 失败

#### 4.3 修改 myext2 文件系统的 mknod 操作，进行命令测试

```

Open  namei.c  Save
/usr/src/linux-source-5.0.0/fs/myext2

return PTR_ERR(inode);

myext2_set_file_ops(inode);
mark_inode_dirty(inode);
d_tmpfile(dentry, inode);
unlock_new_inode(inode);
return 0;
}

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

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

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

```

图 1 修改 mknod 函数的处理逻辑

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。之后我们使用 `mount` 命令挂载文件系统之后，在相应的路径之下试图进行创建管道操作。结果便是创建管道的操作不能成功，并且可以在内核输出信息中找到错误信息：

```
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# insmod myext2.ko
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount -t myext2 -o loop ./fs.new /mnt
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cd /mnt
root@ubuntu:/mnt# mknod myfifo p
mknod: myfifo: Operation not permitted
root@ubuntu:/mnt# dmesg|tail
[ 5377.319969] myext2: loading out-of-tree module taints kernel.
[ 5377.320113] myext2: module verification failed: signature and/or required key missing - tainting kernel
[ 6215.968767] EXT4-fs (loop16): mounting ext2 file system using the ext4 subsystem
[ 6215.977769] EXT4-fs (loop16): mounted filesystem without journal. Options: (null)
[ 6644.598572] e1000: ens33 NIC Link is Down
[ 6648.630588] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 6652.662897] e1000: ens33 NIC Link is Down
[ 6656.694943] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 7704.952332] EXT4-fs (loop16): VFS: Can't find ext4 filesystem
[ 9355.695011] haha, mknod is not supported by myext2! you've been cheated!
```

#### 4.4 添加 `myext2` 文件系统创建工具：`mkfs.myext2`

```
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# umount /dev/loop2
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# dd if=/dev/zero of=myfs bs=1M count=1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00535824 s, 196 MB/s
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# ./mkfs.myext2 myfs
losetup: /dev/loop2: detach failed: No such device or address
mke2fs 1.44.1 (24-Mar-2018)
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!
2048+0 records in
2048+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.0261698 s, 40.1 MB/s
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2#
```

```

root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=982548k,nr_inodes=245637,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=201336k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
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/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)

/usr/src/linux-source-5.0.0/fs/myext2/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
/usr/src/linux-source-5.0.0/fs/myext2/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2#

```

图 1 添加完成后进行测试并查看

#### 4.5 修改 myext2 文件系统的 read 和 write 操作，操作加密数据

注：采用的方法二进行加密

**准备工作** 我们首先重复前面的工作，挂载一个 myext2 的文件系统，同时在内新建一个测试文件 如下 由于此时还没有修改文件系统，自然数据是明文的，接下来就开始试着更改

```

root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cd /mnt
root@ubuntu:/mnt# echo 'AAA{Hello Encrypted File System}' > flag
root@ubuntu:/mnt# cat flag
AAA{Hello Encrypted File System}
root@ubuntu:/mnt#

```

图 1 未更改前

```

const struct file_operations myext2_file_operations = {
    .llseek      = generic_file_llseek,
    .read        = new_sync_read_crypt,
    .write       = new_sync_write_crypt,
    .read_iter   = myext2_file_read_iter,
    .write_iter  = myext2_file_write_iter,
    .unlocked_ioctl = myext2_ioctl,
#ifdef CONFIG_COMPAT
    .compat_ioctl = myext2_compat_ioctl,
#endif
    .mmap        = myext2_file_mmap,
    .open        = dquot_file_open,
    .release     = myext2_release_file,
    .fsync       = myext2_fsync,
    .get_unmapped_area = thp_get_unmapped_area,
    .splice_read = generic_file_splice_read,
    .splice_write = iter_file_splice_write,
};

```

图 2 添加文件操作



```

#define KEY    0x9E
static ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len, loff_t *ppos) {
    int i = 0;
    ssize_t ret = new_sync_read(filp, buf, len, ppos);
    for(i = 0; i < len; i++) {
        char* x = (char*)kmalloc(sizeof(char), GFP_ATOMIC);
        copy_from_user(x, buf + i, 1);
        x[0] ^= KEY;
        copy_to_user(buf + i, x, 1);
    }
    printk("[*] read: haha decrypt %ld \n", len);
    return ret;
}

static ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t len, loff_t *ppos) {
    int i = 0;
    for(i = 0; i < len; i++) {
        char* x = (char*)kmalloc(sizeof(char), GFP_ATOMIC);
        copy_from_user(x, buf + i, 1);
        x[0] ^= KEY;
        copy_to_user(buf + i, x, 1);
    }
    printk("[*] write: haha encrypt %ld \n", len);
    ssize_t ret = new_sync_write(filp, buf, len, ppos);
    return ret;
}

```

图 3 添加加密读写

```

1. #define KEY 0x9E
2. static ssize_t new_sync_read_crypt(struct file *filp, char __user *buf, size_t len, loff_t *ppos) {
3.     int i = 0;
4.     ssize_t ret = new_sync_read(filp, buf, len, ppos);
5.     for(i = 0; i < len; i++) {
6.         char* x = (char*)kmalloc(sizeof(char), GFP_ATOMIC);
7.         copy_from_user(x, buf + i, 1);
8.         x[0] ^= KEY;
9.         copy_to_user(buf + i, x, 1);
10.    }
11.    printk("[*] read: haha decrypt %ld \n", len);
12.    return ret;
13. }
14. static ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t len, loff_t *ppos) {
15.     int i = 0;
16.     for(i = 0; i < len; i++) {
17.         char* x = (char*)kmalloc(sizeof(char), GFP_ATOMIC);
18.         copy_from_user(x, buf + i, 1);
19.         x[0] ^= KEY;
20.         copy_to_user(buf + i, x, 1);
21.    }
22.    printk("[*] write: haha encrypt %ld \n", len);
23.    ssize_t ret = new_sync_write(filp, buf, len, ppos);
24.    return ret;
25. }

```



```

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

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

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

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

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

    ret = call_write_iter(filp, &kiocb, &iter);
    BUG_ON(ret == -EIOCBQUEUED);
}

```

图 4 添加位于 fs/read\_write.c 的静态读写函数

```
root@ubuntu:/mnt# echo 'AAA{Hello Encrypted File System}' > flag
```

```
sfofgalaxy@ubuntu:/mnt$ sudo cat flag
#####$#####'###3#####sfofgalaxy@ubuntu:/mnt$
```

图 5 验证流程之 flag - 1

```

[ 95.682863] myext2: loading out-of-tree module taints kernel.
[ 95.682910] myext2: module verification failed: signature and/or required key mis
ing - tainting kernel
[ 265.320905] [*] write: haha encrypt 33
[ 271.531038] [*] read: haha decrypt 131072
[ 271.538769] [*] read: haha decrypt 131072
[ 285.614190] [*] read: haha decrypt 131072
[ 285.621571] [*] read: haha decrypt 131072

```

图 6 输出信息

之后我们进行验证, 首先我们重新编译 myext2 文件系统之后, 将目录/mnt 挂载到 myext2 文件系统之上, 之后再里面新建文件并查看内容, 可以发现写入的数据和查看到的数据是完全相同的, 也就是说在这一路径之下的读写不会出现问题:

```

sfofgalaxy@ubuntu:/mnt$ echo '1234567' > test.txt
bash: test.txt: Permission denied
sfofgalaxy@ubuntu:/mnt$ sudo echo '1234567' > test.txt
bash: test.txt: Permission denied
sfofgalaxy@ubuntu:/mnt$ echo "echo '1234567' > test.txt" | sudo bash
sfofgalaxy@ubuntu:/mnt$ sudo cat test.txt
1234567
sfofgalaxy@ubuntu:/mnt$

```

图 7 验证流程之 test.txt - 2

接下来，我们使用 `cp` 命令将刚刚新建好的文件复制到另一路径之下，而且需要保证这一路径不是挂载在 `myext2` 文件系统之下的。复制成功之后再再将这一文件的内容输出，可以发现 还是可以正常读取其中的数据，这证明使用命令行环境中的复制操作，仍然需要调用对于加密数据的读写的函数，因此数据的传输是正常的。

然后， 我们来尝试使用图形界面的复制操作来复制这一加密文件到其他的文件系统。 由于 `ubuntu` 下的权限不够问题，我们可以使用 `sudo nautilus` 命令来以 `root` 身份打开图形文件处理界面，并将文件复制到另一路径之下再将其打开，结果可以发现阅读到的是加密文件，也就是说在图形界面之下的复制不会调用加密解密函数：

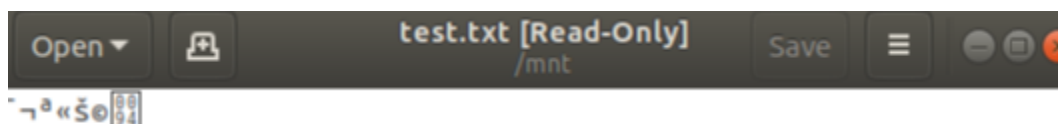


图 8 验证流程之 test.txt 图形窗口 - 3

最后一步操作，回调 `myext2` 文件系统的 `magic number` 为 `0xEF53`，并尝试在 `myext2` 文件 系统下创建加密文件，再将这一路径挂载到 `ext2` 文件系统之下，查看结果。结果发现，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。

```

$ /mnt/
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cd /mnt/
root@ubuntu:/mnt# echo "1234567" > test.txt
root@ubuntu:/mnt# cat test.txt
1234567
root@ubuntu:/mnt# cd ..
root@ubuntu:/# umount /mnt/
root@ubuntu:/# mount -t ext2 -o loop ./myfs /mnt/
mount: /mnt/: failed to setup loop device for ./myfs.
root@ubuntu:/# cd /usr/src/linux-source-5.0.0/fs/myext2/
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# mount -t ext2 -o loop ./
/mnt/
root@ubuntu:/usr/src/linux-source-5.0.0/fs/myext2# cd /mnt
root@ubuntu:/mnt# cat test.txt
*****root@ubuntu:/mnt#

```

图 9 验证流程之 test.txt 更改 magic number - 4


## 五、 讨论和心得

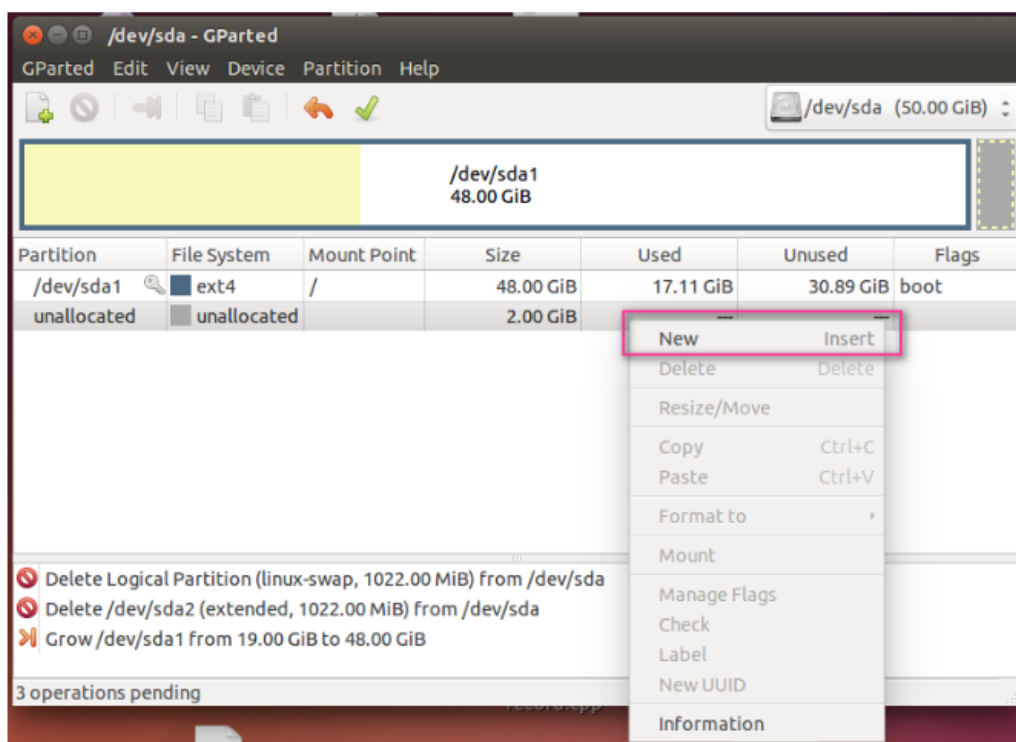
### 5.1 关于 虚拟机内存不够的问题

在下载源代码的时候我遇到了内存分配不够的问题，但是又不知道如何扩容，因此通过搜索以及尝试发现需要将虚拟机关闭后，打开 vmware tool 工具进行扩展，否则将无法扩展，并且在其中还要下载安装 Gparted

通过 `sudo apt-get install Gparted`



7.接下来重新调整/dev/sda1 的大小，点击菜单栏上的 ，重新给/dev/sda1划分大小。我这里调整为48GB，然后剩下2GB作为linux\_swap，重新将未分配的2GB格式化。



最后解决了我遇到的一个问题，参考的链接

[https://www.cnblogs.com/yongdaimi/p/9050155.html?tdsourcetag=s\\_pctim\\_aiomsg](https://www.cnblogs.com/yongdaimi/p/9050155.html?tdsourcetag=s_pctim_aiomsg)

## 5.2 关于 mkfs.myext2 脚本的编写问题

实验指导书上给出了一个 `mkfs.myext2` 的样例写法, 经过分析这个脚本的编写应当是没有问题的, 其实质是将 `/dev/loop0` 作为初始文件路径, 将它的信息关联到新的文件系统之上的方法。但是在我进行实际测试的时候, 发现这个路径已经挂载了一个称为 `sqflursh` 的文件系统, 不能再挂载其他的文件系统, 而且经过测试发现也无法通过 `umount` 命令来讲 `/dev/loop0` 路径解除挂载。因此我选择了另一种较为朴素的建立方法, 直接借助 `mkfs.ext2` 初始化文件系统文件之后, 将其 `magic number` 进行更改, 更改之后再将其重命名的一种建立方式, 这样写的话, 脚本的长度会大大缩短, 且后面的所有操作也都不会出现问题。

## 5.3 关于加密解密函数的编写问题

如果按照实验指导书上写的加密解密函数直接进行操作的话, 从理论上没有问题, 在 Linux 内核版本较为陈旧的环境中应当是可以正常运行的, 但是在我进行实验的时候, 遇到了进程会被 `killed` 的问题, 然后通过不断地缩小范围, 最终将发生错误的代码块锁定在了移位文本数据的操作之上。经过查询, 我了解到了直接对这样的内核态数据进行读写可能是会存在问题的, 这违反了 Linux 内核安全的某些原则。经过查找解决方案, 发现可以借助这样的一组函数: `copy_from_user()` 和 `copy_to_user()` 函数来进行数据模态的转换。观察 `new_sync_read()` 和 `new_sync_write()` 的函数声明, 可以发现在 `buf` 的生命的前面有一个 `__user` 的宏, 这个宏所修饰的数据是不能进行直接操作的。我们首先通过 `copy_from_user()` 函数将 `buf` 的数据读取到另一个变量值中, 这时的接收的变量不再被 `__user` 宏所修饰, 就可以进行直接的操作了。当对数据进行完移位操作之后, 再使用 `copy_to_user()` 函数将数据转存回去就可以了, 这时的操作不会再引起系统的内核堆栈错误。

```
1.      char* x = (char*)kmalloc(sizeof(char), GFP_ATOMIC);
2.      copy_from_user(x, buf + i, 1);
3.      x[0] ^= KEY;
4.      copy_to_user(buf + i, x, 1);
```

## 5.4 心得

本次实验作为最后一个实验, 实则是在是在软件工程专业的考试全部结束之后截至的一个实验, 因此我可以在考试结束后, 回家认真地琢磨该实验的流程, 钻研文件系统的结构等等。实验中, 我其实学会了很多, 复习了很多比如说编译启用内核模块的内容, 也有着很多新的知识点。通过这次实验, 我了解到了有关文件系统的创建, 删除, 编写与挂载, 理解了 Linux 文件系统通过 `mount` 命

令来进行挂载的自由灵活的机制。

除此之外，本来对于操作系统权限问题完全不知道如何处理，了解到了可以 `chmod 777 * -R` 即可将一个文件下所有文件设置为可读可写可执行，`sudo su` 可以直接进入 `root` 模式。并且还找到了 `src` 在磁盘中的位置，如何通过文件管理器打开，当然也可以通过命令进行打开图形界面。

经过了这个学期的学习之后，我对于操作系统和有了深一步的理解和认识，对于 `ubuntu` 操作系统也有了更多的了解，对于 `Linux` 命令也有了更多的了解，对于 `Linux` 内核中的很多关键技术和思想有了更多了解，我认为这些提升，不仅仅是对当前学习有帮助，对于以后的学习和工作也是非常有帮助的