浙江大学报告

一、 实验目的

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。

- 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
- 设计和实现加密文件系统

二、实验内容

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

三、实验环境

1. 电脑操作系统：Windows 10 Home
2. 虚拟机：VMWare Workstattion 12
3. 内核版本：3.18.24

四、实验步骤和结果

1. 添加一个类似 ext2 的文件系统 myext2

1) 源代码复制

把 fs 文件夹下的 ext2 文件夹复制一次， 并且把里面 ext2.h 的文件名改为 myext2.h

```
andy@andy:~/linux-3.18.24/fs$ cp -R ext2 myext2
andy@andy:~/linux-3.18.24/fs$ cd myext2
andy@andy:~/linux-3.18.24/fs/myext2$ mv ext2.h myext2.h
```

最后把/lib/modules/$(uname -r)/build /include/linux 中的 ext2_fs.h 文件、/lib/modules/$(uname -r)/build /include/asm-generic/bitops 中的 ext2-atomic.h 文件和 ext2-atomic-setbit.h 文件各复制一遍，并把文件名中的 ext2 改为 myext2。

```
root@andy:/lib/modules/3.18.24/build/include/linux# cp ext2_fs.h myext2_fs.h
```

```
root@andy:/lib/modules/3.18.24/build/include/asm-generic/bitops# cp ext2-atomic.
h myext2-atomic.h
root@andy:/lib/modules/3.18.24/build/include/asm-generic/bitops# cp ext2-atomic-
setbit.h myext2-atomic-setbit.h
```

2) 修改上面添加的文件的内容

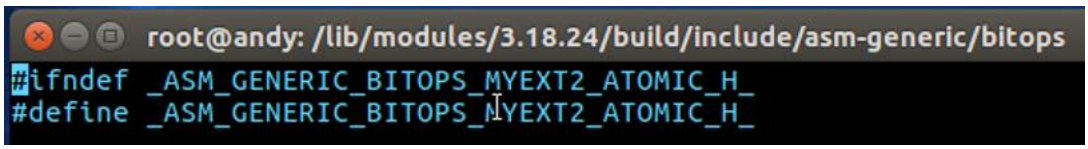
首先把上面复制的 myext2 文件夹中每个文件里的 ext2 和 EXT2 分别替换成 myext2 和 MYEXT2。为了简单起见，使用下面的脚本进行替换。

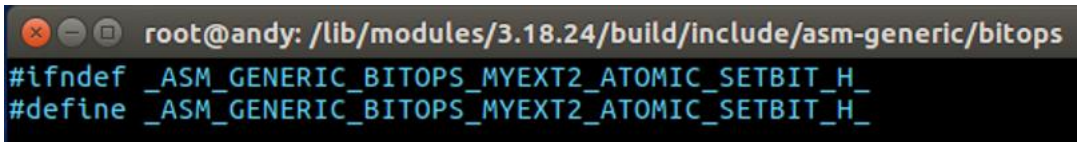
```
#!/bin/bash
SCRIPT=substitute.sh

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
```

```
andy@andy:~/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
```

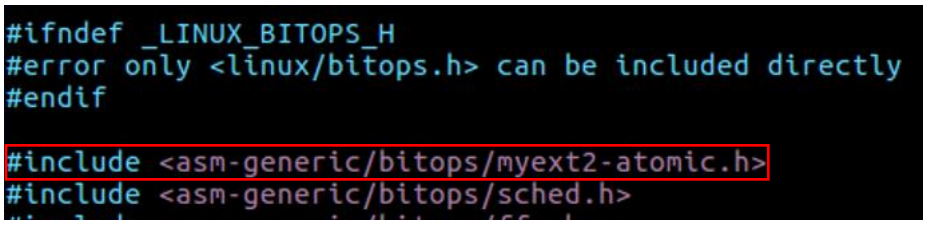
然后把第一步中另外三个复制的文件里的 ext2 和 EXT2 分别替换成 myext2 和 MYEXT2。

```
root@andy: /lib/modules/3.18.24/build/include/linux

#ifndef _LINUX_MYEXT2_FS_H
#define _LINUX_MYEXT2_FS_H
```

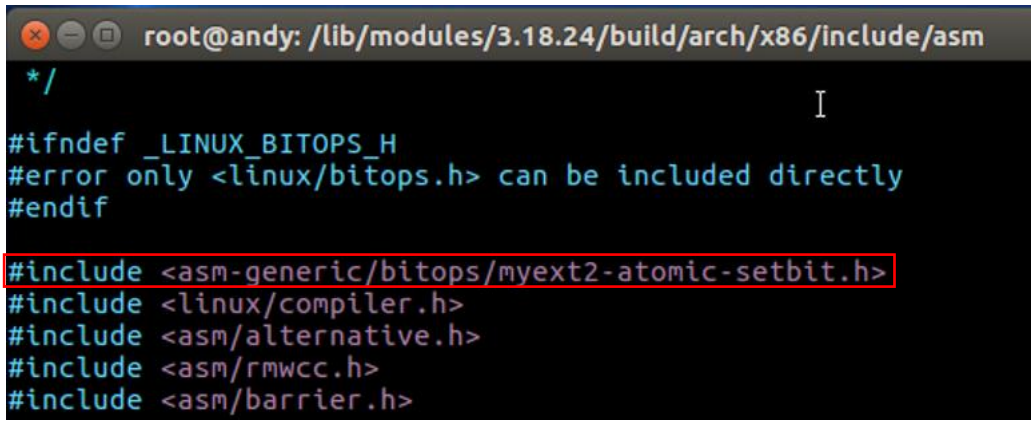




最后在/lib/modules/$(uname -r)/build/include/asm-generic/bitops.h 文件中添加#include <asm-generic/bitops/myext2-atomic.h>。



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>。



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

```
root@andy: /lib/modules/3.18.24/build/include/uapi/linux
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

#define MYEXT2_SUPER_MAGIC      0xef53
#define ADFS_SUPER_MAGIC        0xadf5
```

3) 加载内核模块

修改 myext2 文件夹中 Makefile 文件，修改内容如下。

```
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
```

使用 make 命令编译内核模块。

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# make
make -C /lib/modules/3.18.24/build M=/home/andy/linux-3.18.24/fs/myext2 modules
make[1]: Entering directory '/home/andy/linux-3.18.24'
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/balloc.o
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/dir.o
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/file.o
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/ialloc.o
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/inode.o
```

加载内核模块并查看是否加载成功。

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# insmod myext2.ko
root@andy:/home/andy/linux-3.18.24/fs/myext2# cat/proc/filesystem | grep myext2
root@andy:/home/andy/linux-3.18.24/fs/myext2# cat /proc/filesystem | grep myext2
cat: /proc/filesystem: No such file or directory
root@andy:/home/andy/linux-3.18.24/fs/myext2# cat /proc/filesystems | grep myext2
        myext2
```

4) 测试

创建大小为 1M，文件名为 myfs，内容全为 0 的文件，并将其格式化成 ext2 文件系统。

```
root@andy:/home/andy# 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, 0.000986697 s, 1.1 GB/s
root@andy:/home/andy# /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
```

用 myext2 文件系统格式将 myfs 通过 loop 设备 mount 到/mnt 目录下并检查当前系统的 mount 情况。下列结果显示 myext2 已经被内核所认可。

```
root@andy:/home/andy# mount -t myext2 -o loop ./myfs /mnt
root@andy:/home/andy# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
/home/andy/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@andy:/home/andy# umount /mnt
```

用 ext2 文件系统格式将 myfs 通过 loop 设备 mount 到/mnt 目录下并检查当前系统的 mount 情况。下列结果显示 ext2 已被内核认可，即 ext2 和 myext2 完全一致。

```
root@andy:/home/andy# mount -t ext2 -o loop ./myfs /mnt
root@andy:/home/andy# mount
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,use
r_id=1000,group_id=1000)
/home/andy/myfs on /mnt type ext2 (rw,relatime)
root@andy:/home/andy# umount /mnt
root@andy:/home/andy# rmmod myext2
```

2. 修改 myext2 的 magic number

1) 修改 myext2 的 magic number

```
root@andy: /lib/modules/3.18.24/build/include/uapi/linux
#ifndef __LINUX_MAGIC_H__
#define __LINUX_MAGIC_H__

#define MYEXT2_SUPER_MAGIC      0x6666
#define ADFS_SUPER_MAGIC        0xadf5
#define AFFS_SUPER_MAGIC        0xadff
```

2) 加载内核模块

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# make
make -C /lib/modules/3.18.24/build M=/home/andy/linux-3.18.24/fs/myext2 modules
make[1]: Entering directory '/home/andy/linux-3.18.24'
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/balloc.o
```

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# insmod myext2.ko
```

3) 编译以下程序，用于创建 myfs 文件系统的 magic number

```
#include <stdio.h>
main() {
    int ret; FILE *fp_read, *fp_write; unsigned char buf[2048];
    fp_read=fopen("./myfs","rb");
    if(fp_read == NULL) {
        printf("open myfs failed!\n");
        return 1;
    }
    fp_write=fopen("./fs.new","wb");
    if(fp_write==NULL) {
        printf("open fs.new failed!\n");
        return 2;
    }
    ret=fread(buf,sizeof(unsigned char),2048,fp_read);
    printf("previous magic number is 0x%x%x\n",buf[0x438],buf[0x439]);
    buf[0x438]=0x66; buf[0x439]=0x66;
    fwrite(buf,sizeof(unsigned char),2048,fp_write);
    printf("current magic number is 0x%x%x\n",buf[0x438],buf[0x439]);
    while(ret == 2048) {
        ret=fread(buf,sizeof(unsigned char),2048,fp_read);
        fwrite(buf,sizeof(unsigned char),ret,fp_write);
    }
    if(ret < 2048 && feof(fp_read)) printf("change magic number ok!\n");
    fclose(fp_read); fclose(fp_write);
    return 0;
}
```

```
root@andy:/home/andy# gcc -o changeMN changeMN.c
changeMN.c:2:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

4) 创建大小 1M，文件名为 myfs，内容全为 0 的文件

```
root@andy:/home/andy# 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, 0.00148664 s, 705 MB/s
root@andy:/home/andy# /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
```

5) 运行 changeMN 将 myfs 的 magic number 修改为 0x6666

```
root@andy:/home/andy# ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
```

6) 用 myext2 文件系统格式将 myfs 通过 loop 设备 mount 到/mnt 目录下并检查当前系统的 mount 情况。下列结果显示 myext2 已经被内核所认可。

```
root@andy:/home/andy# mount -t myext2 -o loop ./fs.new /mnt
root@andy:/home/andy# mount
/home/andy/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@andy:/home/andy# umount /mnt
```

7) 由于 magic number 不匹配，所以用-t ext2 去 mount myext2 文件系统失败

```
root@andy:/home/andy# sudo 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@andy:/home/andy# rmmod myext2
```

3. 修改文件系统操作

1) fs/myext2/namei.c 中的 myext2_mknod 函数作如下修改。

```
root@andy: /home/andy/linux-3.18.24/fs/myext2
static int myext2_mknod (struct inode * dir, struct dentry *dentry, umode_t mode, dev_t rde
v)
{

        printk(KERN_ERR"haha, mknod is not supported by myext2! you've been cheated!\n");
        return -EPERM;

/*      struct inode * inode;
```

2) 加载内核模块。

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# make
make -C /lib/modules/3.18.24/build M=/home/andy/linux-3.18.24/fs/myext2 modules
make[1]: Entering directory '/home/andy/linux-3.18.24'
  CC [M]  /home/andy/linux-3.18.24/fs/myext2/namei.o
  LD [M]  /home/andy/linux-3.18.24/fs/myext2/myext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/andy/linux-3.18.24/fs/myext2/myext2.mod.o
  LD [M]  /home/andy/linux-3.18.24/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/andy/linux-3.18.24'
root@andy:/home/andy/linux-3.18.24/fs/myext2# insmod myext2.ko
```

3) 将 myfs mount 到/mnt 目录下。

```
root@andy:/home/andy# mount -t myext2 -o loop ./fs.new /mnt
mount: /home/andy/fs.new is already mounted
root@andy:/home/andy# cd /mnt
root@andy:/mnt# mknod myfifo p
mknod: myfifo: Operation not permitted
root@andy:/mnt# dmesg | tail -1
[ 9556.124275] haha, mknod is not supported by myext2! you've been cheated!
```

4. 添加文件系统创建工具

1) 创建大小为 1M，文件名为 myfs，内容全为 0 的文件，并将其格式化成 ext2 文件系统。

```
root@andy:/home/andy# 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, 0.00361147 s, 290 MB/s
```

2) 修改 changgeMN.c 程序并重新编译

```
int ret; FILE *fp_read, *fp_write; unsigned char buf[2048];
fp_read=fopen("./tmpfs","rb");
if(fp_read == NULL) {
        printf("open myfs failed!\n");
        return 1;
}
```

```
root@andy:/home/andy# gcc -o changeMN changeMN.c
changeMN.c:2:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

3) 主目录下编辑 bash 程序 mkfs.myext2 并执行

```
#!/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
```

```
root@andy:/home/andy# sudo bash mkfs.myext2 myfs
```

4) 将 myfs mount 到/mnt 目录下。

```
root@andy:/home/andy# sudo mount -t myext2 -o loop ./myfs /mnt
root@andy:/home/andy# mount
/home/andy/myfs on /mnt type myext2 (rw,relatime,errors=continue)
root@andy:/home/andy# umount /mnt
```

5. 修改加密文件系统的 read 和 write 操作

1) 把 fs/read_write.c 中的 new_sync_write 和 new_sync_read 两个函数复制到 file.c 中，并添加头文件#include <linux/uio.h>，然后添加以下函数

```
    ssize_t new_sync_read_crypt(struct file *filp, const char __user *buf,
size_t len, loff_t *ppos)
    {
        int i;
        char *mybuf = buf;
        ssize_t ret = new_sync_read(filp, buf, len, ppos);
        char tmp;
        for(i=0; i<len; i++) {
            get_user(tmp, mybuf+i);
            tmp = tmp - 25;
            if(tmp < 0) tmp = tmp + 128;
            put_user(tmp, mybuf + i);
        }
        printk("haha encrypt %ld\n", len);
        return ret;
    };

    ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf,
size_t len, loff_t *ppos)
    {
        int i;
```

```
        char *mybuf = buf;
        char tmp;
        for(i=0; i<len; i++) {
            get_user(tmp, mybuf+i);
            tmp = tmp + 25;
            if(tmp >= 128) tmp = tmp - 128;
            put_user(tmp, mybuf + i);
        }
        printk("haha encrypt %ld\n", len);
        return new_sync_write(filp, mybuf, len, ppos);
};
```

2) 重新编译加载模块

```
root@andy:/home/andy/linux-3.18.24/fs/myext2# make
make -C /lib/modules/3.18.24/build M=/home/andy/linux-3.18.24/fs/myext2 modules
make[1]: Entering directory '/home/andy/linux-3.18.24'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/home/andy/linux-3.18.24'
```

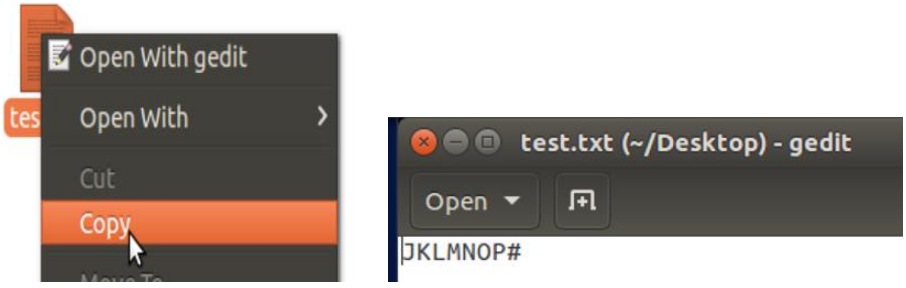
```
root@andy:/home/andy/linux-3.18.24/fs/myext2# insmod myext2.ko
```

3) 创建文件并测试结果

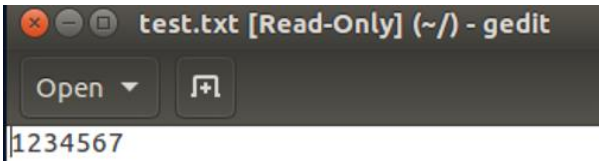
```
root@andy:/home/andy# mount -t myext2 -o loop ./fs.new /mnt
```

```
root@andy:/home/andy# cd /mnt
root@andy:/mnt# vi test.txt
root@andy:/mnt# cat test.txt
1234567
root@andy:/mnt# cp test.txt /home/andy
```

(a) 使用文件管理器复制查看结果



(b) 使用 cp 命令后查看结果



4) 把之前的 magic number 改回 0xEF53，然后重新编译和加载 myext2 模块，执行以下命令。

```
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
```

五、讨论、心得（20 分）

本次实验总共有 5 个部份组成，一开始我使用的内核是 4.15.1，而实验指导则是使用 3.18.24 内核，也因此我按照实验步骤做到要进行文件系统挂载以及要修改 magic number 时都出现了一些问题，像是 myext2 文件系统无法挂载，以及修改 magic number 在 4.15.1 的内核版本里没有定义一开始的 MYEXT2_SUPER_MAGIC，还有一些小问题等，都造成实验结果错误。最后我选择更换内核，先重新编译 3.18.24 的内核，比造实验指导中使用的内核版本，再按照时验步骤依序坐下来，之后就没有遇到什么问题了，最终完成了加密文件系统的制作，也对于使用命命令透过终端操作许多指令有了更深入的了解，而对于文件系统，我也学到了许多知识。