

# 苏州大学实验报告

院、系	计算机学院	年级专业	20 计科	姓名	柯骅	学号	2027405033
课程名称	操作系统课程实践					成绩	
指导教师	李培峰	同组实验者	无	实验日期	2023.05.18		

实验名称 实验9 文件系统设计

## 一、实验目的

1. 深入理解 Linux 文件系统的原理。
2. 学习并理解 Linux 的 VFS 文件系统管理技术
3. 学习并理解 Linux 的 ext2 文件系统实现技术
4. 设计并实现一个简单的类 ext2 文件系统

## 二、实验内容

设计并实现一个类似于 ext2 但能够对磁盘上的数据块进行加密的文件系统 myext2。本实验的主要内容如下。

- (1) 添加一个类似于 ext2 的文件系统 myext2。
- (2) 修改 myext2 文件系统的 magic number。
- (3) 修改文件系统操作。
- (4) 添加文件系统创建工具。

对于 myext2 文件系统，要求如下：

- (1) myext2 文件系统的物理格式定义与 ext2 文件系统基本一致，但 myext2 文件系统的 magic number 是 0x6666, 而 ext2 文件系统的 magic number 是 0xEF53。
- (2) myext2 文件系统是 ext2 文件系统的定制版本，前者不但支持 ext2 文件系统的部分操作，而且添加了文件系统创建工具。

## 三、实验步骤

本次实验使用的为 4.16.10 内核的 Linux 系统，且用户具有 root 权限

### 1. 添加一个类似于 ext2 的文件系统 myext2

为了添加一个类似于 ext2 的文件系统 myext2, 首先需要确定实现 ext2 文件系统的内核源代码由哪些文件组成。Linux 源代码结构很清楚地告诉我们: fs/ext2 目录下的所有文件都属于 ext2 文件系统。检查一下这些文件中所包含的头文件, 可以初步总结出 Linux 源代码中属于 ext2 文件的文件:

```
# 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
# ...
```

#### (1) 复制源代码

在分析完 ext2 文件系统由哪些文件组成之后, 下面开始进行实际的复制操作。第一个步骤是通过复制源代码, 添加 myext2 文件系统的源代码到 Linux 源代码。具体操作为把 ext2 文件系统的源代码复制到 myext2 文件系统中, 即复制一份以上所列的 ext2 源代码文件给 myext2 文件系统用。

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,头文件则存放到 include/linux 下。在 Linux Shell 下,可以执行如下操作:

```
# cd /usr/src/linux-4.16.10 /*内核源代码被解压到主目录的 linux-4.16.10 子目录下*/
# cd fs
# cp -R ext2 myext2
# cd /usr/src/linux/fs/myext2
# mv ext2.h myext2.h
# cd /lib/modules/$(uname -r)/build/ include/linux
# cp ext2_fs.h myext2_fs.h
# cd /lib/modules/$(uname -r)/build/include/asm-generic/bitops
# cp ext2-atomic.h myext2-atomic.h
# cp ext2-atomic-setbit.h myext2-atomic-setbit.h
```

这样就完成了复制文件系统工作的第一步复制源代码。对于复制文件系统来说,这当然还远远不够,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 文件系统改掉,所以现在该文件连编译都通不过。

## (2) 修改文件内容

为了使复制的源代码可以正确编译,下面开始进行复制文件系统工作的第二步—修改上面添加的文件的内容。为了简单起见,我们做了如下替换:将原来的 EXT2 替换成 MYEXT2,并将原来的 ext2 替换成 myext2。

对于 fs/myext2 目录下文件中字符串的替换,可以使用附件中 substitute.sh 脚本进行替换,代码如下:

```
1. #!/bin/bash
2. SCRIPT=substitute.sh
3. for f in *
4. do
5. if [ $f = $SCRIPT ]
6. then
7.     echo "skip $f"
8.     continue
9. fi
10. echo -n "substitute ext2 to myext2 in $f..."
11. cat $f | sed 's/ext2/myext2/g' > ${f}_tmp
12. mv ${f}_tmp $f
13. echo "done"
14. echo -n "substitute EXT2 to MYEXT2 in $f..."
15. cat $f | sed 's/EXT2/MYEXT2/g' > ${f}_tmp
16. mv ${f}_tmp $f
17. echo "done"
18. done
```

把上面这个脚本命名为 substitute.sh,放在 fs/myext2 目录下,加上可执行权限,运行之后即可把当前目录下所有文件里面的 ext2 和 EXT2 都替换成对应的 myext2 和 MYEXT2。

注意:

➤不要复制 Word 文档中的 substitute.sh 脚本,在 Linux 环境下应将其重新输入一遍,且 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。

➤在替换或修改内核代码时，可以使用 gedit 编辑器，但要注意大小写。

使用编辑器的替换功能把/lib/modules/\$(uname -r)/build/include/linux/myext2\_fs.h 以及 /lib/modules/\$(uname -r)/build/include/asm-generic/bitops/下的 myext2-atomic.h 和 myext2-atomic-secbith 文件中的 ext2、EXT2 分别替换成 myext2、MYEXT2, 同时进行如下修改。

- 在/lib/modules/\$(uname -r)/build/include/asm-gencric/bitops.h 文件中添加：

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

- 在/lib/modules/\$(uname -r)/build/arch/x86/include/asm/bitops.h 文件中添加：

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

- 在/lib/modules/\$(uname -r)/build/include/uapi/linux/magic.h 文件中添加：

- 1. #define MYEXT2\_SUPER\_MAGIC 0xEF53

修改源代码的工作到此结束

### (3) 把 myext2 编译成内核模块

接下来要做的第三步工作是把 myexl2 编译成内核模块。为了编译内核模块，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：

```
1. # Makefile for the linux myext2-filesystem routines.
2. obj-m := myext2.o
3. myext2-y := balloc.o dir.o file.o ialloc.o inode.o \
4.          ioctl.o namei.o super.o symlink.o
5. KDIR := /lib/modules/$(shell uname -r)/build
6. PWD := $(shell pwd)
7. default:
8.          make -C $(KDIR) M=$(PWD) modules
```

详见附件/test9/Makefile

编译内核模块，在 myext2 目录下执行命令：

```
# make
```

编译好模块后，使用 insmod 命令加载文件系统：

```
# insmod myext2.ko
```

最后查看一下 myext2 是否加载成功：

```
# cat /proc/filesystems | grep myext2
```

### (4) 对 myext2 文件进行测试

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

然后对添加的 myext2 文件系统进行测试，具体命令如下：

```
# dd if=/dev/zero of=myfs bs=1M count=1
# /sbin/mkfs.ext2 myfs
# mount -t myext2 -o loop ./myfs /mnt
# mount //该命令会列出已加载文件系统的信息
# umount /mnt
# mount -t ext2 -o loop ./myfs /mnt
# mount
# umount /mnt
# rmmod myext2 //卸载模块
```

## 2. 修改 myext2 文件系统的 magic number

在步骤 1 的基础上，找到 myext2 的 magic number，并将其改为 0x6666：

4. 16.10 内核版本中该值在 include/uapi/linux/magic.h 文件中，代码变更如下：

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

修改完之后，使用 make 命令重新编译内核模块，然后使用 insmod 命令安装编译好的 myext2.ko 内核模块。

在测试之前，需要编写程序 changeMN.c 来修改创建的 myfs 文件系统的 magic number，使其与内核中记录的 myext2 文件系统的 magic number 相匹配，只有这样 myfs 文件系统才能被正确加载。changeMN.c 中的代码详见附件 changeMN.c。

对 changeMN.c 进行编译后，将产生名为 changeMN 的可执行程序，命令如下：

```
# gcc -o changeMN changeMN.c
```

对修改 magic number 后的 myext2 文件系统进行测试，命令如下：

```
# 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
# sudo umount /mnt //卸载文件挂载
# sudo mount -t ext2 -o loop ./fs.new /mnt //尝试使用 ext2
# rmmod myext2
```

## 3. 修改文件系统操作

myext2 只是一个实验性质的文件系统，我们希望它只要能够支持简单的文件操作即可。因此在搭建完成 myext2 文件系统的总体框架以后，下面来修改 myext2 文件系统所支持的一些操作，以加深对文件系统操作的理解。以裁剪 myext2 文件系统的 mknod 操作为例，具体的实现流程说明如下。

Linux 将对块设备、字符设备和命名管道的操作，都看成对文件的操作。mknod 操作用来产生块设备、字符设备和命名管道所对应的节点文件。在 ext2 文件系统中，mknod 操作的实现函数和 ext2\_mknod() 函数详见附件/test9/mknod.c，其中 ext2\_mknod() 函数定义在 ext2\_dir\_inode\_operations 中。

当然，从 ext2 文件系统复制过去的 myext2 文件系统 myext2\_mknod() 和 myext2\_dir\_inode\_operations() 与上面的程序是一样的。对于 myext2\_mknod() 函数，在 myext2 文件系统中进行如下修改：

```
1. fs/myext2/namei.c
2. static int myext2_mknod (struct inode *dir, struct dentry *dentry, int mode, int rdev){
3.     printk(KERN_ERR "haha,mknod is not supported by myext2!you've been cheated!\n");
4.     return -EPERM;
5.     //将其他代码注释掉
6.     /* ...//其他代码 */
7. }
```

在上述修改后的程序中，第一行打印信息，指明 mknod 操作不被支持；第二行将错误号为 EPERM 的结果返回给 Shell，目的是告诉 Shell，在 myext2 文件系统中 mknod 操作不被支持；最后，myext2\_mknod() 函数原来的代码被注释掉了。因此，完成修改后，原来 myext2\_mknod() 函数的功能被删除了，取而代之的是输出不支持 mknod 操作的提示信息。

修改完成后，先使用 make 命令重新编译内核模块，在使用 insmod 命令安装编译好的 myext2.ko 内核模块。在 shell 中执行如下测试命令：

```
# mount -t myext2 -o loop ./fs.new /mnt //将 fs.new mount 到/mnt 目录下
# cd /mnt //进入/mnt 目录
# mknod myfifo p //创建一个名为 myfifo 的命名管道
mknod: 'myfifo': Operation not permitted
```

其中，最后一行为执行结果，也正是我们删除了 myext2\_mknod() 函数中的操作而将错误号 EPERM 退回给 Shell 的结果。需要注意的是，如果是在图形用户界面下使用虚拟控制台，那么 printk 打印出来的信息不一定能在终端显示出来，但可以通过命令 dmesg|tail 进行查看。

#### 4. 添加文件系统创建工具

文件系统的创建对于文件系统来说是首等重要的，因为如果连文件系统都不存在的话，那么所有的文件系统操作都是空操作，即无用操作。

其实，前面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。接下来所做工作的主要目的就是将这些内容总结一下，并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)。

首先需要确定的是程序的输入和输出。为了灵活和方便，这里的输入是一个文件，这个文件的大小就是 myext2 文件系统的大小；输出则是带了 myext2 文件的文件。

Shell 示例程序如下：

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

第 1 行：表明是 Shell 程序。

第 2 行：如果有程序用了 /dev/loop2，就将其卸载。

第 3 行：使用 losetup 将第一个参数代表的文件装到 /dev/loop2 上。

第 4 行：使用 mkfs.ext2 格式化 /dev/loop2，即使用 ext2 文件系统格式化新文件系统。

第 5 行：将文件系统的头 2KB 内容取出来，复制到临时文件 tmpfs 中。

第 6 行：调用程序 changeMN，读取 tmpfs 文件中的内容并将其复制到 fs.new 文件系统中，同时将 fs.new 文件系统的 magic\_number 改成 0x666

第 7 行：将被修改的 2KB 内容再写回去。

第 8 行：从 devloop2 上卸载文件系统。

第 9 行：将临时文件 tmpfs 删除。

编译完之后，进行如下测试：

```
# dd if=/dev/zero of=myfs bs=1M count=1
# chmod +x mkfs.myext2
# ./mkfs.myext2 myfs (或 sudo bash mkfs.myext2 myfs )
# sudo mount -t myext2 -o loop ./myfs /mnt
# mount
```

#### 四、实验结果

##### 1. 添加一个类似于 ext2 的文件系统 myext2

为了添加个类似于 ext2 的文件系统 myext2，首先需要确定实现 ext2 文件系统的内核源代码由哪些文件组成。Linux 源代码结构很清楚地告诉我们：fs/ext2 目录下的所有文件都属于 ext2 文件系统。检查一下这些文件中所包含的头文件，可以初步总结出 Linux 源代码中属于 ext2 文件系统的文件：

```
# 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
# ...
```

##### (1) 复制源代码

把 ext2 部分的源代码克隆到 myext2，按照 Linux 源代码的组织结构，把 myext2 文件系统的源代码存放到 fs/myext2 下，头文件放到 include/linux 下。执行如下操作：

```
# cd /usr/src/linux-4.16.10 /*内核源代码被解压到主目录的 linux-4.16.10 子目录下*/
# cd fs
# cp -R ext2 myext2
# cd /usr/src/linux/fs/myext2
# mv ext2.h myext2.h
# cd /lib/modules/$(uname -r)/build/include/linux
# cp ext2_fs.h myext2_fs.h
# cd /lib/modules/$(uname -r)/build/include/asm-generic/bitops
# cp ext2-atomic.h myext2-atomic.h
# cp ext2-atomic-setbit.h myext2-atomic-setbit.h
```

命令的执行没有输出结果。复制完之后，检查是否完成复制，结果如下：

```
root@ubuntu:/usr/src/linux/fs# cp -R ext2 myext2
root@ubuntu:/usr/src/linux/fs# cd /usr/src/linux/fs/myext2
root@ubuntu:/usr/src/linux/fs/myext2# mv ext2.h myext2.h
root@ubuntu:/usr/src/linux/fs/myext2# ls
acl.c      dir.c      inode.c   Makefile  super.c   xattr.h      xattr_user.c
acl.h      file.c     ioctl.c   myext2.h  symlink.c xattr_security.c
balloc.c   ialloc.c  Kconfig  namei.c   xattr.c   xattr_trusted.c

edma.h      mv643xx_i2c.h      sys.h
eeprom_93cx6.h      mvebu-pmu.h        syslog.h
eeprom_93xx46.h     mxm-wmi.h           sysrq.h
efi-bgrt.h          myext2_fs.h         sys_soc.h
efi.h               namei.h              sysv_fs.h
efs_vh.h            nd.h                 t10-pi.h

root@ubuntu:/lib/modules/4.16.10/build/include/linux# cd /lib/modules/$(uname -r)/build/include/asm-generic/bitops
root@ubuntu:/lib/modules/4.16.10/build/include/asm-generic/bitops# cp ext2-atomic-setbit.h myext2-atomic-setbit.h
root@ubuntu:/lib/modules/4.16.10/build/include/asm-generic/bitops# ls
arch_hweight.h  builtin-fls.h  ext2-atomic-setbit.h  find.h  hweight.h  myext2-atomic-setbit.h
atomic.h        builtin-fls.h  ffs.h                fls64.h  le.h        non-atomic.h
builtin_ffs.h   const_hweight.h  ffs.h                fls.h    lock.h      sched.h
builtin_ffs.h   ext2-atomic.h  ffz.h                fls.h    myext2-atomic.h
```

可以发现，复制成功。



## (2) 修改文件内容

将原来的 EXT2 替换为 MYEXT2，将原来的 ext2 替换为 myext2，并给 substitute.sh 脚本加上可执行权限后执行，命令如下：

```
# chmod +x substitute.sh
# bash substitute.sh
```

遇到 windows 转 linux 格式问题，使用 dos2unix 工具解决，结果如下左图：

```
root@ubuntu:/usr/src/linux/fs/myext2# dos2unix substitute.sh
dos2unix: 正在转换文件 substitute.sh 为Unix格式...
root@ubuntu:/usr/src/linux/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 toctl.c...done
substitute EXT2 to MYEXT2 in toctl.c...done
substitute ext2 to myext2 in kconfig...done
substitute EXT2 to MYEXT2 in kconfig...done
substitute ext2 to myext2 in Makefile...done
substitute EXT2 to MYEXT2 in Makefile...done
substitute ext2 to myext2 in myext2.h...done
substitute EXT2 to MYEXT2 in myext2.h...done
substitute ext2 to myext2 in namei.c...done
substitute EXT2 to MYEXT2 in namei.c...done
skip substitute.sh
substitute ext2 to myext2 in super.c...done
substitute EXT2 to MYEXT2 in super.c...done
substitute ext2 to myext2 in symlink.c...done
substitute EXT2 to MYEXT2 in symlink.c...done
substitute ext2 to myext2 in xattr.c...done
substitute EXT2 to MYEXT2 in xattr.c...done
substitute ext2 to myext2 in xattr.h...done
substitute EXT2 to MYEXT2 in xattr.h...done
substitute ext2 to myext2 in xattr_security.c...done
substitute EXT2 to MYEXT2 in xattr_security.c...done
substitute ext2 to myext2 in xattr_trusted.c...done
substitute EXT2 to MYEXT2 in xattr_trusted.c...done
substitute ext2 to myext2 in xattr_user.c...done
substitute EXT2 to MYEXT2 in xattr_user.c...done
```



使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”。也可以使用 vim 进行修改，如上右图修改 myext2\_fs.h 文件。

同时进行如下修改。

- 在/lib/modules/\$(uname -r)/build/include/asm-generic/bitops.h 文件中添加：

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

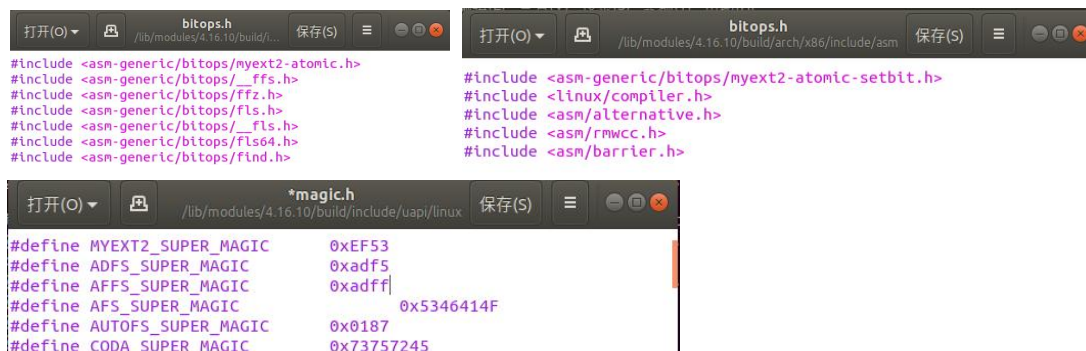
- 在/lib/modules/\$(uname -r)/build/arch/x86/include/asm/bitops.h 文件中添加：

```
#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
```

结果如下：



修改源代码的工作到此结束

### (3) 把 myext2 编译成内核模块

使用如下命令进行编译：

```
# make
```

运行结果如下：

```
root@ubuntu:/usr/src/linux/fs/myext2# make
make -C /lib/modules/4.16.10/build M=/usr/src/linux/fs/myext2 modules
make[1]: 进入目录"/usr/src/linux-4.16.10"
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev, libelf-devel or elfutils-libelf-devel"
CC [M] /usr/src/linux/fs/myext2/balloc.o
CC [M] /usr/src/linux/fs/myext2/dir.o
CC [M] /usr/src/linux/fs/myext2/file.o
CC [M] /usr/src/linux/fs/myext2/ialloc.o
CC [M] /usr/src/linux/fs/myext2/inode.o
CC [M] /usr/src/linux/fs/myext2/ioctl.o
CC [M] /usr/src/linux/fs/myext2/namei.o
CC [M] /usr/src/linux/fs/myext2/super.o
CC [M] /usr/src/linux/fs/myext2/symlink.o
LD [M] /usr/src/linux/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
CC /usr/src/linux/fs/myext2/myext2.mod.o
LD [M] /usr/src/linux/fs/myext2/myext2.ko
make[1]: 离开目录"/usr/src/linux-4.16.10"
```

编译好内核模块后，使用 insmod 命令加载文件系统，并查看 myext2 文件系统是否加载成功。

```
# insmod myext2.ko
```

```
# cat /proc/filesystems |grep myext2
```

如果加载成功，则/proc/filesystems 目录中将包含该文件系统，如下图所示：

```
root@ubuntu:/usr/src/linux/fs/myext2# insmod myext2.ko
root@ubuntu:/usr/src/linux/fs/myext2# cat /proc/filesystems |grep myext2
myext2
```

### (4) 对 myext2 文件进行测试

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

然后对添加的 myext2 文件系统进行测试，具体命令如下：

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

```
# /sbin/mkfs.ext2 myfs
```

其中，第 1 条命令使用/dev/zero 文件创建了 myfs 文件，第 2 条命令使用 ext2 文件系统对 myfs 文件进行了格式化。结果如下图所示：

```
root@ubuntu:/usr/src/linux/fs/myext2# dd if=/dev/zero of=myfs bs=1M count=1
记录了1+0 的读入
记录了1+0 的写出
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00106207 s, 987 MB/s
root@ubuntu:/usr/src/linux/fs/myext2# /sbin/mkfs.ext2 myfs
mke2fs 1.44.1 (24-Mar-2018)
丢弃设备块： 完成
创建含有 1024 个块（每块 1k）和 128 个inode的文件系统

正在分配组表： 完成
正在写入inode表： 完成
写入超级块和文件系统账户统计信息： 已完成
```

使用 myext2 文件系统将文件 myfs 挂载到设备/mnt 上并查看。执行以下两条命令：

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

```
# mount //该命令会列出已加载文件系统的信息
```

刚加载的信息会在最后显示出来，myfs 文件已经以 myext2 格式被挂载到了设备/mnt 上，运行结果如下图所示：

```
tmpfs on /run/user/121 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700,uid=121,gid=125)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,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)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700)
/usr/src/linux/fs/myext2/myfs on /mnt type myext2 (rw,relatime,errors=continue)
```

将文件 myfs 从/mnt 设备上卸载，并使用 ext2 文件系统重新挂载，操作也可以成功，因为 ext2 文件系统和我们创建的 myext2 文件系统是相同的。同样可以通过 mount 命令进行查看，代码如下：

```
# umount /mnt
```

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

```
# mount
```



运行结果如下：

```
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,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)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700)
/usr/src/linux/fs/myext2/myfs on /mnt type ext2 (rw,relatime,block_validity,barrier,user_xattr,acl)
```

可以发现，myfs 文件的格式已变为 ext2。

卸载 myfs 文件，同时卸载刚才挂载的模块，命令如下：

```
# umount /mnt
# rmmod myext2 //卸载模块
```

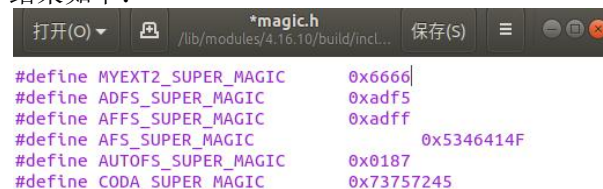
## 2. 修改 myext2 文件系统的 magic number

在步骤 1 的基础上，找到 myext2 的 magic number，并将其改为 0x6666：

4. 16. 10 内核版本中该值在 include/uapi/linux/magic.h 文件中，代码变更如下：

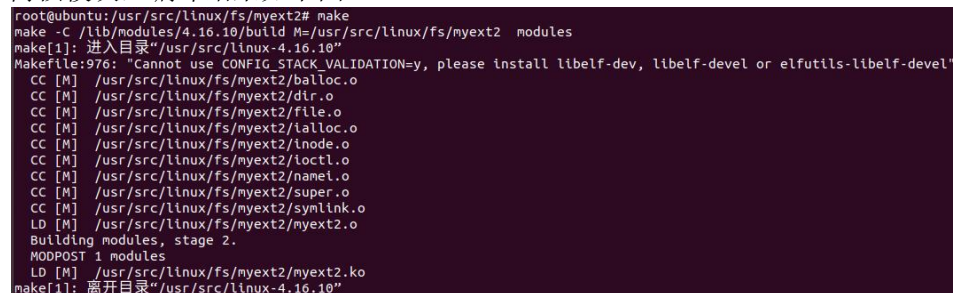
```
- # define MYEXT2_SUPER_MAGIC    0xEF53
+ # define MYEXT2_SUPER_MAGIC    0x6666
```

结果如下：



```
*magic.h
#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
```

修改完之后，使用 make 命令重新编译内核模块，然后使用 insmod 命令安装编译好的 myext2.ko 内核模块，编译结果如下图：



```
root@ubuntu:/usr/src/linux/fs/myext2# make
make -C /lib/modules/4.16.10/build M=/usr/src/linux/fs/myext2 modules
make[1]: 进入目录 "/usr/src/linux-4.16.10"
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev, libelf-devel or elfutils-libelf-devel"
CC [M] /usr/src/linux/fs/myext2/balloc.o
CC [M] /usr/src/linux/fs/myext2/dir.o
CC [M] /usr/src/linux/fs/myext2/file.o
CC [M] /usr/src/linux/fs/myext2/lalloc.o
CC [M] /usr/src/linux/fs/myext2/lnode.o
CC [M] /usr/src/linux/fs/myext2/loctl.o
CC [M] /usr/src/linux/fs/myext2/namel.o
CC [M] /usr/src/linux/fs/myext2/super.o
CC [M] /usr/src/linux/fs/myext2/symlink.o
LD [M] /usr/src/linux/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
LD [M] /usr/src/linux/fs/myext2/myext2.ko
make[1]: 离开目录 "/usr/src/linux-4.16.10"
```

编写程序 changeMN.c，修改创建的 myfs 文件系统的 magic number，使其与内核中记录的 myext2 文件系统的 magic number 匹配，只有这样 myfs 文件系统才能被正确加载。编译完 changeMN.c 程序后，产生的可执行程序名为 changeMN，命令如下：

```
# gcc -o changeMN changeMN.c
```

输入如下测试命令，创建测试文件 myfs，然后使用 ext2 文件系统对其进行格式化。

```
# dd if=/dev/zero of=myfs bs=1M count=1
# /sbin/mkfs.ext2 myfs
# ./changeMN myfs
```

运行结果如下：



```
root@ubuntu:/home/kh/test/test9# dd if=/dev/zero of=myfs bs=1M count=1
记录了1+0 的读入
记录了1+0 的写出
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00139507 s, 752 MB/s
root@ubuntu:/home/kh/test/test9# /sbin/mkfs.ext2 myfs
mke2fs 1.44.1 (24-Mar-2018)
丢弃设备块: 完成
创建含有 1024 个块 (每块 1k) 和 128 个inode的文件系统

正在分配组表: 完成
正在写入inode表: 完成
写入超级块和文件系统账户统计信息: 已完成
```

执行可执行程序 changeMN: 新建了文件 fs.new, 同时 fs.new 文件的 magic number 被修改为 0x6666。结果如下图:

```
root@ubuntu:/home/kh/test/test9# ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
```

继续测试, 使用 myext2 文件系统对文件 fs.new 进行格式化并将其挂载到设备/mnt 上, 然后使用 mount 命令列出已挂载文件的信息并查看挂载结果, 执行如下命令:

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

结果如下:

```
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,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)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700)
/home/kh/test/test9/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
```

最后, 尝试一下 fs.new 文件是否可以使用 ext2 文件系统进行格式化并挂载, 命令如下:

```
# sudo umount /mnt //卸载文件挂载
# sudo mount -t ext2 -o loop ./fs.new /mnt //尝试使用 ext2
# rmmod myext2
```

结果得到错误提示, 如下图所示:

```
root@ubuntu:/home/kh/test/test9# sudo umount /mnt
root@ubuntu:/home/kh/test/test9# sudo mount -t ext2 -o loop ./fs.new /mnt
mount: /mnt: wrong fs type, bad option, bad superblock on /dev/loop29, missing codepage or helper program, or other error
```

原因是修改了 magic number 后的 fs.new 文件无法与 ext2 文件系统匹配。

### 3. 修改文件系统操作

在搭建完 myext2 文件系统的总体框架后, 修改 myext2 文件系统支持的一些操作, 以加深读者对文件系统操作的理解。我们以裁剪 myext2 文件系统的 mknod 操作为例, 说明一下具体的实现流程。修改所用的代码请参考实验步骤部分。

修改完毕后, 重新编译内核, 结果如下图:

```
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 = dqot_initialize(dir);
    if (err)
        return err;

    inode = ext2_new_inode(dir, mode, &dentry->d_name);
    err = PTR_ERR(inode);
    if (IS_ERR(inode)) {
        int special_inode(inode, inode->i_mode, rdev);
    }
    #ifdef CONFIG_EXT2_FS_XATTR
        inode->i_op = &ext2_special_inode_operations;
    #endif
    mark_inode_dirty(inode);
    err = ext2_add_nondir(dentry, inode);
    }
    return err;*/
}
```

```
root@ubuntu:/usr/src/linux/fs/myext2# make
make -C /lib/modules/4.16.10/build M=/usr/src/linux/fs/myext2 modules
make[1]: 进入目录"/usr/src/linux-4.16.10"
Makefile:976: "Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev, libelf-devel or elfutils-libelf-devel"
CC [M] /usr/src/linux/fs/myext2/namei.o
LD [M] /usr/src/linux/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
CC /usr/src/linux/fs/myext2/myext2.mod.o
LD [M] /usr/src/linux/fs/myext2/myext2.ko
make[1]: 离开目录"/usr/src/linux-4.16.10"
```

在 shell 中执行如下测试命令:

```
# mount -t myext2 -o loop ./fs.new /mnt //将 fs.new mount 到/mnt 目录下
# cd /mnt //进入/mnt 目录
# mknod myfifo p //创建一个名为 myfifo 的命名管道
mknod: 'myfifo': Operation not permitted
```

运行结果如下:

```
root@ubuntu:/home/kh/test/test9# cd /mnt
root@ubuntu:/mnt# mknod myfifo p
mknod: myfifo: 不允许的操作
```

结果与代码第四行预期结果一致, 成功。

此外程序还会将错误号 EPERM 返回给 Shell。需要注意的是，如果是在图形用户界面下使用虚拟控制台，那么 printk 打印出来的信息不一定能在终端显示出来，但可以通过命令 `dmesg|tail` 进行查看，如下图所示：

```
root@ubuntu:/mnt# dmesg | tail
[13853.005303] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[13853.006855] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[18953.479798] perf: interrupt took too long (39519 > 39510), lowering kernel.perf_event_max_sample_rate to 5000
[22232.762555] myext2: loading out-of-tree module taints kernel.
[22232.762749] myext2: module verification failed: signature and/or required key missing - tainting kernel
[22763.459953] EXT4-fs (loop29): mounting ext2 file system using the ext4 subsystem
[22763.467010] EXT4-fs (loop29): mounted filesystem without journal. Opts: (null)
[24617.230106] perf: interrupt took too long (49864 > 49398), lowering kernel.perf_event_max_sample_rate to 4000
[25080.458695] EXT4-fs (loop29): VFS: Can't find ext4 filesystem
[27309.626466] haha, mknod is not supported by myext2! you've been cheated!
```

同样也获得了预期输出。

#### 4. 添加文件系统创建工具

我们的目的是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）。Shell 脚本的编写请参考实验指导部分。编辑完之后，运行如下代码：

```
# dd if=/dev/zero of=myfs bs=1M count=1
# chmod +x mkfs.myext2
# ./mkfs.myext2 myfs (或 sudo bash mkfs.myext2 myfs )
# sudo mount -t myext2 -o loop ./myfs /mnt
```

结果如下：

```
root@ubuntu:/home/kh/test/test9# ./mkfs.myext2 myfs
losetup: myfs: 设置回环设备失败: 设备或资源忙
mke2fs 1.44.1 (24-Mar-2018)
/dev/loop2 有一个 squashfs 文件系统
Proceed anyway? (y,N) y
/dev/loop2 已经挂载; 取消建立 文件系统 !
记录了2+0 的读入
记录了2+0 的写出
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.000443389 s, 4.6 MB/s
previous magic number is 0x00
current magic number is 0x6666
change magic number ok!
dd: 正在写入 '/dev/loop2': 不允许的操作
记录了1+0 的读入
记录了0+0 的写出
0 bytes copied, 0.000307493 s, 0.0 kB/s
```

使用 `# mount` 查看挂载情况，结果如下：

```
tmpfs on /run/user/121 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700,uid=121,gid=125)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=401524k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/home/kh/test/test9/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
```

可以发现 fs.new 已被成功挂载，实验成功。

### 五、实验思考与总结

#### 1. 思考：windows 的回车符与 linux 的换行符

- windows 换行是 `\r\n`，十六进制数值是：0D0A。
- Linux 换行是 `\n`，十六进制数值是：0A
- 所以在 linux 保存的文件在 windows 上用记事本看的话会出现黑点，我们可以在 LINUX 下用命令把 linux 的文件格式转换成 win 格式的。
- unix2dos 是把 linux 文件格式转换成 windows 文件格式
- dos2unix 是把 windows 格式转换成 linux 文件格式。
- 例如：

```
1. # 安装 dos2unix
2. sudo apt-get install dos2unix
3. # 转换文件格式
4. dos2unix substitute.sh
```

## 2. 总结

本次的实验内容是设计一个简单的文件系统,在 linux4.16.10 内核源码的基础上复制并修改相关代码与配置文件实现了文件系统模块,并对相关功能进行了验证,最后添加了文件系统创建工具。在此过程中,深入理解了 Linux 文件系统的原理,学习并理解了 Linux 的 VFS 文件系统管理技术、ext2 文件系统实现技术,最后运用这些技术设计并实现了一个简单的类 ext2 文件系统,对 Linux 内核的理解进一步加深。