

浙江大学

本科实验报告

课程名称: 操作系统

姓 名: 贺婷婷

学 院: 计算机科学与技术学院

系: 计算机科学与技术系

专 业: 软件工程

学 号: 3170104341

指导教师: 夏莹杰

2019 年 12 月 20 日

浙江大学实验报告

课程名称： 操作系统 实验类型： 综合

实验项目名称： Linux 操作系统环境

学生姓名： 贺婷婷 专业： 软件工程 学号： 3170104341

电子邮件地址： 3170104341@zju.edu.cn 手机： 17326086580

实验地点： 曹光彪二期 503 实验日期： 2019 年 12 月 20 日

一、实验目的和要求

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读写。

三、主要仪器设备

计算机配置： 处理器 英特尔 Core i5-5300U @ 2.30GHz 双核

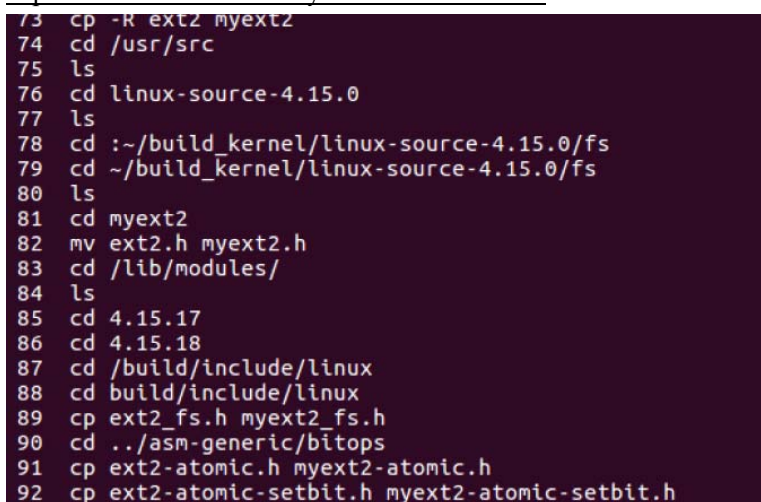
内存 7872MB (DDR3 1600MHz)
主硬盘 三星 MZNLN512HCJH-000L1 (512 GB / 固态硬盘)
显卡 英特尔 HD Graphics 5500 (128 MB / 联想)
操作系统环境: Windows 10 64 位 (DirectX 12)
Linux 版本: Ubuntu-18.10

四、操作方法和实验步骤

2.1 添加一个类似 ext2 的文件系统 myext2

1. 源代码复制: 克隆 ext2 文件系统文件, 将 fs/ext2/... 克隆至 fs/myext2/..., include/linux/ext2_fs.h 至 include/linux/myext2_fs.h

```
#cd /usr/src/linux /*内核源代码目录, 假设内核源代码解压在主目录的 linux 子目录*/  
#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
```



```
73 cp -R ext2 myext2  
74 cd /usr/src  
75 ls  
76 cd linux-source-4.15.0  
77 ls  
78 cd ~/build_kernel/linux-source-4.15.0/fs  
79 cd ~/build_kernel/linux-source-4.15.0/fs  
80 ls  
81 cd myext2  
82 mv ext2.h myext2.h  
83 cd /lib/modules/  
84 ls  
85 cd 4.15.17  
86 cd 4.15.18  
87 cd /build/include/linux  
88 cd build/include/linux  
89 cp ext2_fs.h myext2_fs.h  
90 cd ../asm-generic/bitops  
91 cp ext2-atomic.h myext2-atomic.h  
92 cp ext2-atomic-setbit.h myext2-atomic-setbit.h
```

2. 修改文件内数据结构、函数、宏的名字

将原来“EXT2”替换成“MYEXT2”; 将原来的“ext2”替换成“myext2”, 使用如下脚本文件进行替换, 将该脚本 (substitute.sh) 加上可执行权限。

```
#sudo bash substitute.sh
```

➤ Warning: 先删除 fs/myext2 目录下的 *.o 文件, 再运行脚本程序。

```
#!/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

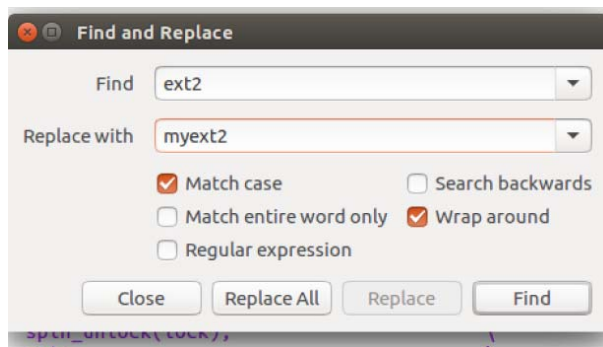
```

```

substitute EXT2 to MYEXT2 in ioctl.c...done
substitute ext2 to myext2 in Kconfig...done
substitute EXT2 to MYEXT2 in Kconfig...done
substitute ext2 to myext2 in Makefile...done
substitute EXT2 to MYEXT2 in Makefile...done
substitute ext2 to myext2 in myext2.h...done
substitute EXT2 to MYEXT2 in myext2.h...done
substitute ext2 to myext2 in namei.c...done
substitute EXT2 to MYEXT2 in namei.c...done
skip substitute.sh
substitute ext2 to myext2 in super.c...done
substitute EXT2 to MYEXT2 in super.c...done
substitute ext2 to myext2 in symlink.c...done
substitute EXT2 to MYEXT2 in symlink.c...done
substitute ext2 to myext2 in xattr.c...done
substitute EXT2 to MYEXT2 in xattr.c...done
substitute ext2 to myext2 in xattr.h...done
substitute EXT2 to MYEXT2 in xattr.h...done

```

利用编辑器替换把 /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”



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/myext2-atomic-setbit.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/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
```

```
#define MYEXT2_SUPER_MAGIC 0xEF53
```

3.把 myext2 编译成内核模块: 生成一个 makefile 文件后执行 make 命令。

```
#  
# 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  
  
#  
# 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  
|
```

```
erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ make  
make -C /lib/modules/4.15.18/build M=/home/erica/build_kernel/linux-source-4.15.0/fs/myext2 modules  
make[1]: Entering directory '/home/erica/build_kernel/linux-source-4.15.0'  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/balloc.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/dir.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/file.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/ialloc.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/inode.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/ioctl.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/namei.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/super.o  
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/symlink.o  
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o  
Building modules, stage 2.  
MODPOST 1 modules  
CC /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.mod.o  
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko  
make[1]: Leaving directory '/home/erica/build_kernel/linux-source-4.15.0'
```

4.测试文件系统: insmod 加载模块, 查看是否加载成功。

```
#insmod myext2.ko
```

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



```
erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ sudo insmod myext2.ko
[sudo] password for erica:
erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ cat /proc/filesystems
|grep myext2
myext2
```

成功后进行下一步测试，首先将当前目录设置为主目录，测试如下命令：

```
#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)
```

```
getlb)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=22,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=19640)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201796k,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/erica/myfs on /mnt type myext2 (rw,relatime,errors=continue)
```

```
#umount /mnt
```

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

```
#mount
```

```
.....
```

```
..... on /mnt type ext2 (rw)
```

```
getlb)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=22,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=19640)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201796k,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/erica/myfs on /mnt type ext2 (rw,relatime)
```

```
#umount /mnt
```

```
#rmmod myext2 /*卸载模块*/
```

```
erica@ubuntu:~$ sudo umount /mnt
erica@ubuntu:~$ rmmod myext2
rmmod: ERROR: ../libkmod/libkmod-module.c:793 kmod_module_remove_module() could
not remove 'myext2': Operation not permitted
rmmod: ERROR: could not remove module myext2: Operation not permitted
erica@ubuntu:~$ sudo rmmod myext2
```

2.2 修改 myext2 的 magic number

- 1.修改 magic number 为 0x6666: `include/uapi/linux/magic.h` 中:

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

```
//#define MYEXT2_SUPER_MAGIC 0xEF53
#define MYEXT2_SUPER_MAGIC    0x6666
```

- 2.重新编译内核模块:

```
#make
```

```
#insmod myext2.ko
```

- 3.使用 `changeMN.c` 修改 `mysf` 文件系统的 magic number:

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

```

4.运行 changMN 并进行测试:

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

#/sbin/mkfs.ext2 myfs

#!/changeMN myfs

```

erica@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, 0.00674675 s, 155 MB/s
erica@ubuntu:~$ /sbin/mkfs.ext2 myfs
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

erica@ubuntu:~$ ./changeMN myfs
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!

```

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

#mount

..... on /mnt type myext2 (rw)

```

getlb)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=22,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=19640)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
configs on /sys/kernel/config type configfs (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201796k,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/erica/fs.new on /mnt type myext2 (rw,relatime,errors=continue)
erica@ubuntu:~$

```

#sudo umount /mnt

sudo mount -t ext2 -o loop ./fs.new /mnt

mount: wrong fs type, bad option, bad superblock on /dev/loop0, ...


```
erica@ubuntu:~$ 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.
```

rmmod myext2

2.3 修改文件系统操作-裁剪 mknod 操作

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。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。fs/ext2/namei.c, line 141 中定义实现函数，在 fs/ext2/namei.c, line 428 中的 ext2_dir_inode_operations 结构中定义

1. 重定义 myext2_mknod 函数，打印错误信息，并返回错误号 EPERM

```
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 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;
    */
}
```

2. make 重新编译内核模块，insmod 安装 myext2.ko，测试如下：

```

erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ make
make -C /lib/modules/4.15.18/build M=/home/erica/build_kernel/linux-source-4.15.0/fs/myext2 modules
make[1]: Entering directory '/home/erica/build_kernel/linux-source-4.15.0'
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.mod.o
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/erica/build_kernel/linux-source-4.15.0'

```

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

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

【我的妈呀这边的-写成了-（中英文符号区别）然后一直不行，人傻了】

#mount fs.new 到 mnt 目录下

#cd /mnt #进入文件系统

#mknod myfifo p

mknod: `myfifo': Operation not permitted

#若无显示可食用 dmesg|tail 来显示结果（printk）

```

erica@ubuntu:/mnt$ sudo mknod myfifo p
mknod: myfifo: Operation not permitted
erica@ubuntu:/mnt$ dmesg|tail
[ 46.544222] Bluetooth: RFCOMM TTY layer initialized
[ 46.544235] Bluetooth: RFCOMM socket layer initialized
[ 46.544249] Bluetooth: RFCOMM ver 1.11
[ 4345.144880] myext2: loading out-of-tree module taints kernel.
[ 4345.145691] myext2: module verification failed: signature and/or required key missing - tainting kernel
[ 4623.475575] EXT4-fs (loop0): mounting ext2 file system using the ext4 subsystem
[ 4623.500727] EXT4-fs (loop0): mounted filesystem without journal. Opts: (null)
[ 5384.748581] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[ 5397.514078] EXT4-fs (loop0): VFS: Can't find ext4 filesystem
[ 6722.581894] haha, mknod is not supported by myext2! you've been cheated!

```

2.4 添加文件系统创建工具

1. 制作一个方便快捷的 myext2 文件系统的创建工具：mkfs.myext2，输入为一个文件（大小为 myext2 文件系统的大小），输出带了 myext2 文件系统的文件

~/mkfs.myext2

#!/bin/bash

/sbin/losetup -d /dev/loop2 #释放/dev/loop2

/sbin/losetup /dev/loop2 \$1 #losetup 将第一个参数代表的文件装到/dev/loop2

/sbin/mkfs.ext2 /dev/loop2 用 ext2 文件系统格式格式化我们的文件系统

dd if=/dev/loop2 of=./tmpfs bs=1k count=2 将文件系统的头 2K 字节的内容取出来

./changeMN \$1 ./tmpfs #调用程序 changeMN 读取 tmpfs, 复制到 fs.new, 并且将 fs.new 的 magic number 改成 0x6666

dd if=./fs.new of=/dev/loop2

/sbin/losetup -d /dev/loop2

rm -f ./tmpfs

```

Open  [?] 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

```

3. 修改 changeMN 程序适合本节需要。

/dev/loop2 的文件信息被发送给了 ./tmpfs 文件，所以 changeMN 应该去 ./tmpfs 文件中读取信息而去写 fs.new

```

#include <stdio.h>
main()
{
    int ret;
    FILE *fp_read;
    FILE *fp_write;
    unsigned char buf[2048];
    fp_read=fopen("./tmpfs","rb");
    if(fp_read == NULL){
        printf("open tmpfs failed!\n");
        return 1;
    }
    fp_write=fopen("./fs.new","wb");
    if(fp_write==NULL){
        printf("open fs.new failed!\n");
        return 2;
    }
}

```

- 3.测试:

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

```
# ./mkfs.myext2 myfs (或 sudo bash mkfs.myext2 myfs)
```

```

eric@ubuntu:~$ sudo bash mkfs.myext2 myfs
losetup: /dev/loop2: detach failed: No such device or address
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

2+0 records in
2+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.00774407 s, 264 kB/s
previous magic number is 0x53ef
current magic number is 0x6666
change magic number ok!
4+0 records in
4+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.00237645 s, 862 kB/s

```

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

```
# mount
```

```
/dev/loop on /mnt myext2 (rw)
```

```

cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hu
getlb)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,fr
eezer)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blki
o)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime
,perf_event)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=34,pgrp=1,time
out=0,minproto=5,maxproto=5,direct,pipe_ino=19951)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_i
d=0,group_id=0,default_permissions,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=201796k,mode=7
00,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime
,user_id=1000,group_id=1000)
/home/erica/myfs on /mnt type myext2 (rw,relatime,errors=continue)
erica@ubuntu:~$

```

2.5 修改加密文件系统的 read 和 write 操作

1. 修改 file.c，添加两个函数 new_sync_read_crypt 和 new_sync_write_crypt，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。

因为 buf 是 _user* 类型的变量，表示用户空间内容，内核空间不能修改，需要用到 copy_to_user 和 copy_from_user 来进行值的传递与修改。

```

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）
    char* crypt=(char *)kmalloc(sizeof(char)*len, GFP_KERNEL);
    int i;
    copy_from_user(crypt,buf,len);
    for(i=0;i<len;i++){
        crypt[i]=( crypt[i]+25)%128;
    }
    copy_to_user(buf,crypt,len);
    printk("haha encrypt %ld\n", len);
    return new_sync_write(filp, buf, len, ppos); //调用默认的写函数，把加密数据写入
}

```



```

ssize_t new_sync_write_crypt(struct file *filp, const char __user *buf, size_t
len, loff_t *ppos)
{
    //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）
    char* crypt=(char *)kmalloc(sizeof(char)*len, GFP_KERNEL);
    int i;
    copy_from_user(crypt,buf,len);
    for(i=0;i<len;i++){
        crypt[i]=(crypt[i]+25)%128;
    }
    copy_to_user(buf,crypt,len);
    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
}

```

```

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）
    char* decrypt=(char *)kmalloc(sizeof(char)*len, GFP_KERNEL);
    int i;
    copy_from_user(decrypt,buf,len);
    for(i=0;i<len;i++){
        decrypt[i]=(decrypt[i]-25 +128 )%128;
    }
    copy_to_user(buf,decrypt,len);

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

```

```

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）
    char* decrypt=(char *)kmalloc(sizeof(char)*len, GFP_KERNEL);
    copy_from_user(decrypt,buf,len);
    for(i=0;i<len;i++){
        decrypt[i]=(decrypt[i]-25 +128 )%128;
    }
    copy_to_user(buf,decrypt,len);
    printk("haha encrypt %ld\n", len);
    return ret;
}

```

2. 将这两个函数指针赋给 myext2_file_operations 结构中的 read 和 write 操作。


```

const struct file_operations myext2_file_operations = {
    .llseek      = generic_file_llseek,
    .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,
    .read        = new_sync_read_crypt,
    .write       = new_sync_write_crypt,
};

```

4. 重新编译 myext2 模块，insmod 安装，创建 myext2 文件系统并尝试写入字符串文件。

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

```
cd /mnt/
```

```

erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ sudo insmod myext2.ko
erica@ubuntu:~/build_kernel/linux-source-4.15.0/fs/myext2$ cd ~
erica@ubuntu:~$ sudo mount -t myext2 -o loop ./fs.new /mnt/
mount: wrong fs type, bad option, bad superblock on /dev/loop0,
       missing codepage or helper program, or other error

       In some cases useful info is found in syslog - try
       dmesg | tail or so.
erica@ubuntu:~$ sudo mount -t myext2 -o loop ./fs.new /mnt
mount: wrong fs type, bad option, bad superblock on /dev/loop0,
       missing codepage or helper program, or other error

       In some cases useful info is found in syslog - try
       dmesg | tail or so.
erica@ubuntu:~$ sudo mount -t myext2 -o loop ./myfs /mnt
erica@ubuntu:~$ cd /mnt/
erica@ubuntu:/mnt$

```

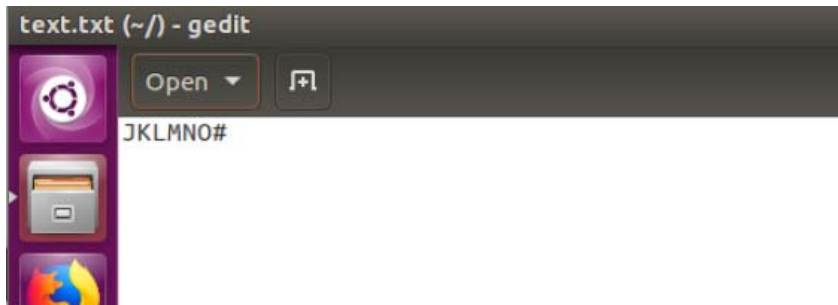
5. 新建文件 text.txt 写入字符串“123456” 查看内容。
6. 把 test.txt 复制到主目录下，查看内容

```

erica@ubuntu:/mnt$ cat text.txt
123456
erica@ubuntu:/mnt$ mv text.txt ~/
mv: cannot remove 'text.txt': Permission denied
erica@ubuntu:/mnt$ sudo mv text.txt ~/
erica@ubuntu:/mnt$ cd ~
erica@ubuntu:~$ cat text.txt
123456
erica@ubuntu:~$ dmesg |tail -5
[ 2814.631844] haha encrypt 131072
[ 2823.070521] haha encrypt 131072
[ 2823.071385] haha encrypt 131072
[ 2826.997318] haha encrypt 131072
[ 2826.998002] haha encrypt 131072
erica@ubuntu:~$

```

7. 使用文件管理器的复制，再查看结果？
打开后发现是加密后的内容



8. 把之前的 magic number 改回 0xEF53。重新编译 myext2 模块安装 myext2.ko, 执行以下指令使得即使使用 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。

前往 `/lib/modules/4.15.18/build/include/uapi/linux/magic.h` 修改:

```
#define MYEXT2_SUPER_MAGIC 0xEF53
```

重新编译:

```
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/ialloc.o
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/inode.o
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/loctl.o
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/namei.o
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/super.o
CC [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/symlink.o
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.o
Building modules, stage 2.
MODPOST 1 modules
LD [M] /home/erica/build_kernel/linux-source-4.15.0/fs/myext2/myext2.ko
make[1]: Leaving directory '/home/erica/build_kernel/linux-source-4.15.0'
```

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

`/sbin/mkfs.ext2 myfs`

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

```
erica@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, 0.00693174 s, 151 MB/s
erica@ubuntu:~$ /sbin/mkfs.ext2 myfs
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

erica@ubuntu:~$ mount -t myext2 -o loop ./myfs /mnt
mount: only root can use "--options" option
erica@ubuntu:~$ sudo mount -t myext2 -o loop ./myfs /mnt
```

`cd /mnt`

`echo "1234567" > test.txt`

`cat test.txt` 显示原始内容

`cd`

`umount /mnt`

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

`cd /mnt`

`cat test.txt` 显示加密后内容

```

erica@ubuntu:/mnt$ cat test.txt
1234567
erica@ubuntu:/mnt$ cd ~
erica@ubuntu:~$ sudo umount /mnt
erica@ubuntu:~$ sudo mount -t ext2 -o loop ./myfs /mnt
erica@ubuntu:~$ cd /mnt/
erica@ubuntu:/mnt$ cat test.txt
JKLMNOP#erica@ubuntu:/mnt$

```

五、讨论和心得

总结：通过这次实验，对于 Linux 文件系统的运行有了一个比较粗浅的认识，作为课堂外的实践来说非常不错，就是做之前发现我不知道为啥删掉了上次实验的虚拟机，新装的虚拟机里又没有 ext2，只好苦涩地抱着我的憨憨电脑又重建了六个小时内核呜呜呜呜

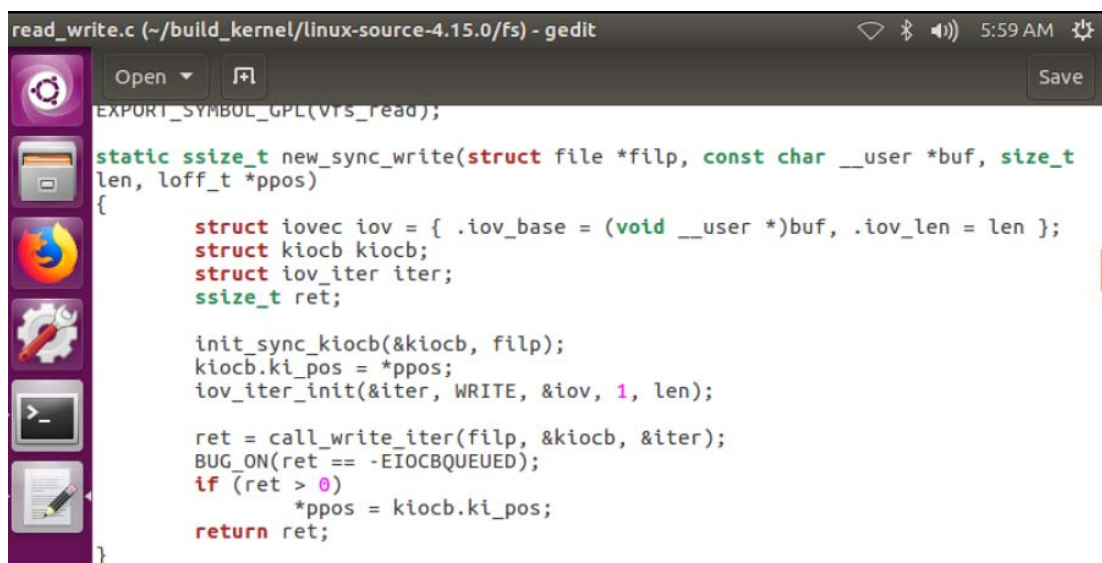
1. Linux 在操作时均需要 root 权限才能操作，比如挂载和卸载文件系统，保证系统安全性。
2. 做的途中有个地方的英文-写成了中文-，导致摸不着头脑地失败了。告诫我们不要复制粘贴...
3. 在加密过程中体会到了内核空间 and 用户空间的分离性，内核空间无法修改 char_user* 类型的变量，只能通过 copy_to_user 和 copy_from_user 来进行值的传递与修改。
4. 【file.c 中不存在 new_sync_write 和 new_sync_read 函数，导致报错】

```

copy_to_user(void __user *to, const void *from, unsigned long n)
^
/home/erica/build_kernel/linux-source-4.15.0/fs/myext2/file.c:193:9: error: impl
icit declaration of function 'new_sync_write' [-Werror=implicit-function-declara
tion]
    return new_sync_write(filp, buf, len, ppos); //调用默认的写函数，把♦♦
    ^
/home/erica/build_kernel/linux-source-4.15.0/fs/myext2/file.c: In function 'new_
sync_read_crypt':
/home/erica/build_kernel/linux-source-4.15.0/fs/myext2/file.c:200:23: error: imp
licit declaration of function 'new_sync_read' [-Werror=implicit-function-declara
tion]
    ssize_t ret = new_sync_read(filp, buf, len, ppos);

```

解决方法:找到了定义这个的文件（fs/read_write.c），把这两个函数复制了过去。



```

read_write.c (~/.build_kernel/linux-source-4.15.0/fs) - gedit
EXPORT_SYMBOL_GPL(vfs_read);

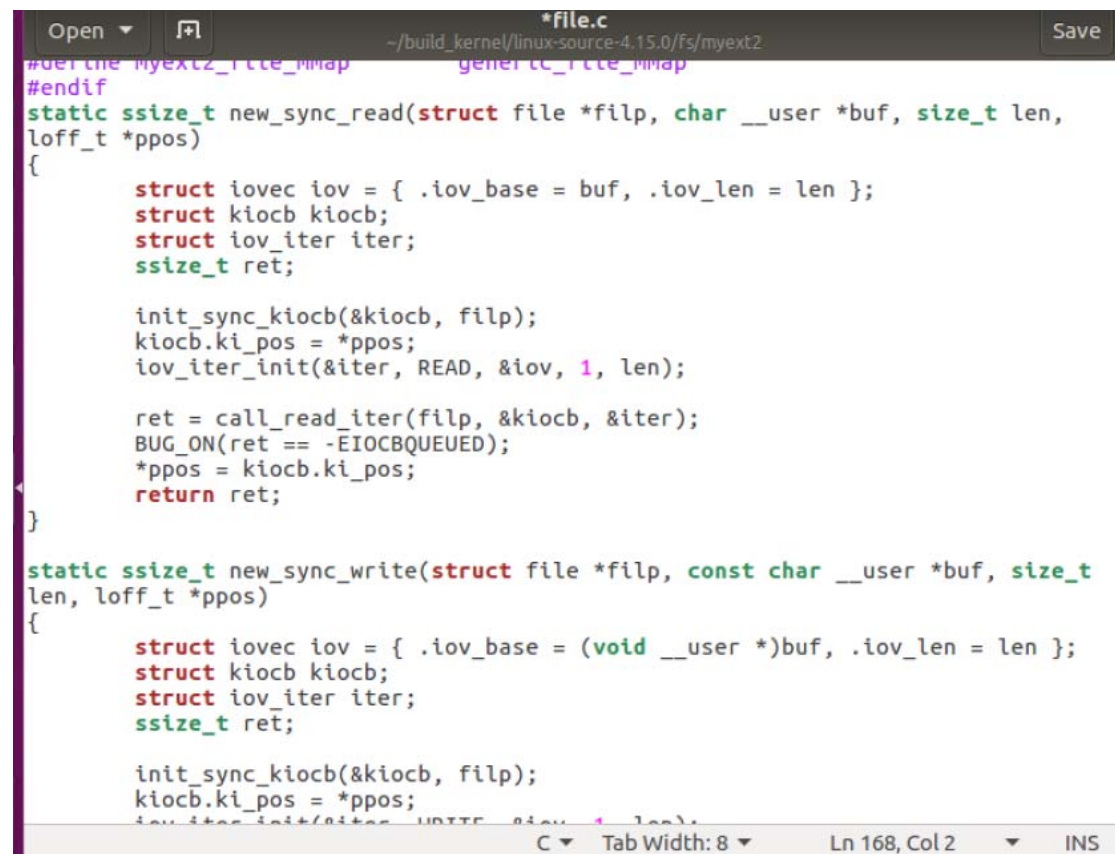
static ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t
len, loff_t *ppos)
{
    struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len };
    struct kiocb kiocb;
    struct iov_iter iter;
    ssize_t ret;

    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);
    if (ret > 0)
        *ppos = kiocb.ki_pos;
    return ret;
}

```


File.c:



```
Open  [icon] *file.c ~/build_kernel/linux-source-4.15.0/fs/myext2 Save
#ifdef MYEXT2_FUSE_MMAP generic_fuse_mmap
#endif
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 iovec iter;
    ssize_t ret;

    init_sync_kiocb(&kiocb, filp);
    kiocb.ki_pos = *ppos;
    iovec_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 iovec iter;
    ssize_t ret;

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

六、附录

相关命令记录:

umount: umount 命令用于卸载已经加载的文件系统。利用设备名或挂载点都能 umount 文件系统，不过最好还是通过挂载点卸载，以免使用绑定挂载（一个设备，多个挂载点）时产生混乱。

- a: 卸除/etc/mstab 中记录的所有文件系统;
- h: 显示帮助;
- n: 卸除时不要将信息存入/etc/mstab 文件中;
- r: 若无法成功卸除，则尝试以只读的方式重新挂入文件系统;
- t<文件系统类型>: 仅卸除选项中所指定的文件系统;
- v: 执行时显示详细的信息;
- V: 显示版本信息。

Mount: mount 命令用于加载文件系统到指定的加载点。此命令的最常用于挂载 cdrom，使我们可以访问 cdrom 中的数据，因为你将光盘插入 cdrom 中，Linux 并不会自动挂载，必须使用 Linux mount 命令来手动完成挂载。

- V: 显示程序版本;
- l: 显示已加载的文件系统列表;
- h: 显示帮助信息并退出;
- v: 冗长模式，输出指令执行的详细信息;

-n: 加载没有写入文件 “/etc/mstab” 中的文件系统;

-r: 将文件系统加载为只读模式;

-a: 加载文件 “/etc/fstab” 中描述的所有文件系统。

dd: dd 命令用于复制文件并对原文件的内容进行转换和格式化处理。dd 命令功能很强大的, 对于一些比较底层的问题, 使用 dd 命令往往可以得到出人意料的效果。用的比较多的还是用 dd 来备份裸设备。但是不推荐, 如果需要备份 oracle 裸设备, 可以使用 rman 备份, 或使用第三方软件备份, 使用 dd 的话, 管理起来不太方便。建议在有需要的时候使用 dd 对物理磁盘操作, 如果是文件系统的话还是使用 tar backup cpio 等其他命令更加方便。另外, 使用 dd 对磁盘操作时, 最好使用块设备文件。