Lab7

Lu Yiyun 518370910015

Hu Zhengdong 517370910249

1.What is a kernel module, and how does it different from a regular library?

A kernel module is an object file that contains code to extend the running kernel of an operating systems. It is a standalone-file, typically used to add support for new hardware.Kernel modules are usually stored in the **/lib/modules** subdirectories. The name of each subdirectory is based on the release number of the kernel.

Kernel modules are linked only to the kernel.Kernel modules do not link in the same libraries that user programs link in. The only functions a kernel module can call are functions that are exported by the kernel.

Without loadable kernel modules, an operating system would have to include all possible anticipated functionality compiled directly into the base kernel. Much of that functionality would reside in memory without being used, wasting memory, and would require that users rebuild and reboot the base kernel every time they require new functionality.

2.How to compile a kernel module?

An example:

```
obj-m += hello-1.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

3.How are mutex defined and used? How good is this approach? As he insists on the last question it clearly appears that he expects an honest response, not just “Of course dad your code is perfect, nobody can do better than you.”

Use static `DEFINE_MUTEX()` to define a mutex.

Use `mutex_lock_interruptible()` to lock a mutex.

Use `mutex_unlock()` to unlock a mutex.

It's good to define a mutex which can be interrupted by signals.

4.How is information shared between the kernel and user spaces?

`copy_to_iter(buffer, nbytes, to)` and `copy_from_iter(buffer, len, from)`

5.Changes made to dadfs

First, the error is located to `generic_write_checks(filp, ppos, &len, 0);`. However, in the linux version 5.4, the syntax is as follows:

```
ssize_t generic_write_checks(struct kiocb *, struct iov_iter *);
```

This leads to lots of related changes:

-
  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
  ssize_t dadfs_write(struct kiocb *iocb, struct iov_iter *from)
  #else
  ssize_t dadfs_write(struct file * filp, const char __user * buf, size_t len,
                 loff_t * ppos)
  #endif
  ```

  Following how `/fs/fuse/file.c` uses `generic_write_checks`, we add some new parameters in `dadfs_write`:

-
  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      struct file * filp = iocb->ki_filp;
      size_t len = iov_iter_count(from);
      loff_t *ppos = &(iocb->ki_pos);
  #endif
  ```

- Since `dadfs_write` syntax is changed, file operations must be changed, too. Following the definition in `/include/linux/fs.h`:

  ```
  const struct file_operations dadfs_file_operations = {
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      .read_iter = dadfs_read,
      .write_iter = dadfs_write,
  #else
      .read = dadfs_read,
      .write = dadfs_write,
  #endif
  };
  ```

- Then come back to `generic_write_checks`:

  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      retval = generic_write_checks(iocb, from);
  #else
      retval = generic_write_checks(filp, ppos, &len, 0);
  #endif
  ```

- Since `.read` is changed to `.read_iter`. Change `dadfs_read` like `dadfs_write` following how `/arch/s390/hypfs/inode.c` uses it

  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
  ssize_t dadfs_read(struct kiocb *iocb, struct iov_iter *to)
  #else
  ssize_t dadfs_read(struct file * filp, char __user * buf, size_t len,
                loff_t * ppos)
  #endif
  ```

  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      struct file *filp = iocb->ki_filp;
      size_t len = iov_iter_count(to);
      loff_t* ppos = &(iocb->ki_pos);
  #endif
  ```

- Then we notice `/arch/s390/hypfs/inode.c` uses `copy_to_iter`, we find that this is a new function that replaces `copy_to_user` defined in `/include/linux/uio.h`.

  Similarly, there is also a new function that replaces `copy_from_user` in `dadfs_write`, which is `copy_from_iter` . Then make some changes.

  Add a new header:

  ```
  #include <linux/uio.h>
  ```

  In `dadfs_write`:

  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      if (copy_from_iter(buffer, len, from)!=len) {
  #else
      if (copy_from_user(buffer, buf, len)) {
  #endif
  ```

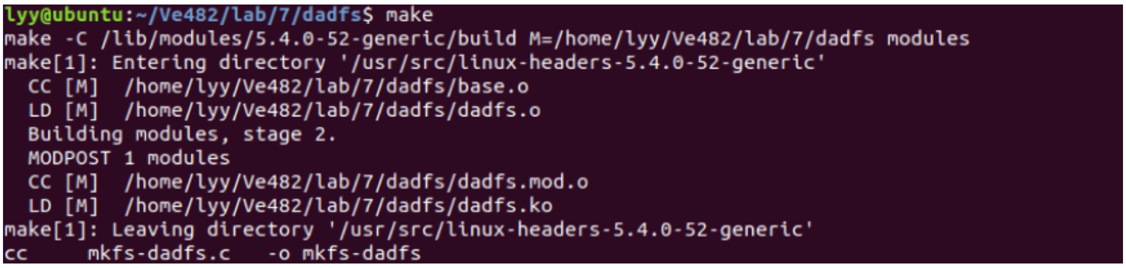
  In `dadfs_read`:

  ```
  #if LINUX_VERSION_CODE >= KERNEL_VERSION(3, 11, 0)
      if (!copy_to_iter(buffer, nbytes, to)) {
  #else
      if (copy_to_user(buf, buffer, nbytes)) {
  #endif
  ```

All the changes are done!

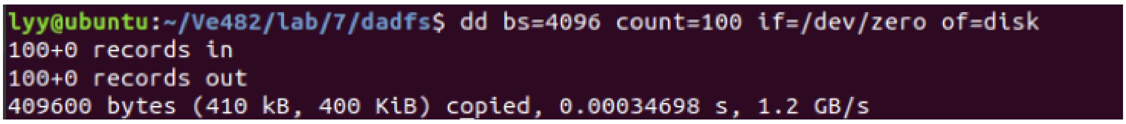
6.Simple process to test dadfs features:

- compile: make



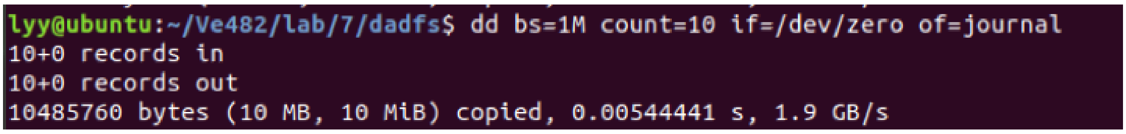
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ make
make -C /lib/modules/5.4.0-52-generic/build M=/home/lyy/Ve482/lab/7/dadfs modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-52-generic'
  CC [M]  /home/lyy/Ve482/lab/7/dadfs/base.o
  LD [M]  /home/lyy/Ve482/lab/7/dadfs/dadfs.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/lyy/Ve482/lab/7/dadfs/dadfs.mod.o
  LD [M]  /home/lyy/Ve482/lab/7/dadfs/dadfs.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-52-generic'
cc     mkfs-dadfs.c   -o mkfs-dadfs
```

- create a small virtual disk (to be formated in dadfs): dd bs=4096 count=100 if=/dev/zero of=disk



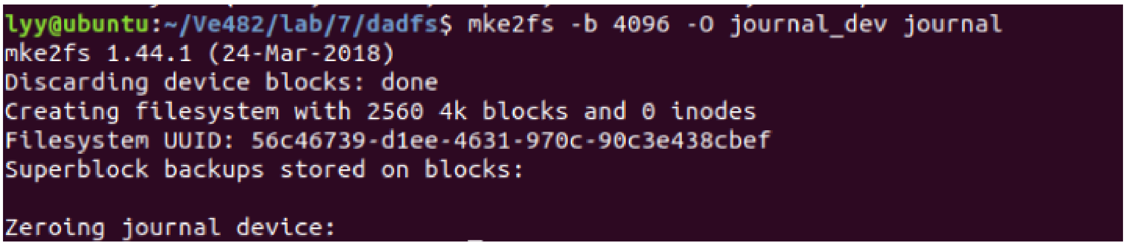
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ dd bs=4096 count=100 if=/dev/zero of=disk
100+0 records in
100+0 records out
409600 bytes (410 kB, 400 KiB) copied, 0.00034698 s, 1.2 GB/s
```

- create a small virtual disk (to be used as dadfs' journal): dd bs=1M count=10 if=/dev/zero of=journal



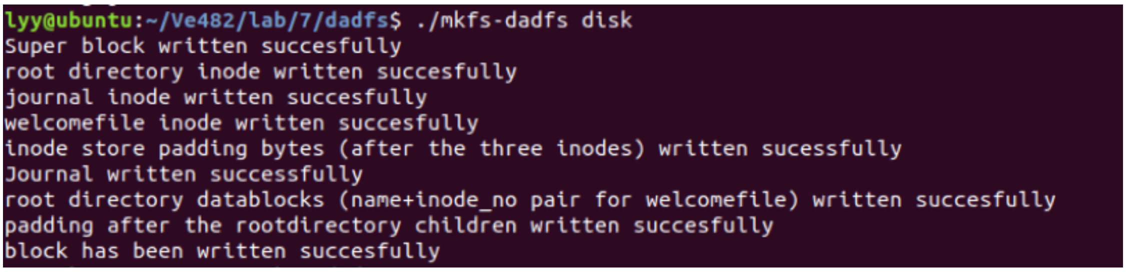
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ dd bs=1M count=10 if=/dev/zero of=journal
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.00544441 s, 1.9 GB/s
```

- initialise the journla: mke2fs -b 4096 -O journal_dev journal



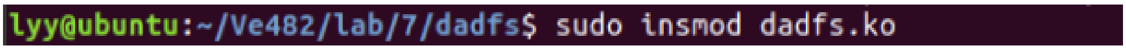
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ mke2fs -b 4096 -O journal_dev journal
mke2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 2560 4k blocks and 0 inodes
Filesystem UUID: 56c46739-d1ee-4631-970c-90c3e438cbef
Superblock backups stored on blocks:

Zeroing journal device:
```

- format the disk: ./mkfs-dadfs disk



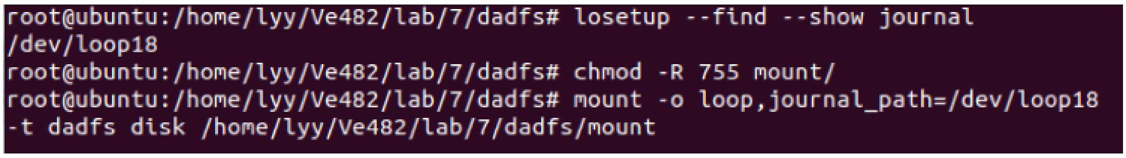
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ ./mkfs-dadfs disk
Super block written succesfully
root directory inode written succesfully
journal inode written succesfully
welcomefile inode written succesfully
inode store padding bytes (after the three inodes) written sucessfully
Journal written successfully
root directory datablocks (name+inode_no pair for welcomefile) written succesfully
padding after the rootdirectory children written succesfully
block has been written succesfully
```

- load dadfs module: insmod



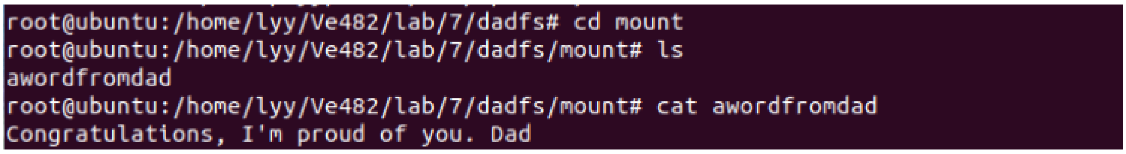
```
lyy@ubuntu:~/Ve482/lab/7/dadfs$ sudo insmod dadfs.ko
```

- mount disk: losetup, mount (loop,journal_path)



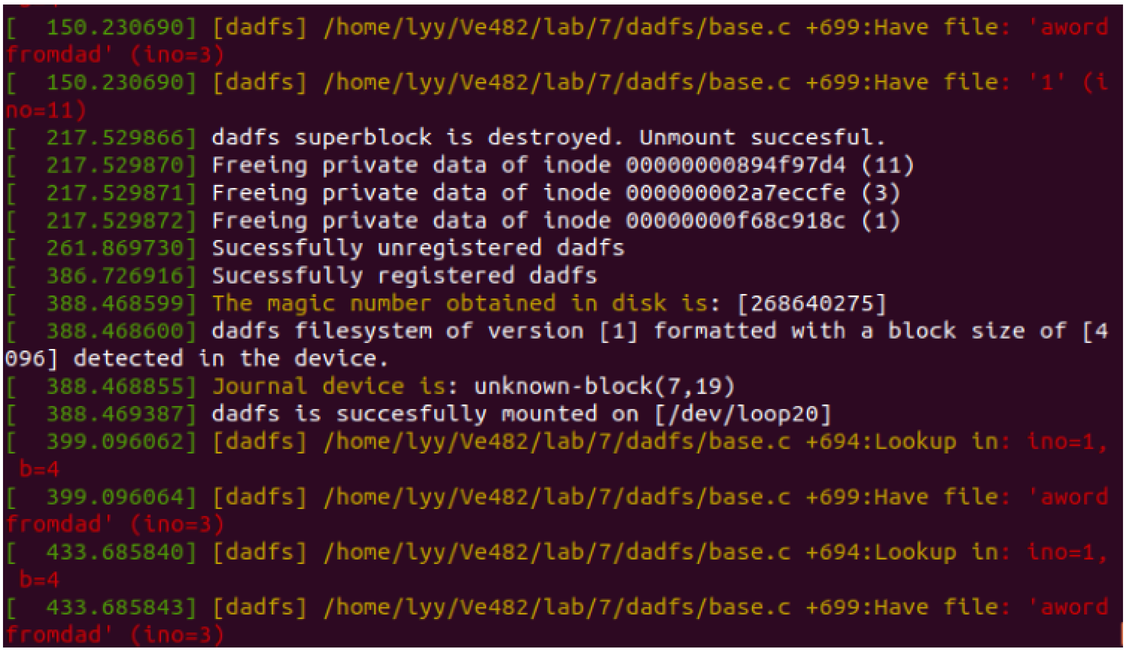
```
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# losetup --find --show journal
/dev/loop18
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# chmod -R 755 mount/
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# mount -o loop,journal_path=/dev/loop18
-t dadfs disk /home/lyy/Ve482/lab/7/dadfs/mount
```

- play with dad filesystem



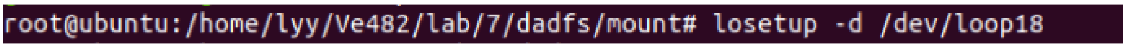
```
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# cd mount
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs/mount# ls
awordfromdad
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs/mount# cat awordfromdad
Congratulations, I'm proud of you. Dad
```

- check the logs: /var/log, dmesg

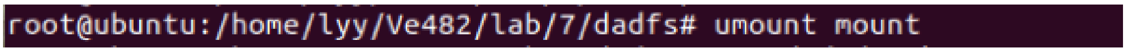


```
[  150.230690] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +699:Have file: 'aword
fromdad' (ino=3)
[  150.230690] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +699:Have file: '1' (i
no=11)
[  217.529866] dadfs superblock is destroyed. Unmount succesful.
[  217.529870] Freeing private data of inode 00000000894f97d4 (11)
[  217.529871] Freeing private data of inode 000000002a7eccfe (3)
[  217.529872] Freeing private data of inode 00000000f68c918c (1)
[  261.869730] Sucessfully unregistered dadfs
[  386.726916] Sucessfully registered dadfs
[  388.468599] The magic number obtained in disk is: [268640275]
[  388.468600] dadfs filesystem of version [1] formatted with a block size of [4
096] detected in the device.
[  388.468855] Journal device is: unknown-block(7,19)
[  388.469387] dadfs is succesfully mounted on [/dev/loop20]
[  399.096062] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +694:Lookup in: ino=1,
 b=4
[  399.096064] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +699:Have file: 'aword
fromdad' (ino=3)
[  433.685840] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +694:Lookup in: ino=1,
 b=4
[  433.685843] [dadfs] /home/lyy/Ve482/lab/7/dadfs/base.c +699:Have file: 'aword
fromdad' (ino=3)
```

- umount disk: losetup, umount

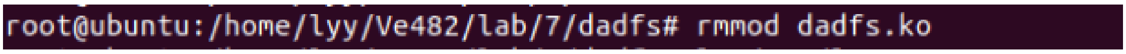


```
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs/mount# losetup -d /dev/loop18
```



```
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# umount mount
```

- unload module: rmmod



```
root@ubuntu:/home/lyy/Ve482/lab/7/dadfs# rmmod dadfs.ko
```