VE482 — Introduction to Operating Systems Lab 7

Group-pair 8: 518370910200 Shengyuan Xu, 518370910216 Xiangjie Li

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

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. A module can be configured as built-in or loadable. To dynamically load or remove a module, it has to be configured as a loadable module in the kernel configuration.

A shared library is another software module with these properties: The `.text` address space is accessible to all applications. The `.data` and `.bss` address space is in the same address space as the application that uses the shared library. So, they are shared in the sense that the `.text` is shared.

2 How to compile a kernel module?

1. edit Makefile:

```
obj-m = foo.o
KVERSION = $(shell uname -r)
all:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

2. Compile module using make command:

```
make
```

3. load it using `insmod` command. (root user is recommended)

```
insmod foo.ko
```

3 Write and test all the commands that are only hinted in the README file.

Done. Check the source code.

4 How are mutex defined and used? How good is this approach?

Three mutexs are defined:

- `dadfs_sb_lock`: Used for locking super block when action is taken on block
- `dadfs_inodes_mgmt_lock`: Used for locking super block when block inode need to be changed
- `dadfs_directory_children_update_lock`: Used for locking super block when children of the block is changed

Benefits of this approach:

- without these, troubles are likely to occur during the operation of the filesystem.
- inhance the efficiency of the code.

5 How is information shared between the kernel and user spaces?

By use the kernel APIs for manipulating user memory, such as `copy_to_iter` and `copy_from_iter`. The source codes are as follows.

```
// Copies a block of data from the kernel to user space
```

```
size_t copy_to_iter(const void *addr, size_t bytes, struct iov_iter *i)
{
  const char *from = addr;
  iterate_and_advance(i, bytes, v,
    __copy_to_user(v.iov_base, (from += v.iov_len) - v.iov_len,
             v.iov_len),
    memcpy_to_page(v.bv_page, v.bv_offset,
             (from += v.bv_len) - v.bv_len, v.bv_len),
    memcpy(v.iov_base, (from += v.iov_len) - v.iov_len, v.iov_len)
  )

  return bytes;
}
EXPORT_SYMBOL(copy_to_iter);

// Copies a block of data from user space to the kernel
size_t copy_from_iter(void *addr, size_t bytes, struct iov_iter *i)
{
  char *to = addr;
  iterate_and_advance(i, bytes, v,
    __copy_from_user((to += v.iov_len) - v.iov_len, v.iov_base,
         v.iov_len),
    memcpy_from_page((to += v.bv_len) - v.bv_len, v.bv_page,
         v.bv_offset, v.bv_len),
    memcpy((to += v.iov_len) - v.iov_len, v.iov_base, v.iov_len)
  )

  return bytes;
}
EXPORT_SYMBOL(copy_from_iter);
```

6 Change

```
17,20d16
< #include <linux/uio.h>
< #include <linux/kmod.h>
< #include <linux/blktrace_api.h>
<
275d270
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
278,280d272
< #else
< ssize_t dadfs_read(struct kiocb *kiocb, struct iov_iter *to)
< #endif
285,291d276
< #if LINUX_VERSION_CODE >= KERNEL_VERSION(3,11,0)
<     struct file* filp = kiocb->ki_filp;
```

```
<         loff_t *ppos = &(kiocb->ki_pos);
<         size_t len = iov_iter_count(to);
< #endif
<
<
316,321c301
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
<     if (copy_to_user(buf, buffer, nbytes))
< #else
<     if(!copy_to_iter(buffer,len,to))
< #endif
<     {
---
>         if (copy_to_user(buf, buffer, nbytes)) {
375d354
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
378,380d356
< #else
< ssize_t dadfs_write(struct kiocb* kiocb, struct iov_iter * from)
< #endif
396,403c372
< #if LINUX_VERSION_CODE >= KERNEL_VERSION(3,11,0)
<     struct file* filp = kiocb->ki_filp;
<         loff_t *ppos = &(kiocb->ki_pos);
<         size_t len = iov_iter_count(from);
< #endif
<     sb = filp->f_inode->i_sb;
<
<
---
>         sb = filp->f_path.dentry->d_inode->i_sb;
409,420c378,380
<
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
<     retval = generic_write_checks(filp, ppos, &len, 0);
< #else
<     retval = generic_write_checks(kiocb,from);
< #endif
< # if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
<     if (retval){
< #else
<     if(!retval){
< #endif
<         return retval;}
---
>         retval = generic_write_checks(filp, ppos, &len, 0);
>         if (retval)
>                 return retval;
435d394
```

```
<
445,450c404,405
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
<     if (copy_from_user(buffer, buf, len))
< #else
<     if (!copy_from_iter(buffer,len,from))
< #endif
<     {
---
>
>         if (copy_from_user(buffer, buf, len)) {
495,502c450,451
<
< #if LINUX_VERSION_CODE < KERNEL_VERSION(3,11,0)
<         .read = dadfs_read,
<             .write = dadfs_write,
< #else
<         .read_iter = dadfs_read,
<         .write_iter = dadfs_write,
< #endif
---
>         .read = dadfs_read,
>         .write = dadfs_write,
773c722,723
<
---
>         printk(KERN_INFO "Journal device is: %s\n", __bdevname(dev, b));
>
775d724
<         printk(KERN_INFO "Journal device is: %s\n", bdevname(bdev, b));
1039d987
<
```

7 Test result

0 install dependencies

Done.

1 compile: make

```
bill@bill-virtual-machine:~/Desktop/lab7/dadfs$ make
make -C /lib/modules/5.11.0-40-generic/build M=/home/bill/Desktop/lab7/dadfs modules
make[1]: Entering directory '/usr/src/linux-headers-5.11.0-40-generic'
  CC [M]  /home/bill/Desktop/lab7/dadfs/base.o
  LD [M]  /home/bill/Desktop/lab7/dadfs/dadfs.o
  MODPOST /home/bill/Desktop/lab7/dadfs/Module.symvers
  CC [M]  /home/bill/Desktop/lab7/dadfs/dadfs.mod.o
  LD [M]  /home/bill/Desktop/lab7/dadfs/dadfs.ko
  BTF [M] /home/bill/Desktop/lab7/dadfs/dadfs.ko
Skipping BTF generation for /home/bill/Desktop/lab7/dadfs/dadfs.ko due to unavailability of
 vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.11.0-40-generic'
cc     mkfs-dadfs.c    -o mkfs-dadfs
```

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

```
bill@bill-virtual-machine:~/Desktop/lab7/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, 0.000551509 s, 743 MB/s
```

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

```
bill@bill-virtual-machine:~/Desktop/lab7/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, 0.0075165 s, 1.4 GB/s
```

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

```
bill@bill-virtual-machine:~/Desktop/lab7/dadfs$ mke2fs -b 4096 -O journal_dev journal
mke2fs 1.45.5 (07-Jan-2020)
Discarding device blocks: done
Creating filesystem with 2560 4k blocks and 0 inodes
Filesystem UUID: cb85f9a6-81c3-4126-babc-afb47b22af50
Superblock backups stored on blocks:

Zeroing journal device:
```

5 format the disk: ./mkfs-dadfs disk

```
bill@bill-virtual-machine:~/Desktop/lab7/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
```

6 load dadfs module: insmod

```
bill@bill-virtual-machine:~/Desktop/lab7/dadfs$ sudo insmod dadfs.ko
[sudo] password for bill:
```

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

```
bill@bill-virtual-machine:~/Desktop/lab7/dadfs$ sudo losetup -f --show journal
/dev/loop9
```

```
bill@bill-virtual-machine:~/Desktop/lab7/dadfs$ sudo -i && cd /tmp/
root@bill-virtual-machine:~# mount -l
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,noexec,relatime,size=964700k,nr_inodes=241175,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=199488k,mode=755,inode64)
/dev/sda5 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,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755,inode64)
cgroup2 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)
none on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
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/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
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)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=28,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=24285)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
/var/lib/snapd/snaps/bare_5.snap on /snap/bare/5 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/core18_2246.snap on /snap/core18/2246 type squashfs (ro,nodev,relatime,x-gdu.hide)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
vmware-vmblock on /run/vmblock-fuse type fuse.vmware-vmblock (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other)
/var/lib/snapd/snaps/core18_2128.snap on /snap/core18/2128 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/gtk-common-themes_1519.snap on /snap/gtk-common-themes/1519 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/gnome-3-34-1804_72.snap on /snap/gnome-3-34-1804/72 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/gtk-common-themes_1515.snap on /snap/gtk-common-themes/1515 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/snapd_12704.snap on /snap/snapd/12704 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/snapd_13640.snap on /snap/snapd/13640 type squashfs (ro,nodev,relatime,x-gdu.hide)
/var/lib/snapd/snaps/snap-store_547.snap on /snap/snap-store/547 type squashfs (ro,nodev,relatime,x-gdu.hide)
/dev/sda1 on /boot/efi type vfat (rw,relatime,fmask=0077,dmask=0077,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=199488k,mode=700,uid=1000,gid=1000,inode64)
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/bill/Ubuntu 20.04.3 LTS amd64 type iso9660 (ro,nosuid,nodev,relatime,nojoliet,check=s,map=n,blocksize=2048,uid=1000,gid=1000,dmode=500,fmode=400,iocharset=utf8,uhelper=udisks2) [Ubuntu
 20.04.3 LTS amd64]
/home/bill/Desktop/lab7/dadfs/disk on /tmp type dadfs (rw,nosuid,nodev,noexec,relatime)
```

8 play with dad filesystem: mkdir, mv, cp, cat, rm, ls, cd, touch, etc.

```
root@bill-virtual-machine:~# cd /tmp/
root@bill-virtual-machine:/tmp# cat awordfromdad
Congratulations, I'm proud of you. Dad
```

9 check the logs: /var/log, dmesg

```
[    6.122235] kernel: NET: Registered protocol family 40
[    6.727120] kernel: Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[    6.727126] kernel: Bluetooth: BNEP filters: protocol multicast
[    6.727129] kernel: Bluetooth: BNEP socket layer initialized
[    7.222075] kernel: e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[    7.223186] kernel: IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[    9.123887] kernel: Bluetooth: RFCOMM TTY layer initialized
[    9.123894] kernel: Bluetooth: RFCOMM socket layer initialized
[    9.123898] kernel: Bluetooth: RFCOMM ver 1.11
[    9.331854] kernel: loop9: detected capacity change from 0 to 8
root@bill-virtual-machine:/tmp# cat /var/log/dmesg | grep loop9
[    9.331854] kernel: loop9: detected capacity change from 0 to 8
```

10 umount disk: losetup, umount

Done.

11 unload module: rmmod

Done.