VE482 - Lab 7

Introduction to Operating Systems

Weili Shi 519370910011

Kexuan Huang 518370910126

November 18, 2021

2 Fixing Dadfs

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

A kernel module (or loadable kernel mode) is an object file that contains code that can extend the kernel functionality at runtime (it is loaded as needed); When a kernel module is no longer needed, it can be unloaded. In this way, when a new kernel module is introduced, the kernel doesn't need to be re-compiled. However, regular libraries are collections of prewritten code that users can use to optimize tasks, when a library of the kernel is added or modified, the kernel needs to be re-compiled and reboot.

2. How to compile a kernel module?

Compiling a kernel module differs from compiling an user program. First, other headers should be used. Also, the module should not be linked to libraries. And, last but not least, the module must be compiled with the same options as the kernel in which we load the module. For these reasons, there is a standard compilation method (kbuild). This method requires the use of two files: a `Makefile` and a `Kbuild` file.

Sample `Makefile`:

```
KDIR = /lib/modules/`uname -r`/build

kbuild:
        make -C $(KDIR) M=`pwd`

clean:
        make -C $(KDIR) M=`pwd` clean
```

Sample `Kbuild` file:

```
EXTRA_CFLAGS = -Wall -g
obj-m        = modul.o
```

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

```
#!/bin/bash

# compile
```

```
make

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

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

# initialise the journal
mke2fs -b 4096 -O journal_dev journal

# format the disk
./mkfs-dadfs disk

# load dadfs module
sudo insmod dadfs.ko

# mount disk
losetup -f --show journal
sudo mount -o loop,owner,group,users,journal_path=/dev/loop10 -t dadfs
    disk /tmp

# play with dad filesystem
sudo ls /tmp
sudo cat /tmp/awordfromdad

# check the logs: /var/log, dmesg
cat /var/log/dmesg

# umount disk
sudo umount /dev/loop10 /tmp

# unload module
sudo rmmod dadfs
```

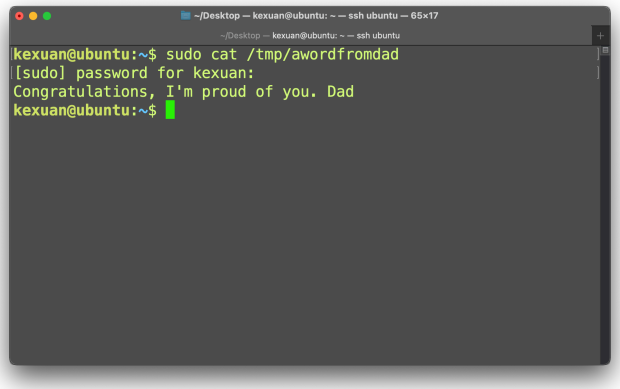

Figure 1: “I’m proud of you, Dad”

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

- `dadfs_sb_lock`: lock super block when action is on block
- `dadfs_inodes_mgmt_lock`: lock super block when block inode to be changed
- `dadfs_directory_children_update_lock`: lock super block when children block is changed

This approach improves efficiency by separating mutex into 3 aspects, as well as makes the locking logic clear at the same time.

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

With `mmap`, userspace process can access the memory allocated by the kernel in kernel space via a typical memory mapping mechanism to its user address space.
`mmap()` creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in `addr`. The length argument specifies the length of the mapping (which must be greater than 0). If `addr` is `NULL`, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If `addr` is not `NULL`, then the kernel takes it as a hint about where to place the mapping.

6. Document your changes in the README file.

Please refer to `dadfs/README.md`