Shanghai Jiao Tong University

CS353 Linux Kernel

Project 4: File System

Chao Gao

5142029014

May 26, 2017

1 Project description

In this project, we are required to implement three practices.
Source
super.c/Makefile(kernel source of romfs)
Test.img(a romfs image, you can mount it to a dir with 'mount -o loop test.img xxx')
Practices
1. Change romfs code to hide a file/dir with special name.
2. Change romfs code to correctly read info of "encrypted" romfs. For example, change bb's content 'bbbbb' to 'ccccc'.
3. Change romfs code to add 'x'(execution) bit for a special file, change '-rw-r–r–' to '-rwxr-xr-x'.

2 Environment

Virtual OS: Ubuntu 16.04.
kernel version: 4.9.13.

3 Introduction

This part is actually the knowledge preparation for this project.

3.1 VFS

VFS(virtual file system), is a common interface to several kinds of filesystems, enables system calls such as open(), read(), and write() to work regardless of file system or storage media. The important thought is **common file model**, which is capable of representing all supported file systems, consists of superblock object, inode object, file object, dentry object.

3.2 Romfs

Romfs(ROM filesystem) is an extremely simple file system lacking many features, intended for burning important files onto an EEPROM. It is very useful as an

initial ROM holding kernel modules that can be loaded later as needed. It is very small, code wise.

3.3 genromfs

genromfs is used to create a romfs file system image, usually directly on a block device, or for test purpose, in a plain file. It is the mkfs equivalent of other filesystems. genromfs will scan the current directory and its subdirectories, build a romfs image from the files found, and output it to the file or device you specified.

3.4 Mount

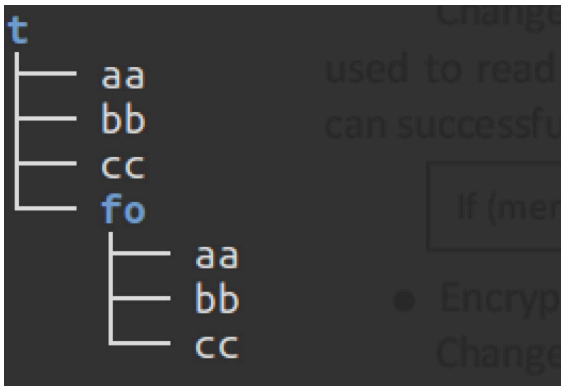
Mounting takes place before a computer can use any kind of storage device. The user or their operating system must take it accessible through the computer's file system. A user can only access files on mounted media.
A mount point is a physical location in the partition used as a root filesystem.

4 Implementation

Actually the implementation is easy, just some modifications. Here I will present the whole implementation of this porject.

4.1 Preparation

Create the folder **t**.



Generate the romfs image.

```
genromfs -f test.img -d t
```

Copy the source code **fs/romfs** folder to the work directory, the modification is mainly in the file **super.c**.

4.2 Declaration

```
static char* hided_file_name;
static char* encryp_file_name;
static char* exec_file_name;

module_param(hided_file_name, charp, 0644);
module_param(encryp_file_name, charp, 0644);
module_param(exec_file_name, charp, 0644);
```

4.3 hide a file with a special name

In **romfs_readdir** function, add the statement. This function is used to read the files in the directory, by checking the file name, we can simply hide a file with a special name.

```
if(strcmp(fsname, hided_file_name) == 0)
{
    offset = nextfh & ROMFH_MASK;
    continue;
}
```

4.4 encrypted romfs

In **romfs_readpage** funtion, add the statement. This function is used to read the content of a file. Here I change the content all to 'c'.

```
if(strcmp(fsname, encryp_file_name) == 0)
{
    memset(buf, 'c', fillsize -1);
    *((char *)buf + fillsize -1) = '\n';
}
```

4.5 add execution bit

In **romfs_iget** function, add the statement. This function is used to get a romfs inode based on its position in the image.

```
if(strcmp(fsname, exec_file_name) == 0)
{
    mode |= S_IXUGO;
}
```

5 Test result

Modify the Makefile.

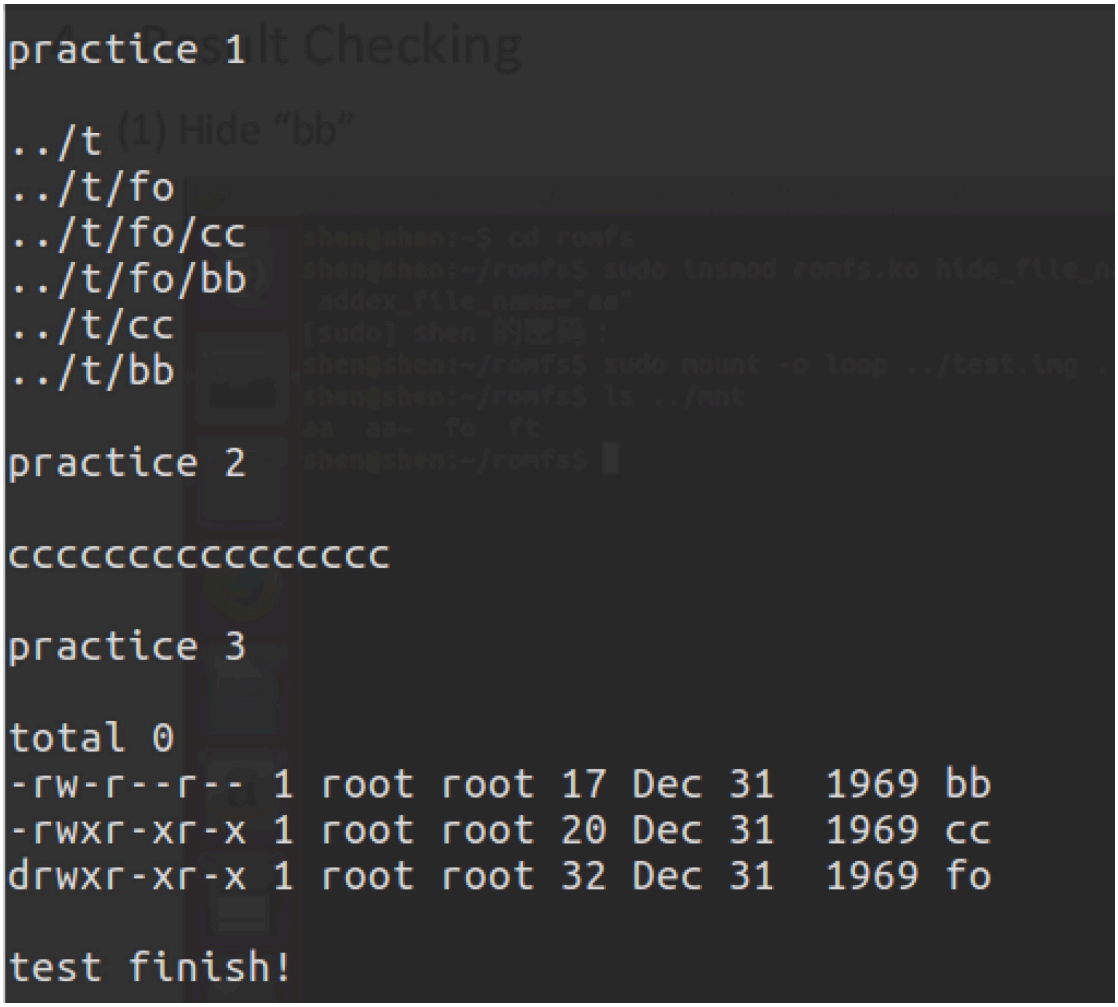
```
obj-$(CONFIG_ROMFS_FS) += romfs.o
romfs-y := storage.o super.o
KDIR := /lib/modules/$(shell uname -r)/build
EXTRA_FLAGS := -I(PWD)
PWD := $(shell pwd)

ifneq ($(CONFIG_MMU),y)
romfs-$(CONFIG_ROMFS_ON_MTD) += mmap-nommu.o
endif

all:
    make -C $(KDIR) M=$(PWD) modules
clean:
```

```
14      make -C $(KDIR) M=$(PWD) clean
```

Write a simple shell script to test the result.

```
1  find ../t
2  cat ../t/bb
3  ls -l ../t
4  make
5  insmod romfs.ko hided_file_name="aa" encryp_file_name=
6      "bb" exec_file_name="cc"
7  mount -o loop ../test.img ../t
8  echo "\npractice 1\n"
9  find ../t
10 echo "\npractice2 \n"
11 cat ../t/bb
12 echo "\npractice 3\n"
13 ls -l ../t
14 echo "\ntest finish!\n"
15 umount ../test.img
16 rmmod romfs.ko
17 make clean
```

We can get the following result.



6 Summary

Although understanding linux virtual file system is a hard job, I did not give up and really learned a lot. And the usage of some functions required me to read the source code in linux kernel, which helped me get to know its architecture and how it is implemented. The simple practices help me familiar with the file system.

Thanks a lot to Prof.Chen for impressive teaching about file system and TAs for answering questions.