CS353 Linux 内核 Project 4

实验内容

以Linux内核中的`/fs/romfs`作为文件系统源码基础，修改并编译生成模块`romfs.ko`，实现以下功能：

- `romfs.ko`接受三个参数：`hided_file_name`，`encrypted_file_name`和`exec_file_name`
  - `hided_file_name=xxx`：需要被隐藏的文件路径
  - `encrypted_file_name=xxx`：需要被加密的文件路径
  - `exec_file_name=xxx`：需要修改权限为可执行的文件路径

通过`insmod romfs.ko`安装修改后的romfs模块，使用`genromfs`生成格式为romfs的镜像文件，使用`mount`命令挂载镜像文件并验证修改是否成功。

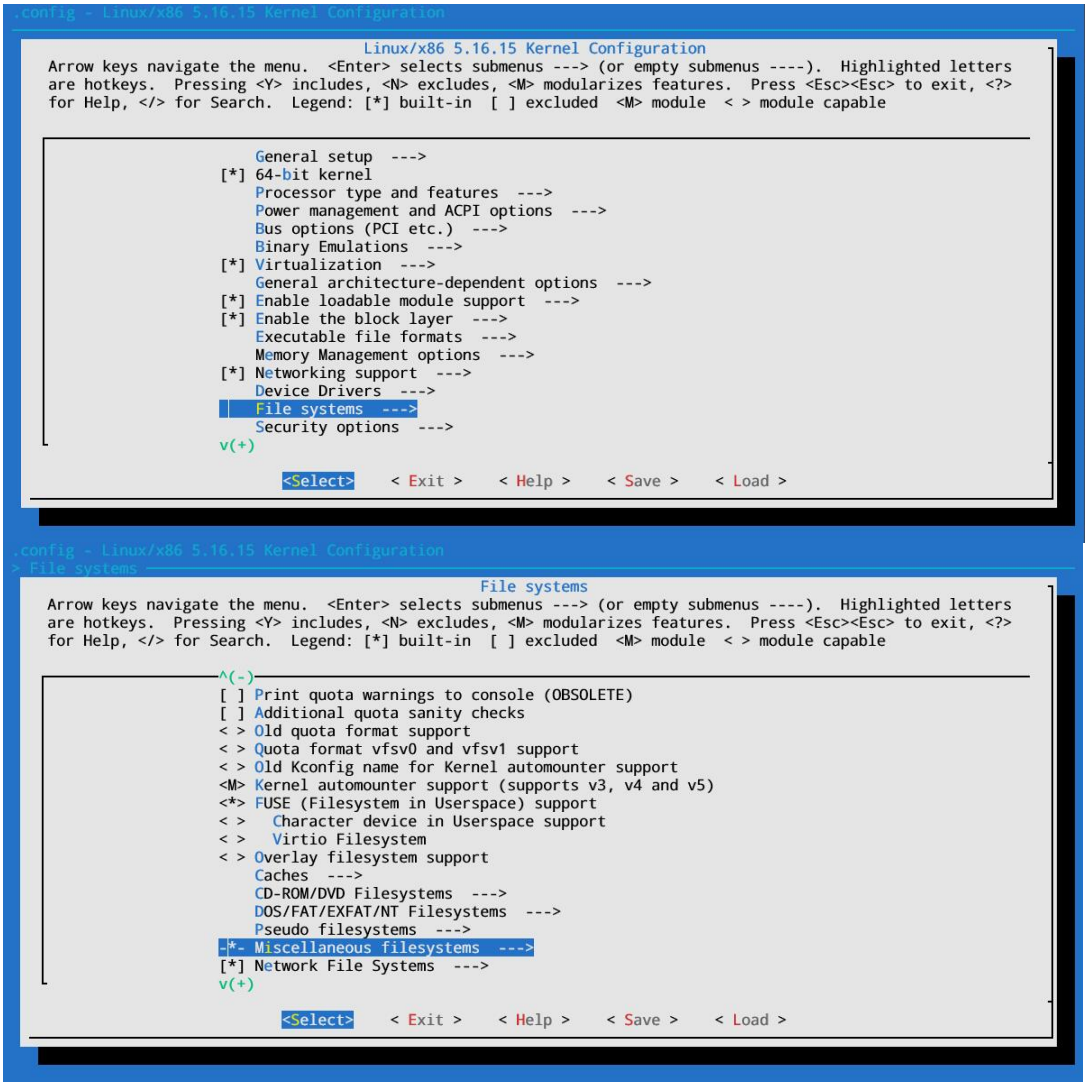
实验思路

```
/fs/romfs/super.c

// 0.在合适的位置声明模块参数

// 1.实现隐藏文件功能
static int romfs_readdir(struct file *file, struct dir_context *ctx) {
    // TODO:如果文件名与要隐藏的文件名相同，则跳过该文件
}

// 2.实现加密文件的功能
static int romfs_readpage(struct file *file, struct page *page) {
    // TODO:1.获取文件名 2.如果文件名与目标相同，则修改文件内容
}

// 3.实现修改权限的功能
static struct dentry *romfs_lookup(struct inode *dir, struct dentry *dentry,
unsigned int flags) {
    // TODO:如果文件名与目标相同，则修改文件的的权限为可执行
}
```

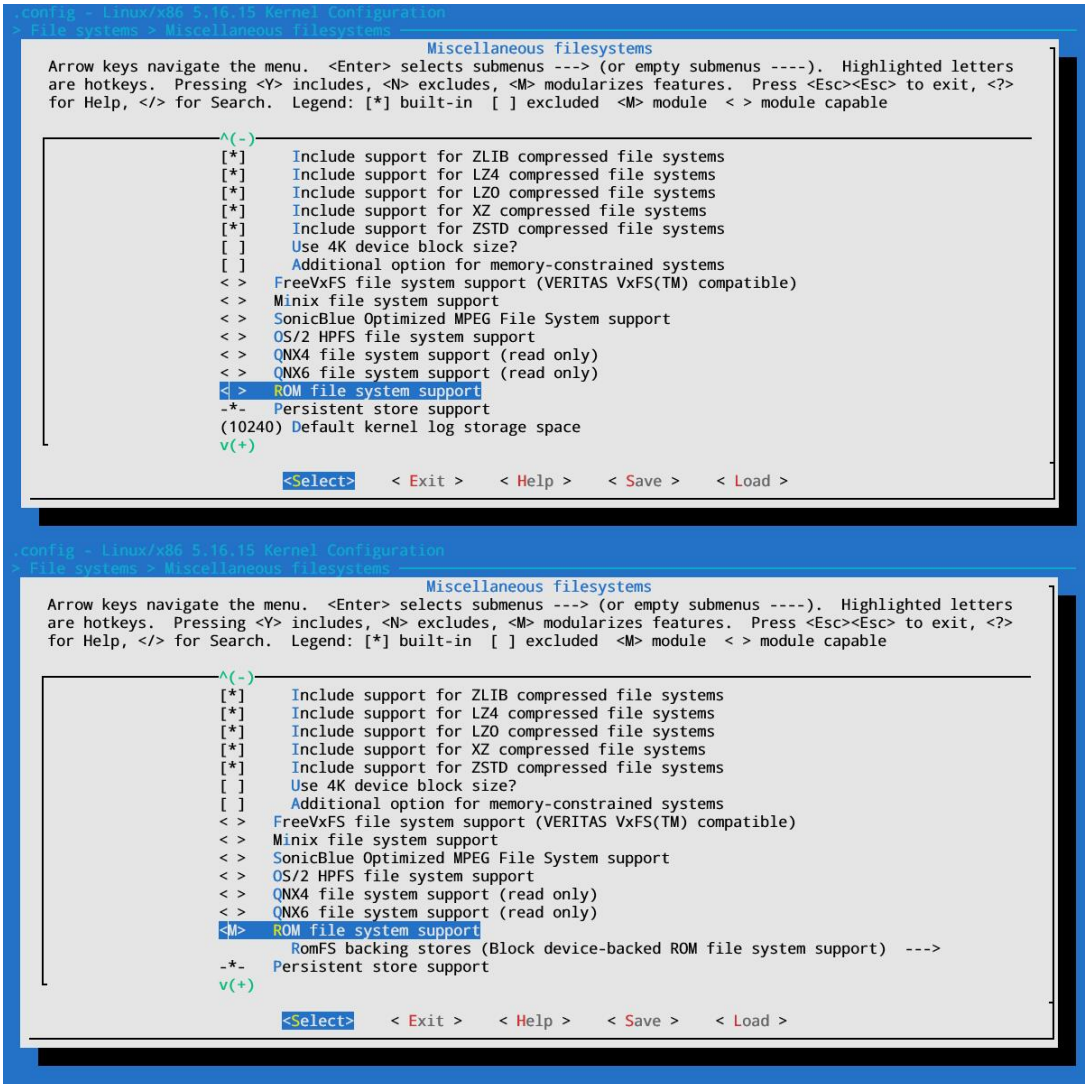
注：以上模板仅供参考，也可以完全按照自己的思路来实现

编译安装

在源码目录下`make menuconfig`，按照下图依次选择：*File systems* → *Miscellaneous filesystems* → *ROM file system support* → **change to M**


.config - Linux/x86 5.16.15 Kernel Configuration
Linux/x86 5.16.15 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General setup --->
[*] 64-bit kernel
Processor type and features --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Binary Emulations --->
[*] Virtualization --->
General architecture-dependent options --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
Executable file formats --->
Memory Management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Security options --->
v(+)
<Select> < Exit > < Help > < Save > < Load >

.config - Linux/x86 5.16.15 Kernel Configuration
> File systems
File systems
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
^(-)
[ ] Print quota warnings to console (OBSOLETE)
[ ] Additional quota sanity checks
< > Old quota format support
< > Quota format vfsv0 and vfsv1 support
< > Old Kconfig name for Kernel automounter support
<M> Kernel automounter support (supports v3, v4 and v5)
<*> FUSE (Filesystem in Userspace) support
< > Character device in Userspace support
< > Virtio Filesystem
< > Overlay filesystem support
Caches --->
CD-ROM/DVD Filesystems --->
DOS/FAT/EXFAT/NT Filesystems --->
Pseudo filesystems --->
-*- Miscellaneous filesystems --->
[*] Network File Systems --->
v(+)
<Select> < Exit > < Help > < Save > < Load >



保存退出后，重新编译内核。编译完成后`romfs.ko`位于`/fs/romfs`目录下。之后也可使用命令

`make CONFIG_ROMFS_FS=m -C <path to linux source code> M=<path to linux source code>/fs/romfs modules`单独编译模块

安装及测试

github仓库中提供了`test.img`，镜像中包括`aa` `bb`和`cc`三个文件，`bb`中的内容为 *A message sent from Alice to Bob*，`cc`的执行结果会输出 *pass*

```
insmod romfs.ko hided_file_name=aa encrypted_file_name=bb exec_file_name=cc #安装模块
mount -o loop test.img /mnt -t romfs #挂载镜像到/mnt下
ls -l /mnt #查看/mnt目录下的文件 发现aa文件不存在
cat /mnt/bb #查看bb内容，应该输出加密后的内容
/mnt/cc #输出pass
```

#

实验提示

1. `romfs`的主要功能都在`/fs/romfs/super.c`中实现，在该文件中通过`module_param`接收参数
2. 阅读`super.c`，`romfs`是如何查找目录下的文件的？如果想要隐藏一个文件，那么在函数中，当我们发现某个文件名是我们想要隐藏的文件，要如何修改使得系统"忽略"该文件？

3. `romfs`是通过哪一个函数来读取某一个文件的具体内容？如何从文件的inode获取文件名？文件的内容被读取到哪里？通过对buffer内容修改进行加密操作。（注意这里加密只是一个概念性操作，采用简单的古典加密方法即可）
4. 文件的权限信息在inode中存储，我们需要在哪个函数中去修改文件的权限信息呢？
5. `kmap`：建立物理页的虚拟地址的映射

实验提交

提交渠道：Canvas

提交文件：`学号_project4.zip`，其中包含修改后的`super.c`文件以及实验报告`学号_project4_report.pdf`

`super.c`文件中应当有适当的注释来说明修改的部分，实验报告内容包括但不限于对设计函数的说明、实验效果截图、实验心得（实验过程中遇到的困难、解决的方法，或者是值得分享的小技巧）