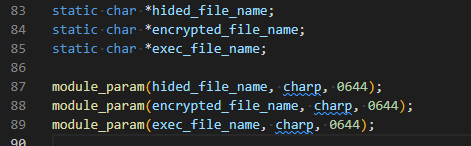
LAB4

吴非 519021910924

内核环境：5.5.11-050511-generic

首先根据提示复制super.c到实验目录，第一步要做的事首先根据传参设置三个参数：



实验一使用过此函数：

module\_param (name, type, perm)是一个宏，表示向当前模块传入参数

charp/\*一个字符指针值. 内存为用户提供的字串分配, 指针因此设置.\*/

下面根据三个功能依次阐述：

1．隐藏文件：

romfs\_readdir : read the entries from a directory

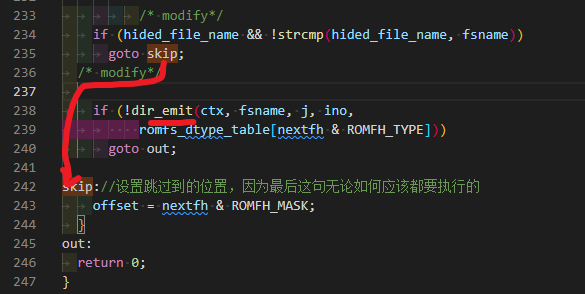
.iterate\_shared = romfs\_readdir,

iterate\_shared: called when the VFS needs to read the directory contents

when filesystem supports concurrent dir iterators

可见这个函数的作用是读取文件夹下文件时（ls -l /mnt）遍历(iterate)给定目录下的所有文件，这一点也可以从函数内的for循环得知。

而我需要做的工作就是在for循环中找到每个文件名fsname并和我们传入的文件名称参数比较，得到文件名后我们可以发现下面有个显眼的函数dir\_emit（查找其用法，读目录项，然后调用dir\_emit函数填充至用户空间，也就是显示在终端） 那么我们要做的工作就是如果文件名匹配跳过这个函数的执行，这也决定了我插入的位置就在这个函数之前，即实现了隐藏，代码如下。



2 文件加密：

romfs\_readpage read a page worth of data from the image

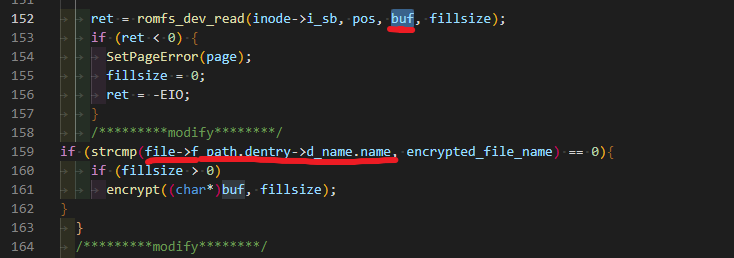
.readpage = romfs\_readpage

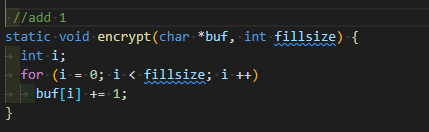
readpage: called by the VM to read a page from backing store.

虽然描述有些模糊，但是可以推测出应该是cat的时候被调用了（此时已经存在于backing store中了）

由注释可以大致知道这个函数是读取文件内容的，根据实验展示我们需要判断是否是加密文件，如果是，我们把文件的内容用一个简单的加密算法计算后然后再输出给用户。

其中第一步，根据提示我们需要根据inode找到文件名，我留意到romfs\_readdir函数中有获取文件名的函数，主要通过romfs\_dev\_strnlen和romfs\_dev\_read函数实现，观察到缺少offset变量，但是这个变量经尝试没找到怎么获得。转变思路从file结构体入手，此结构体中没有对应的文件名，查看其结构体中的第二个成员[f\_path](https://docs.huihoo.com/doxygen/linux/kernel/3.7/structfile.html#afe3cb672a6728a24b3031327966b041c)，在其结构体中看到了熟悉的dentry成员，继续搜索相关dentry结构体信息，In the path /mnt/cdrom/foo, the components /, mnt, cdrom, and foo are all dentry objects.，以及struct qstr d\_name; /\* dentry name \*/ 可以看到这个成员就是我们要找的文件名，之后通过fillsize = size > PAGE\_SIZE ? PAGE\_SIZE : size;和下面的读取函数romfs\_dev\_read可以得到fillsize是最后读出的数据量，由于我们的数据是char占一个字节，所以直接用fillsize为循环终止数即可。最后实际的操作我直接把所有的char加1视作加密了，插入的位置根据提示，直接加在读出数据到buf的函数后面即可，代码如图：





3. 修改权限：

.lookup = romfs\_lookup,

lookup: called when the VFS needs to look up an inode in a parent directory.

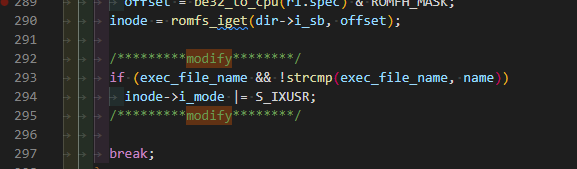
我暂且认为这个函数在我挂载镜像的时候就执行了。

和上两个功能一样，需要找到文件名，可以看到源代码中直接给出了，因此我们只需要修改权限即可。而本函数的所有变量中只有两个可能与文件权限有关，inode，dentry。搜索inode结构体，第一个成员就是叫i\_mode的。

<https://blog.csdn.net/jinking01/article/details/105771173>得知确实和权限相关，参考其他地方的linux源码，通过或操作实现给权限的功能，

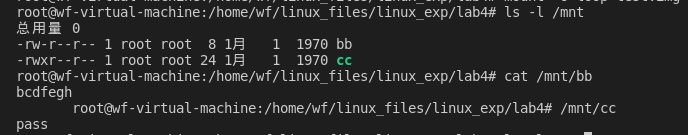
因此通过此方式给S\_IXUSR权限即可。

最后是mount -o loop test.img /mnt -t romfs #挂载镜像到/mnt下，这个命令，-o代表选项，loop为伪设备，（<https://zh.wikipedia.org/wiki/Loop%E8%AE%BE%E5%A4%87>） （-t vfstype 指定文件系统的类型，通常不必指定，mount 会自动选择正确的类型。）最后插入的位置也就合理的选在获取inode变量代码之后。如下：



结果：

使用run.sh中的相关命令编译后：



总结：

本次实验基于 Linux 内核虚拟文件系统的相关理论，实践了 Linux 中各种用于文件管理的数据结构和操作，提高了我阅读分析大规模系统软件源码的能力和调试能力。虽然相对前几次实验较简单，但是也有一些函数和数据结构比较像，需要仔细查找资料区分和应用。

实验过程中的问题：

开始我只用了super.c编译，但是提示没有internal.h,从usr/src/目录中提取此文件复制到实验目录下，而又发现另一个报错说没有一些结构体，查阅后发现在storage.c中，遂复制一份并修改makefile文件一起编译。

然后就是一些API函数的功能查找和修改，总体没有耗费太多时间。