1. 概述

本实验实现了一个钩子，在vfs调用的ramfs文件系统实现的函数中加入了钩子，并提供方便的可注册钩子函数的代码段。方便在某些特定时刻如创建inode、删除inode时，调用一些用户自定义的函数，实现一些功能。本实验实现了一个最简单的利用钩子来统计inode数目和占用空间的功能。

1. 使用方法
2. 进入目录，修改hooks.c文件，调整自己想要的钩子处理函数；
3. make
4. 使用sudo ./install.sh安装文件系统；使用sudo ./ramdisk.sh挂载
5. 现在hooks.c中的函数已经会被执行，如果是其中有内核态输出，请使用sudo ./viewlog查看输出
6. 具体实现
7. myfs.h中加入钩子的定义，定义了钩子函数的声明、结构体，注册函数的声明

（22行）（43~48行）

1. hooks.c上半部分加入了钩子函数的结构体的实现、初始化，注册函数的实现；下半部分则实现了一个简单地统计inode数目的工具，利用了之前定义的一些工具函数

（hooks.c所有）

1. operations.c中加入了钩子入口。目前只加入了两种（创建inode和删除inode）。

（operations.c 265行 与189~194行）

1. 感想与展望

内核实现理解加深，内核调试的不易

本实验要求有各类文件内函数指针的保存，跨文件调用等等，加深了我对于C语言与内核对于注册函数实现的理解。内核态最方便的还是通过输出调试，很难能够精确定位问题所在，也算是有了一种体会吧。

Linux内核模块的小缺陷

本实验本意是实现一个能够让其他模块来注册钩子函数的模块。但是实践中发现，两个模块间不能很好地访问各自符号，即使利用了EXPORT\_SYMBOL与头文件，必须共享Module.symvers才可以。这导致了我们只能退而求其次，在一个模块中实现钩子函数与文件系统。

能实现更多功能

理论上我们可以在每个操作处实现一个钩子入口，借此实现更多更复杂的功能，例如通过阻止inode创建达到隐藏文件的目的。但是由于时间不足，未能实现，也算是一大遗憾。