epc内容整理

# ko介绍

## ko信息查看

Linux os很多功能模块都是单独以ko形式加载的，lsmod可以列出系统所有已加载ko模块，modinfo [模块名]，可以看到对应ko信息，如modinfo sunrpc部分信息：

[root@host180 ~]# modinfo sunrpc

filename: /lib/modules/5.10.0-136.12.0.86.4.nos1.x86\_64/kernel/net/sunrpc/sunrpc.ko.xz（ko所在路径）

license: GPL

alias: rpc\_pipefs

alias: fs-rpc\_pipefs

srcversion: 77A0FCF5EAB5B36BAC7B770

depends:

retpoline: Y

intree: Y

name: sunrpc

## ko加载

加载ko：inmod [ko绝对路径]

卸载ko：rmmod [ko绝对路径]

或者使用modprobe指令。（自行学习）

## ko编译（以sunrpc为例）

源码路径：linux-3.10.0-862.el7\net\sunrpc

内核所有函数符号地址查看：cat /proc/kallsyms

因为ko最终是要加载到目标os上的，新增模块不可避免的会调用其它模块或者内核基础函数接口，为了ko加载到目标内核ko能够正确找到其它依赖函数地址，编译此ko的编译机内核版本必须与目标主机内核版本一致（保证函数符号表一致），不然ko加载到目标os时，会因为ko中依赖函数地址与os内对应函数符号表地址不一致导致加载失败。

（扩展：是不是把各个os的符号表收集一下，编译epc时，使用不同os的符号表编译不同os的epc.ko，就不用切不同os了？）

## ko加载&卸载入口&退出函数

类似于用户态程序运行第一个被调用函数必是main函数一样，ko加载和卸载也有必定第一个或者最后一个调用的函数。每个模块都会调用下面两个宏，定义本模块加载执行的第一个函数和退出需要执行的函数，以nfsd为例：

module\_init(init\_nfsd)

module\_exit(exit\_nfsd)

# 文件系统注册

## 内核函数接口：

int register\_filesystem(struct file\_system\_type \* fs);

## ceph举例：

static int \_\_init init\_ceph(void)

{

int ret = init\_caches();

if (ret)

goto out;

ceph\_flock\_init();

ceph\_xattr\_init();

ret = register\_filesystem(&ceph\_fs\_type);

if (ret)

goto out\_xattr;

pr\_info("loaded (mds proto %d)\n", CEPH\_MDSC\_PROTOCOL);

return 0;

out\_xattr:

ceph\_xattr\_exit();

destroy\_caches();

out:

return ret;

}

static struct file\_system\_type ceph\_fs\_type = {

.owner = THIS\_MODULE,

.name = "ceph",

.mount = ceph\_mount,

.kill\_sb = ceph\_kill\_sb,

.fs\_flags = FS\_RENAME\_DOES\_D\_MOVE,

};

## vfs及文件系统注册说明

Linux抽象出一层通用文件系统层vfs，vfs定义了所有文件操作语义和接口函数（简单这样理解），如果需要实现自己的文件系统，只需要定义自己的文件系统类型，并实现vfs层定义的相关接口函数，并向vfs注册就行了。

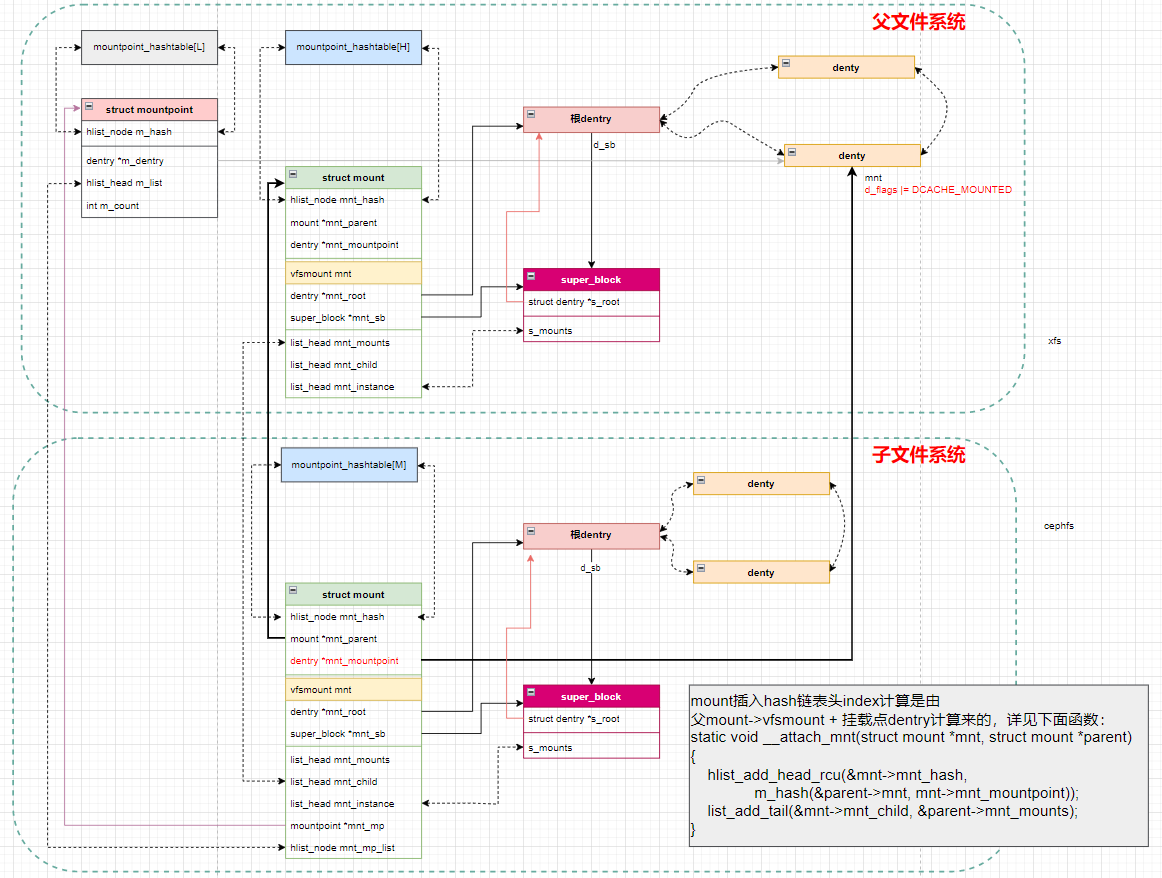
比如执行 mount -t ceph [ip]:[目标目录] [本地目录]

vfs层会根据 -t ceph的字符串参数去查找注册的ceph文件系统，并调用其mount接口函数实现mount操作，mount后就可以像访问本地目录一样访问ceph文件系统了。

# mount

简单说下nfs，ceph，epc文件系统挂载到本地目录后，如何能够通过访问本地目录最终访问到目标目录的。

ls cd一个目录，vfs层是对每层目录dentry，inode的构建，如果dentry有DCACHE\_MOUNTED标志位设置，表明有其它文件系统目录挂载到本目录上，vfs会继续查找，一直查找到最后挂载的文件系统为最终结果并返回。



# epc文件系统介绍

请结合上述内容理解epc ko，文件系统向vfs注册，mount本地目录。

## epc免copy内存

## epc本地socket

## epc op处理