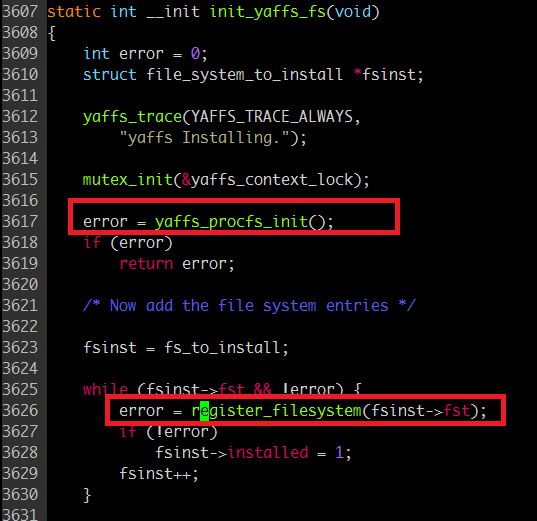
# Yaffs挂载 从vfs到yaffs

1. Vfs是linux中的虚拟层，提供一个统一的接口来供不同的文件系统适配．Vfs主要包括四个重要的数据结构inode,file,dentry,super-block.挂载的主要操作是实现super-block对接.yaffs文件系统不存在物理上的super-block.
2. Yaffs\_vfs.c中有yaffs文件系统挂载的入口函数init\_yaffs\_fs. 内核在启动的时候会加载该函数,该函数主要实现创建/proc/yaffs节点和注册yaffs到系统中的作用. 其中register\_filesystem仅仅是把”yaffs”这个字符串添加到全局变量file\_systems的末尾.真正的挂载还需要应用程序来手动触发才能进行.



1. 当应用程序中输入”mount yaffs /dev/mtdblock0 /system”时,这条命令会进入SYSCALL\_DEFINE5(mount, xxx)这个函数中.这里是Linux系统调用的入口.

SYSCALL\_DEFINE5(mount, xxx)的宏定义展开之后是sys\_mount.

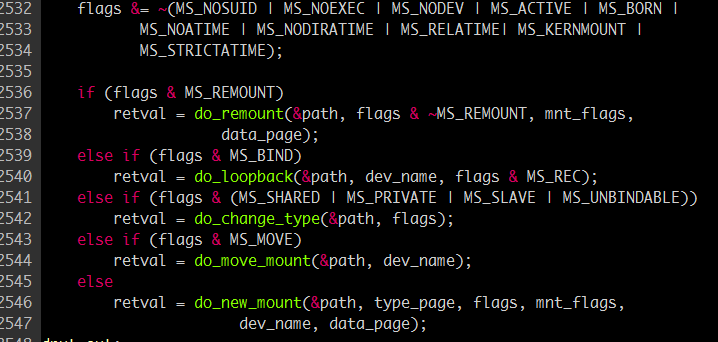
SYSCALL\_DEFINE5(mount, char \_\_user \*, dev\_name, char \_\_user \*, dir\_name,

char \_\_uer \*, type, unsigned long, flags, void \_\_user \*, data)

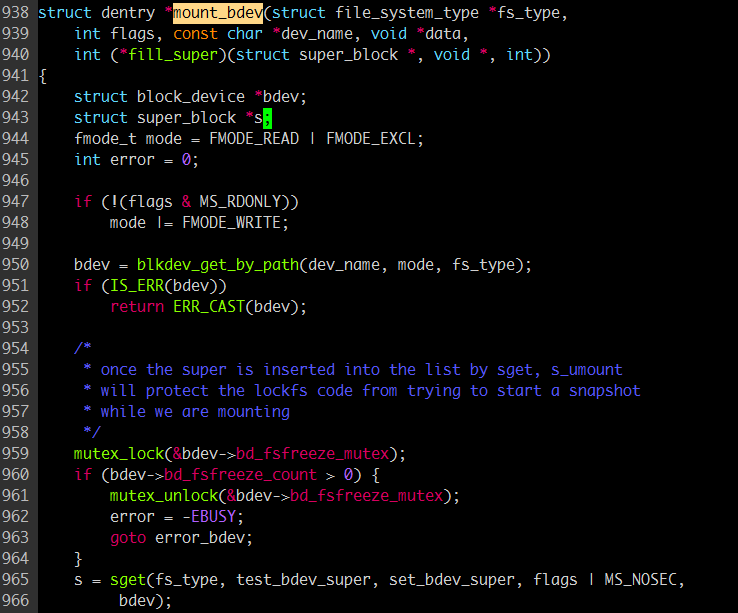
该函数的参数分别代表设备名,挂载的路径,文件系统的类型,挂载的标志,挂载的参数.sys\_mount中会先把这些用户空间的参数拷贝到内核空间,然后调用do\_mount进行挂载.



1. 在do\_mount函数中, 内核会根据flags来选择不同类型的挂载函数,有remount,loopback(挂载镜像文件时一般会用到),修改挂载点.这里选用do\_new\_mount来进行挂载.



1. Do\_new\_mount中,根据文件系统的类型在系统中去查找当前系统是否支持该文件系统.其实就是在file\_systems变量中查找是否存在”yaffs”这个字符串.然后调用vfs\_kern\_mount来继续挂载.该函数会返回一个struct mount结构体指针,挂载成功后,内核会把该结构体添加到内核的挂载树上.
2. Vfs\_kern\_mount函数中会创建struct mnt结构体和struct dentry结构体,dentry就是挂点的目录节点.然后调用mount\_fs继续挂载. Mount\_fs中将调用yaffs的回掉函数yaffs2\_mount来完成挂载.
3. Yaffs文件系统本身并不存在物理意义上的suber block,所以Yaffs2\_mount中调用mount\_bdev先创建一个super-block数据结构,该super-block最终需要调用fill\_super来填充super-block结构体中的各个元素.fill\_super函数的实现是调用yaffs2\_internal\_read\_super\_mtd来实现的,该函数又调用yaffs\_internal\_read\_super这个函数来完成super-block的填充．



1. 在yaffs\_internal\_read\_super函数中,填充了super-block的s\_magic, s\_op(super-block操作函数),s\_flags, s\_blocksize(page cache的大小), s\_blocksize\_bits,s\_fs\_info = yaffs\_dev(yaffs\_dev->driver = mtd), 同时填充了yaffs\_params数据结构体.也会调用yaffs-guts-initialise来填充yaffs\_dev.在guts-initialize中,is\_mounted变量被置1,标志文件系统挂载结束.

# Yaffs文件的读写

文件系统挂载好了,现在可以开始读写了, 文件的读写函数在file\_operations这个结构体中.

