**操作系统实习 lab4 报告**

徐晟 1500012780

本次lab需要修改一个文件系统，建立本地文件系统和服务器的一个连接，让本地文件系统起到一个类似缓存的作用，以获取更大的使用空间。

**Part A**：初始化和挂载一个SD卡

为了使用Ext2文件系统，我们需要在一个SD卡中初始化文件系统，并将其挂载到系统。这一步主要就是按照writeup中的描述执行。主要步骤如下：

1. 修改内核配置选项。用 make menuconfig 命令调出内核配置工具。除了writeup中提到的second extended fs support和Ext2 extended attributes之外，还需要选中Ext2 Security Labels选项。用新的选项重新编译内核。

2. 用mksdcard工具生成SD卡镜像。

3. 用busybox工具格式化SD卡。

这样SD卡的初始化工作就已经完成，只需每次开启虚拟机之后将其挂载到/mnt/sdcard目录下即可。

**Part B**：服务器

为了方便起见，在修改文件系统之前实现了服务器，socket的内容参考了博客：<http://blog.csdn.net/fly_yr/article/details/50387065>。由于服务器中用到了线程相关的函数，编译时应加上-pthread选项。

服务器的主要步骤为：

1. 打开一个监听套接字，在8888端口监听客户端请求。

2. 接收到一个请求后，则创建一个线程处理请求，主线程继续监听客户端请求。

3. 处理请求的线程根据请求内容的不同相应地处理。

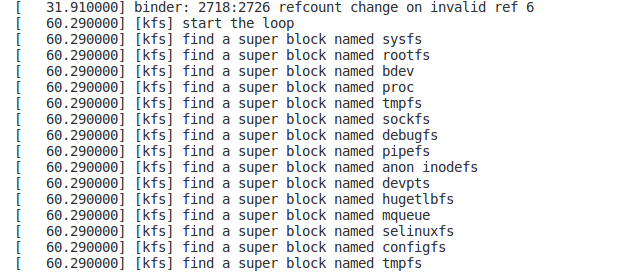
**Part C**：系统自启动线程

首先要在linux/fs.h的super\_operations中添加一个新的操作，需要定义一个函数指针。

int (\*evict\_fs)(struct super\_block \*);

然后需要在fs/evictd.c中实现在内核初始化的时候创建一个叫kfs\_evictd的内核监控线程。每隔一分钟扫描一次挂载的文件系统并调用evict\_fs函数。线程注册函数kthread\_run调用完之后需要用IS\_ERR宏检查是否出错。

结果如下。可以看到每过一分钟会遍历一次文件系统。



**Part D**：挂载文件系统

本部分需要我们给mount命令的-o选项加上四个选项。

首先需要在ext2.h的ext2\_sb\_info结构体中添加四个字段用来保存该文件系统被mount挂载时这些参数的取值。

    uint32\_t s\_srv;

    unsigned short s\_wh;

    unsigned short s\_wl;

    unsigned short s\_evict;

然后在super.c中应当在ext2\_sops结构体中添加一个evict\_fs字段并赋值为ext2\_evict\_fs，但是由于这个函数要到下一个part才实现，暂时将其注释了。

由于需要多加四个选项，所以在枚举中需要添加四个枚举值，然后在match\_table\_t中相应地添加四组，由于这四个选项的值分别是字符串和无符号数，所以需要用到格式串：

    {Opt\_srv, "srv=%s"},

    {Opt\_wh, "wh=%u"},

    {Opt\_wl, "wl=%u"},

    {Opt\_evict, "evict=%u"},

然后在parse\_options函数的switch语句中相应地加入这些case。在这里稍微做了一些合法性检查，但是由于不能确定-o输入时wh、wl和evict的顺序，所以对这三个参数大小的合法性检查不能在这里实现。

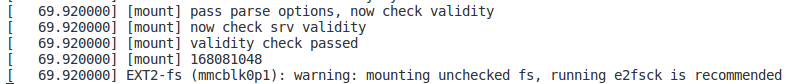
ext2在挂载时会调用ext2\_mount函数，而这个函数会调用ext2\_fill\_super函数，而这个函数则会调用parse\_options函数。在调用parse\_options函数之前做初始化，调用完返回后检查参数合法性。

测试结果：

没有加-o srv参数



加了-o srv参数：



Part E：