

综合实训报告

**姓名：黄思聪**

**班级：计算机科学与技术1602**

**学号：2016317200203**

中国·武汉

2019 年 07 月 10

1. **实训目的**

系统级软件的理解和开发是计算机专业学生应具备的重要能力之一，操作系统在系统软件中具有核心地位。文件系统是操作系统的重要组成部分，是操作系统组织和管理资源必不可少的内容。对操作系统的直观认识就是从使用文件系统开始的，在现代操作系统中，文件系统已是操作系统内核中进程管理，内存管理，设备管理等有关实现赖以管理的基础，进而网络，数据库，各种应用等等也以文件系统为前提。因此，对文件系统的理解关乎对整个计算机体系的理解。本综合实训在内存中仿真Ext2文件系统，包括文件系统的外存格式，内存存在的实现，并以此为基础对文件进行操作和管理。以此理解文件系统的实现原理和文件系统的开发实践，为系统级软件的理解和开发奠定坚实的基础。

要求

（1）要求在内存中仿真Ext2文件系统格式，实现其各个部分的数据结构，理解每个数据结构的内容。

（2）要求实现Ext2文件系统的内存实现和操作。

（3）要求对文件进行打开，读，写等有关操作。

1. 系统原理

系统实现了文件系统以及在文件系统之上的封装的外壳函数，把他暂且叫做shell，shell封装了对文件系统的一些操作——mkdir（创建目录文件），cd(切换目录), ls（列出当前目录的所有文件），touch（创建普通文件），ln（创建符号连接），rm（删除普通文件和符号连接文件），rmdir（删除目录文件），read（读文件），write（写文件）。实现的文件类型与三种——普通文件，符号链接文件，目录文件。系统是完全类似于linux下的ext2文件系统，美中不足的是在本系统中用普通文件代替硬盘进行ext2文件系统的初始化和安装。本来是块设备驱动的ll\_rw\_blk.c源程序被替换成了用c语言库函数fread(),fwrite()编写的rwblock.c程序（因为运行在单进程的环境下也没有了请求数组队列和块设备表项），但是基本原理方式不变。传参设备号以及block号每次对1024B的数据块进行整块读写。但是少了块号到扇区的转换。

1. 文件系统的初始化工作

文件系统初始化工作由init.c文件完成，包括引导块初始化，超级块初始化，i节点位图初始化，数据块位图初始化，磁盘i节点表初始化。其中引导块1KB，超级块1KB，i节点位图2KB，数据块位图2KB，磁盘i节点500KB，之后全是数据块的部分。有一个重要原因将一块分为1024B就是和内存buffer保持一致。在文件系统初始化后挂载根创建文件系统，在挂载根文件系统的时候初始化了系统文件表，然后读取磁盘超级块的1KB内容到Buffer中，判断文件系统魔术是否支持该文件系统，之后初始化内存的i节点位图和数据块位图，从磁盘读出到buffer中，用s\_zmap和s\_imap指向bufferhead。然后把数据块位图和i节点位图第一位复位。然后读文件系统根i节点，并将root和pwd赋值为根i节点。其中读根I节点用iget()操作，先是在i节点表中找一个空项，再在i节点表找和次i节点设备号和i节点号相同的数据项，找到了就释放之前的根i节点，没找到的话从磁盘读出来，放到找到的空数据项中。还有一个很重要的就是buffer的申请，在堆中malloc了3MB大小的内存，然后取得头指针和尾指针，头指针从左到右建立buffer\_head，尾指针从右到左建立bh->b\_data也就是1KB，然后建立空闲链表和hash表管理buffer\_head，hash数组100项，使用当前缓冲去的dev和block相亦或得到。以后在找buffer的时候可以通过散列表加快查询速度。

1. 系统重要的数据结构

在系统中主要维护了几个重要的数据结构包括用户打开文件表，系统文件表，内存i节点表，buffuer缓冲区管理，超级块表，还有一个简化版的进程控制块task\_struct，因为本系统进程的相关操作不需要，这个结构体就三项内容pwd, root, filp。分别代表当前工作目录i节点， 根i节点，用户打开文件表。

1. 8个shell命令的具体实现即内部原理

Mkdir：利用系统调用sys\_mkdir(其实并不会从用户态到内核态），传入两个参数文件名和文件的mode（此处确定文件的目录文件标志位），通过dir\_nami根据第一个参数取得顶层目录文件的i节点并解析该目录文件名称，之后new\_block()申请i节点，置i节点大小为两个目录项，置为脏位，并通过查询数据块位图给inode->i\_zone[0]申请空闲数据块并读入到buffer中，然后设置两个目录文件.和..，其中.的i节点号为自身，..的i节点号对应顶层目录的i节点号。设置本目录i节点的i\_mode，置为脏位，调用add\_entry()加到顶层目录的i节点中，在添加的时候如果发现一块直接索引装不下就再申请一个直接索引装目录项，同时0~6是直接索引，7是间接索引，8是二次间接索引。然后置de的名称和inode节点号。最后将顶层目录的链接数加1。Iput顶层目录i节点和本i节点。

Ls: ls实现起来相对比价简单，因为自己没有实现相应的选项参数，因为在进程控制块存在pwd指向当前工作目录的i节点，也就知道了所有信息，然后判断Mode位得到文件的类型读写权限等，最后输出

Cd：cd和ls是相当于配合起来使用，cd是根据路径参数取顶层目录i节点将其复值给进程控制块的pwd。

Touch: 这个命令是创建普通文件，在调用hsc\_create，第二个参数传mode，包括指明文件类型。然后会调用hsc\_open函数，第二个参数传flag，这里我们给创建和截零标志。然后在用户打开文件表和系统打开文件表找空项，并把引用计数加1。将用户打开文件表指向系统打开文件表项。然后调用Open\_namei，这个函数是最内部的实现。如果文件截零且只读，那么改为只写。找文件顶层目录，并处理得到当前文件名，然后通过顶层目录文件找文件，如果找不到且flag创建标志置位，表明要创建文件，然后通过i节点位图申请i节点号，调用add\_entry()加入到顶层目录文件，然后把目录项的i节点号和相应的名称复值。

Ln:创建符号链接文件，调用hsc\_creat()，第二个参数mode置符号链接标志位，然后调用hsc\_open()，其它操作跟创建普通文件操作一样，最后需要在文件中写入连接文件的路径。

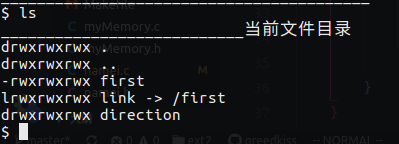
Rm: 删除文件，调用unlink()，流程就是我们考试的题目，先找到顶层目录文件，然后通过顶层目录文件找当前目录项，因为是单进程没有设置文件受限删除和文件受限制访问。如果检查到是目录文件则报错为释放目录文件，如果文件链接计数已经为零则报错，然后将目录项结构的inode置为0，目录项缓冲置脏位。

Rmdir: 删除目录文件，首先调用d\_rmdir()，函数中先找到顶层目录文件，然后根据顶层目录文件找到相应的目录项，取得其i节点，先判断是否是目录文件如果不是目录文件报错返回，再判断目录是否为空（除了.和..是否还有其他文件），如果不为空报错返回，最后置顶层目录i节点指向的i节点号为空，置目录文件的链接数为0,顶层目录的链接数减1，最后iput(inode)和iput(dir);

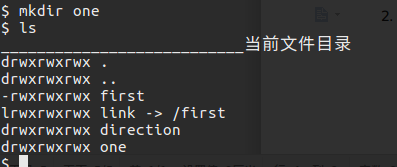
write: 封装的外壳shell要求文件读的时候加偏移值，也就是我们会调用lseek函数改变文件内的偏移指针。最开始会调用hsc\_open（）函数返回文件描述符fd，第二个参数flag参数标志，首先会在用户打开文件表和系统打开文件表找空项，找到后使用户打开文件表项指向系统打开文件，然后找顶层目录，再通过顶层目录找到当前文件I节点返回，并是系统打开文件项指向inode表项。在Open函数中会调用follow\_link()函数，这个函数首先会通过文件的mode判断文件是不是符号链接文件，如果是，那么他会将文件的数据区读到buffer中，而该文件的内容就是链接文件路径，将该路径替换原pathname，再次重复之前的动作。最后返回了文件描述符。然后利用文件描述符找到内存i节点，调用file\_write（）进行读操作，写的时候先根据偏移值除一个数据块的大小找到是哪一块，然后调用bmap找到对应的数据块，然后读入到buffer，然后偏移值mod一块的大小得到块内偏移值，比较写入的数据长度与块内剩余长度，如果足够写就不用重新读入新块，否则写完后读入下一块再写，直到全部写完并返回写入的数据值。

Read: 和write实现原理相同，只不过一个是从用户数据区写入到buffer，另一个从buffer读到用户数据去区。

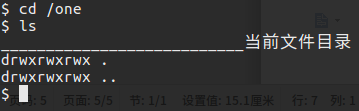
1. 系统测试
2. Ls列出当前目录下的所有文件并表明文件类型



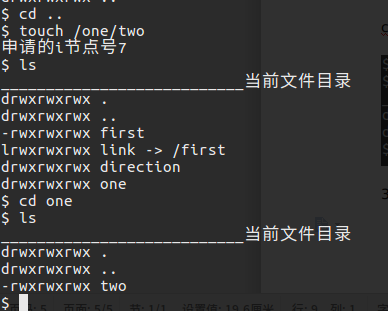
1. mkdir创建目录文件



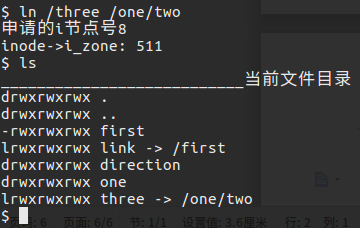
cd进入目录文件



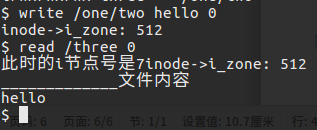
1. 创建普通文件并测试路径解析



1. 创建符号连接文件



1. 在/One/two文件写数据，并从符号连接文件/three中读



1. 改变读的偏移值为1

