

**课程设计报告书**

**基于fuse的文件系统**

**学 院 计算机科学与工程学院**

**专 业 计算机科学与技术**

**学生姓名 陈伟航**

**学生学号 201230600095**

**指导教师 吴一民**

**课程编号 145080**

**课程学分 2**

**起始日期 2015.3.18-2015.6.2**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**基于fuse的文件系统**

# 选题背景

　　FUSE（用户空间文件系统）是这样一个框架：它使得FUSE用户在用户态下编写文件系统成为可能，而不必和内核打交道。

FUSE由三个部分组成：linux内核模块、FUSE库 以及mount 工具。

用户关心的只是FUSE库和mount工具，内核模块仅仅提供kernel的接入口，给了文件系统一个框架，而文件系统本身的主要实现代示位于用户空间中。FUSE库给用户提供了编程的接口，而mount工具则用于挂在用户编写的文件系统。

FUSE起初是为了研究AVFS(A Virtual Filesystem)而设计的，而现在已经成为 SourceForge的一个独立项目，目前适用的平台有Linux, FreeBSD, NetBSD, OpenSolaris和 Mac OS X。

官方的linux kernel版本到 2.6.14 才添加了FUSE模块，因此 2.4 的内核模块下，用户如果要在FUSE中创建一个文件系统，需要先安装一个FUSE内核模块，然后使用 FUSE库和API来创建。

# 二、方案论证(设计理念)

**2.1**用一个二进制文件作为我们的磁盘，通过对这个二进制文件的读写来实现对磁盘的操作。

**2.2** fuse框架可以让我们在自己的文件系统中，输入ls，cd之类的命令时，就调用我们编写的函数来实现对磁盘的操作。

**2.3**通过实现final\_project.pdf中的.getattr,.readdir等函数，实现对磁盘文件的读写，从而将2.1和2.2串联起来

**2.4** 按照final\_project.pdf中的磁盘格式对二进制文件进行初始化，其中bitmap区采用一个bit代表一个block的空闲情况。

**2.5** 根据final\_project.pdf中的要求对文件系统做简化，如：两层目录结构（不需递归寻找路径），chmod 0666（不需对文件编写权限控制逻辑）。

# 三、过程论述

**3.1 安装fuse2.7.0**

tar xvf fuse-2.7.0.tar

cd fuse-2.7.0

./configure –disable-kernel-module

make

sudo make install

在Ubuntu14.04中，执行./configure需要加–disable-kernel-module，因为我之前为了在Ubuntu下读取ntfs分区，安装nfs-3g的时候已经安装了fuse2.8.7，如果覆盖安装fuse2.7.0会导致nfs-3g读取ntfs分区失败。所以fuse2.7.0的安装以及程序的编写调试在虚拟机上完成，而之后发现在fuse2.8.7的环境下编写的程序也可以正常运行。

**3.2 建立一个二进制文件作为磁盘**

dd bs=1K count=5K if=/dev/zero of=diskimg

初始化为全0，大小为5kB\*1k=5MB

**3.3 编写格式化磁盘文件程序format**

每个磁盘块大小为512byte，整个磁盘为5M，即10240个block，

磁盘格式

|  |  |  |
| --- | --- | --- |
| Super block  1 block | Bitmap  1280blocks | Data block  All the rest blocks |

第0块为超级块，需要写入超级块信息

struct sb {

long fs\_size; //size of file system, in blocks

long first\_blk; //first block of root directory

long bitmap; //size of bitmap, in blocks

}

然后写bitmap区：

bitmap中super block区和bitmap区共1281bit,加root所在block为1282个block，所以前1282个bit为1，后面写入0。

Data block：

从第1281块开始写，第1281块写入根目录/的块信息，

struct u\_fs\_disk\_block {

size\_t size; // how many bytes are being used in this block

long nNextBlock; //The next disk block, if needed. This is the next pointer in the linked allocation

list

char data[MAX\_DATA\_IN\_BLOCK];// And all the rest of the space in the block can be used for

actual data storage.

};

具体程序代码见format.c。

**3.4** 编写文件系统ufs，在fuse的接口函数中对diskimg做读写，从而实现diskimg中的磁盘操作。

**3.4.1 获取磁盘文件路径**

将磁盘文件路径作为参数传入

./ufs dir /home/cwh/diskimg

赋值给全局的字符串disk\_path后，从args中移除磁盘路径，再传递给fuse\_main。（ufs.c218-259）

**3.4.2 添加日志**

因为采用-d查看printf时，会有大量fuse的输出信息，所以在diskimg所在目录下建立一个log文件，并将自己的调试信息输出到这里，方便调试。

编写fslog函数来写日志，以可变参数函数的形式封装了fprint函数。（dm.c 15-42）

在运行ufs时加上l参数就可以打开日志功能

./ufs dir /home/cwh/diskimg l

**3.4.3 实现接口函数进行磁盘文件管理（ufs.c dm.c）**

各个函数功能与返回值见final\_project.pdf，此处只描述函数实现思路。

**3.4.3.1 u\_fs\_init(ufs.c 180-197)**

算法描述：

Step1 打开磁盘文件diskimg；

Step2 读取超级块信息

Step3 得到整个磁盘的block数目传给全局变量total\_block\_num

**3.4.3.2 u\_fs\_getattr**

根据路径读取文件或目录大小和类型(dm\_open)传回stat。

dm\_open(dm.c 323-422)

算法描述

Step 1 读取超级块信息，确认根目录的起始块号

Step2 分解路径，得到文件名，拓展名，父目录路径

Step3 读出根目录的块信息到一个u\_fs\_file\_directory中

Step4 如果父目录不是根目录，则顺序读根目录data域，在根目录中找到该目录，读出该目录块信息到一个u\_fs\_file\_directory中

Step5顺序读父目录data域，在父目录中找到该文件（或目录）

**3.4.3.3 u\_fs\_readdir（即dm\_readdir）**

dm\_readdir(dm.c )

算法描述：

Step1 根据路径读取目录块信息

Step2 顺序读取该目录data域，读出目录中的文件名，添加到缓冲区

**3.4.3.4 u\_fs\_mkdir和u\_fs\_mknod（即dm\_create）**

因为我们的文件和目录都是用u\_fs\_file\_directory来处理，所以创建目录和创建文件的思路是类似的，只是flag不同而已。

dm\_create（dm.c 233-312）

算法描述：

Step1 拆分文件路径，得到文件名，拓展名，父目录路径

Step2 读出父目录块信息

Step3遍历目录下的所有文件和目录，如果已存在同名文件或目录，返回

Step4 假如块容量小于目录大小，到Step5，否则到Step7

Step5 在bitmap中找到为0的块号

Step6 为父目录添加一个空的后续块，并写入一个空文件（用flag初始化类型），写回磁盘，结束

Step7 块剩余容量充足，直接扩大块size，到Step6

Note: 在编码测试过程中，发现有些程序（如gedit）修改文件后会自动创建一个后缀为~的备份文件，当文件名或拓展名长度为Max时，备份文件无法正确创建，所以我将fname和fext的最大长度加一，在创建文件时，如果文件名后缀为~，允许创建长度为max+1的文件，否则只允许创建最大长度为max的文件。

**3.4.3.5 u\_fs\_rmdir和u\_fs\_unlink（即dm\_rm）**

dm\_rm：（dm.c 563-599）

Step1 根据路径读取文件（或目录）属性

Step2 检查Flag是否一致

Step3 如果要删除目录，检查目录是否为空，非空则返回错误。（遍历目录下文件，检查flag）

Step3 从文件的起始块开始，清空后续块（从起始块到原文件的结尾块在bitmap中对应的位置为0）

Step4 设置块为未使用，写回磁盘

**3.4.3.6 u\_fs\_write（即dm\_write）**

dm\_write(dm.c 806-912)

Step1 根据路径读取文件属性

Step2 检查要写入的起始位置是否在文件的data域中

Step3 找到起始位置所在的块

Step4 当前块剩余空间小于文件大小，则写满当前块剩余空间，否则只需写入buf中size大小的数据。

Step5 如果数据都已写入，到Step6,如果还有数据未写入，找到尽可能多的连续的空闲块，在磁盘中这个空闲块区的起始位置写入数据。到Step5

Step6 修改文件大小信息，写回磁盘。

3.4.3.7 u\_fs\_read（即dm\_read）

dm\_read(dm.c 423-461)

Step1 根据路径读取文件属性

Step2 假如路径指向一个目录，结束

Step3 从文件的起始块号读出块信息

Step4 检查要读取的起始位置是否在文件内，不是则结束

Step5 从起始位置开始读取块的data域，添加到缓冲区buf中

Step6 返回读取到的数据大小

**3.4.3.8 dm\_search\_free\_blk**

(dm.c 685-754)

Step1 从根目录所在位开始找空闲块

Step2 遍历bitmap中的bit，如果找到的连续空闲块数目多于上次找到的，改变要返回的起始块号为这次的空闲块区的起始块号，否则不改变，假如找到num个连续空闲块，或者已经读到最后一块，到Step4

Step3 没有找到足够的空闲块，跳到刚刚找到的空闲块区的下一个块，到Step2

Step4 返回找到的尽可能长的连续空闲块区的块数，修改参数中空闲块区起始块号为连续空闲块区的起始块号。

**3.4.4** 还有一些小的辅助函数，比如用可变函数简洁地做free操作，逻辑比较简单，在dm.c中已经注释清楚，不再赘述。

**3.5 编写makefile自动化编译**

all:format ufs

format:format.o

gcc format.o -o format

ufs:ufs.o dm.o

gcc ufs.o dm.o -o ufs -Wall -D\_FILE\_OFFSET\_BITS=64 -g -pthread -lfuse -lrt -ldl

ufs.o:global.h ufs.c

gcc -Wall -D\_FILE\_OFFSET\_BITS=64 -g -c -o ufs.o ufs.c

dm.o:dm.h dm.c

gcc -Wall -D\_FILE\_OFFSET\_BITS=64 -g -c -o dm.o dm.c

format.o:global.h format.c

gcc -Wall -D\_FILE\_OFFSET\_BITS=64 -g -c -o format.o format.c

.PHONY : all

clean :

rm -f ufs format ufs.o dm.o format.o

# 四、结果分析

编译：make

格式化磁盘文件（要指定自己建立的diskimg的位置）：

./format /home/cwh/diskimg

显示format done！表示格式化成功。

执行ufs（要指定挂载点，也要指定diskimg位置，可以输入l表示执行日志操作）

测试文件系统命令

ls –al **（测试readdir和getattr）**

mkdir mulu **（测试mkdir）**

echo “new file” >0 （测试mknod和write）

cat 0 **（测试read）**

测试长文件名

echo “long name” > changchang

echo “long name” > chang.aspx

mkdir longdirname

二级目录测试

cd mulu

echo “subdirfile” >0.txt

mkdir subdir

cat 0.txt

echo “testrm” >`1

rm 1 **(测试unlink)**

cd ..

rmdir dir1 **（测试rmdir）**

rm –rf mulu

cd 0

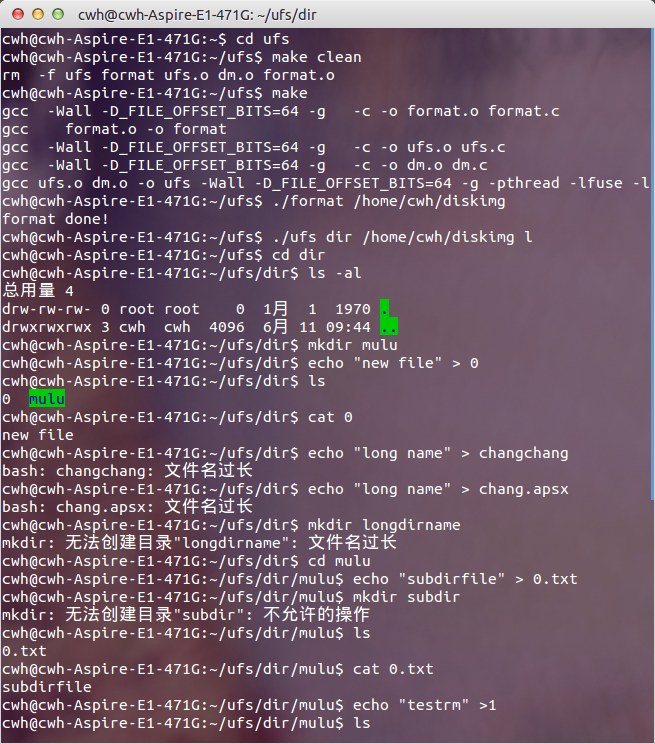
测试写入到目录

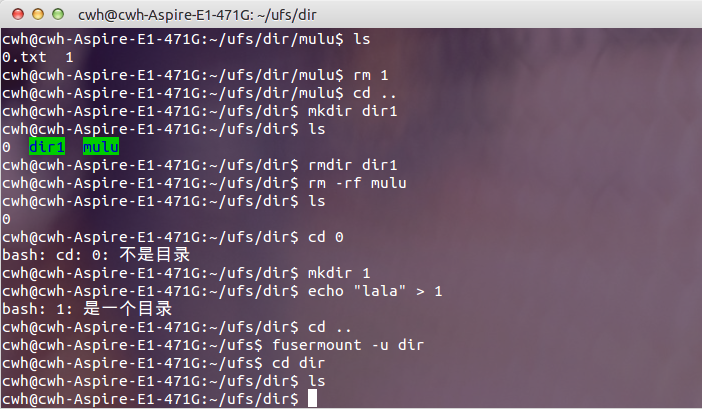
mkdir 1

echo “lala” > 1

解挂载

fusermount –u dir





实验结果与预期相符。

# 五、课程设计总结

大半个学期的课程设计终于完成了。布置课程设计题目时，因为对fuse并没有多少了解，所以听得朦朦胧胧，仔细阅读final\_project.pdf很多遍后才弄清楚课设的要求。通过在网上阅读fuse的相关资料，了解fuse的运行机制，从尝试修改fusexmp开始，一点点地构建起自己的文件系统。

曾经对位图块的操作感到困惑，幸好有老师的点拨，才能把这个工程继续啃下来。惯用Java，用C来开发Linux上的文件系统，着实感到吃力，大部分的时间，都用到了查C语言的一些操作上，但也学到了很多东西，感受到了C的锋利，对编程语言有了更深的体悟。

在这个课程设计里，主要的工作是将Linux文件系统的函数重写为基于我们的磁盘格式的操作，为此下载了Linux源代码，更加清楚了Linux文件系统的组织架构，以前使用操作系统都是知其然而不知其所以然，现在对于操作系统的一些现象，比如磁盘碎片等，更加觉得自然亲切。受益匪浅。

# 六、参考文献

[1]Andrew S.Tanenbaum（著），陈向群，马洪兵（译） 现代操作系统（原书第三版） [M]. 北京：机械工业出版社.2009.7

[2] W. Richard Stevens，Stephen A. Rago 著；戚正伟，张亚英，尤晋元 译 Unix环境高级编程 ［M］．北京：人民邮电出版社． 2014.6

[3] statdm（Linux社区）VirtualBox下设置Windows XP与Ubuntu的共享文件夹［EB/0L］

<http://www.linuxidc.com/Linux/2012-07/65458.htm>

[4] VirtualBox下设置Windows XP与Ubuntu的共享文件夹［EB/0L］

<http://blog.csdn.net/htttw/article/details/6776429>

[5] Ubuntu下使用VirtualBox［EB/0L］

<http://blog.sina.com.cn/s/blog_5c3f32f50101a4yn.html>

[6]使用fuse开发自己的文件系统［EB/0L］

<http://www.ibm.com/developerworks/cn/linux/l-fuse/>

[7] FUSE源码剖析［EB/0L］

<http://tech.uc.cn/?p=1597>

[8] 编译并安装 FUSE［EB/0L］

<http://www.oschina.net/question/12_29050>

[7] Linux makefile 教程 非常详细，且易懂［EB/0L］

<http://blog.csdn.net/liang13664759/article/details/1771246>

[8] #define的部分用法：通过#define连接字符串［EB/0L］

<http://blog.csdn.net/huangjiayun1226/article/details/5754041>

[9] c的位操作［EB/0L］

<http://blog.chinaunix.net/uid-26524139-id-3076894.html>

[10] c文件操作 (转) ［EB/0L］

<http://www.cnblogs.com/duzouzhe/archive/2009/10/24/1589348.html>

[11] struct stat结构体简介［EB/0L］

<http://www.cnblogs.com/CSU-PL/archive/2013/06/06/3120757.html>

[12]lstat［EB/0L］

<http://baike.baidu.com/link?url=ylYEYfDTiL9KghlKzeaEeP4TYw35sU4gre_c_HxVCWqkgl8k70B7eNNfger-rN923FyDmfB-iSJ6iiSj3WnY3_>

[13]strdup［EB/0L］

<http://baike.baidu.com/link?url=o_OjqCwxdmrRCmrPlLaCAoQo_eE4Rav31D9T7N89xXyRV3yl5sSfNY4Q5clj5gxGVxci8sAi5G33iOo-iB_7MK>

[14]strchr［EB/0L］

<http://www.cplusplus.com/reference/cstring/strchr/>

[15]memset［EB/0L］

<http://blog.csdn.net/duke56/article/details/5812726>

[16]memset相关［EB/0L］

<http://bbs.csdn.net/topics/340269536>

[17] 封装printf sprintf fprintf等可变参数函数［EB/0L］

<http://www.cnblogs.com/oceanking/archive/2012/09/12/2682457.html>

[18]Linux 源代码［EB/0L］

<https://www.kernel.org/>

[19] Linux 源代码阅读知识点及要求［EB/0L］

<http://www.cnblogs.com/RbtreeLinux/articles/2241546.html>