**操作系统课程设计实验报告**

——实验四：文件系统实验

负责人姓名：于东方

学号：14061149

日期：2016.5.21

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 于东方 | 14061149 | 新增mkdir指令、绝对路径和多级目录、cf中属性的更正等 |
| 2 | 牛钰浩 | 14061155 | cf向文件中写入内容、整理例会记录、实验报告 |
| 3 | 常涛 | 14061161 | rm递归遍历删除目录、支持扩展簇内容 |
| 4 | 姜铭 | 14061167 | rm递归遍历删除目录、支持扩展簇内容 |

目录

[目录 3](#_Toc11791)

[1.实验目的 5](#_Toc13893)

[2.需求说明 5](#_Toc28038)

[2.1基本要求 5](#_Toc30663)

[2.2 提高要求 6](#_Toc10729)

[2.3 完成情况 6](#_Toc15873)

[3.设计说明 7](#_Toc27308)

[3.1 程序流程图 7](#_Toc12423)

[3.2基本要求实现说明 8](#_Toc10070)

[3.2.1目录列表函数ls 8](#_Toc9715)

[3.2.2改变目录函数cd 8](#_Toc3176)

[3.2.3删除文件函数df 10](#_Toc20438)

[3.2.4创建文件函数cf 13](#_Toc2069)

[3.3 提高要求实现说明 15](#_Toc25326)

[3.3.1新增创建目录命令mkdir 15](#_Toc26134)

[3.3.2新增删除目录命令rm 16](#_Toc1594)

[3.3.3绝对路径和多级目录的支持 19](#_Toc15082)

[3.3.4创建文件命令cf的改进 19](#_Toc26547)

[3.3.5列出目录项命令ls的改进 19](#_Toc8764)

[4.问题及解决方法 21](#_Toc2983)

[4.1 文件系统挂载过程中出现的问题 21](#_Toc12611)

[4.2创建文件的时间错误 23](#_Toc21513)

[4.3 fd\_cf()创建大量文件时的错误 24](#_Toc17751)

[4.4 删除文件夹过程中遇到的问题 26](#_Toc26426)

[5.收获和感想 27](#_Toc22554)

1.实验目的

1. 了解文件管理系统的作用和工作方式，重点是Linux下利用分区表示（如sdb）、格式化分区（mkfs）、装载指令（mount）来构建一个文件系统，使用cd、ls、rm、mkdir等指令来操作文件夹的新建和切换，文件的列出和删除。

2. 了解FAT文件系统的结构，重点是掌握FAT16格式的分区的四个区域及各区域的格式、掌握四个区域内容与具体文件操作的对应关系。

3. 学习文件管理系统的一般开发方法，并通过创建文件（用以模仿U盘）和回环设备（loop，以便能够挂载到系统中，使用shell和资源管理器访问虚拟的U盘）开发一个具有简单功能的FAT16格式文件系统。

2.需求说明

2.1基本要求

准备一个FAT16格式的U盘（此处用虚拟U盘代替），在Linux下编写一个文件系统管理程序，对U盘上的文件进行管理。具体要求如下：

1. 设计并实现一个目录列表函数（无须支持选项，如ls -a、ls -l等），用来显示当前目录下包含的文件信息。

2. 设计并实现一个改变目录函数（无须处理路径名，如../../directoryName等），用来把当前目录切换到上一层目录或当前目录的子目录。

3. 设计并实现一个删除文件函数，使用当前目录中的要删除的文件名作为参数，用来删除指定的文件，要注意文件的隐藏、只读和其他系统属性。

4. 设计并实现一个创建文件函数，使用要创建的文件名和文件大小作为参数，用来创建一个新的文件。

2.2 提高要求

1. 增加创建目录的功能。

2. 增加删除目录的功能：通常需要先判断目录是否为空目录，若目录不为空，则需给出提示，并删除其包含的所有子目录和文件；若是空目录则可以直接删除。

3. 增加绝对路径和多级目录的支持：这里需要对输入的目录路径字符串进行解析，然后逐级查找目录。

4. 对ud\_cf()函数进行改进，使其可以向文件中写入实际内容，并根据写入的内容计算文件的实际大小。

5. 对ud\_ls()函数进行改进，增加对全部非根目录信息的读取（基本要求中仅读取一个扇区的非根目录细信息）。

2.3 完成情况

能够完成本次实验的所有基本要求，并在此基础上增加了以下东西：

1. 增加创建目录的指令mkdir，可以实现在当前目录下创建一个目录，在FAT表中为此目录分配必要的簇。

2. 增加删除目录的指令rm，该命令递归删除目录下的所有待删除内容，并支持参数。若含有参数-r，则直接递归删除所有子目录和文件的内容；否则，先判断目录是否为空，若不空则提示信息。

3. 增加对绝对路径和多级目录的支持，可以在cd、ls等指令中使用绝对路径和多级目录

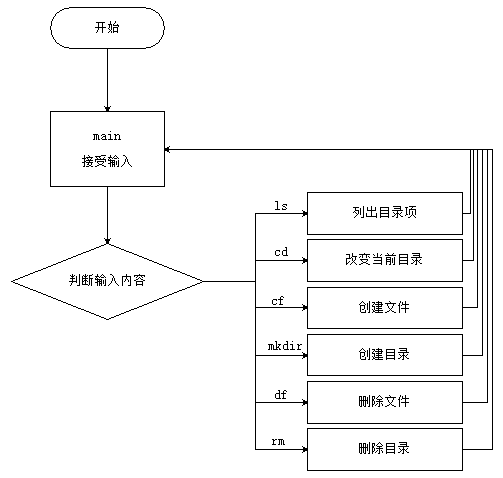
4. 改写创建文件指令cf，使之接收一段输入，向文件内写入真实内容。此处根据内容计算实际大小，分配所需要的簇、记录在FAT表中

5. 改写ls，使之支持全部非根目录信息的读取，并支持读取跨簇的目录项（即文件夹可大于1024项）。

3.设计说明

3.1 程序流程图

程序整体的流程图如下。



程序主体部分是一个大循环，每次要求用户输入一个指令，然后调用相关的指令完成请求的动作。如果输入错误，则会显示所有指令的格式，供用户参考。

该图仅展示了程序的大致轮廓，详细的流程将在后文中详细展示。

3.2基本要求实现说明

基本要求在源程序中已经完成了大部分内容。此处主要是检查完成形况、寻找并修复源程序中存在的一些bug。

3.2.1目录列表函数ls

该命令用来列出当前目录下的所有文件和子目录，包括他们的名字和其他属性。原代码中的ls可以列出根目录的所有目录项，可以列出子目录的一个簇中记录的目录项。该命令的跨簇读取目录项功能详见提高要求实现说明。

3.2.2改变目录函数cd

cd指令用于改变当前目录。在原代码中，该指令仅能够支持跳转一级目录。新设计的代码已经支持多级跳转和绝对路径。

cd跳转的原理是改变当前目录的一个Entry指针curdir，还有父目录的一个堆栈fatherdir[]。对于特殊目录（如根目录“/”，当前目录“.”，父目录“..”），直接修改curdir和fatherdir[]即可。

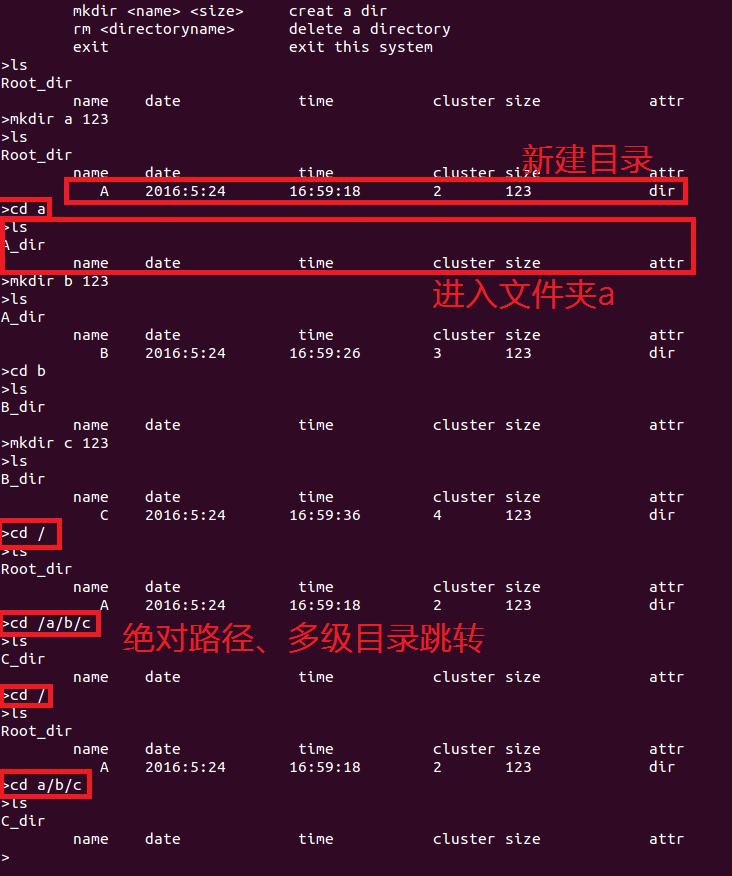


对于一般情况，需要解析字符串，并逐层寻找目标文件夹的目录项，确保所切换的目录确实存在。



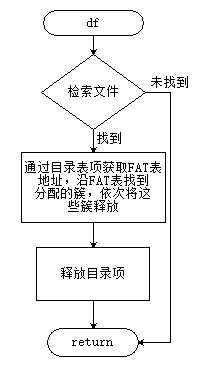


完成后程序的行为如下。可见程序已经可以支持跳转多级目录和绝对路径。



3.2.3删除文件函数df

原程序中已有的df是用来删除一个文件的。删除一个文件，是在当前目录下查找指定的文件，在FAT表中删掉这个文件分配到的所有簇，最后将这个文件的列表进行删除。



然而，原代码中给出的程序中有一个bug：假设当前目录下有A、B、C三个文件，使用df删除B文件，则A文件也会消失。进一步分析得出，代码中有如下部分：

...

if(lseek(fd,ret-0x20,SEEK\_SET)<0)

perror("lseek fd\_df failed");

if(write(fd,&c,1)<0)

perror("write failed");

~~if(lseek(fd,ret-0x40,SEEK\_SET)<0)~~

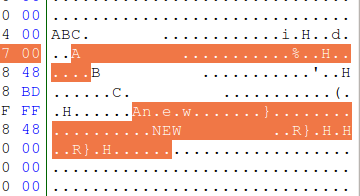
~~perror("lseek fd\_df failed");~~

~~if(write(fd,&c,1)<0)~~

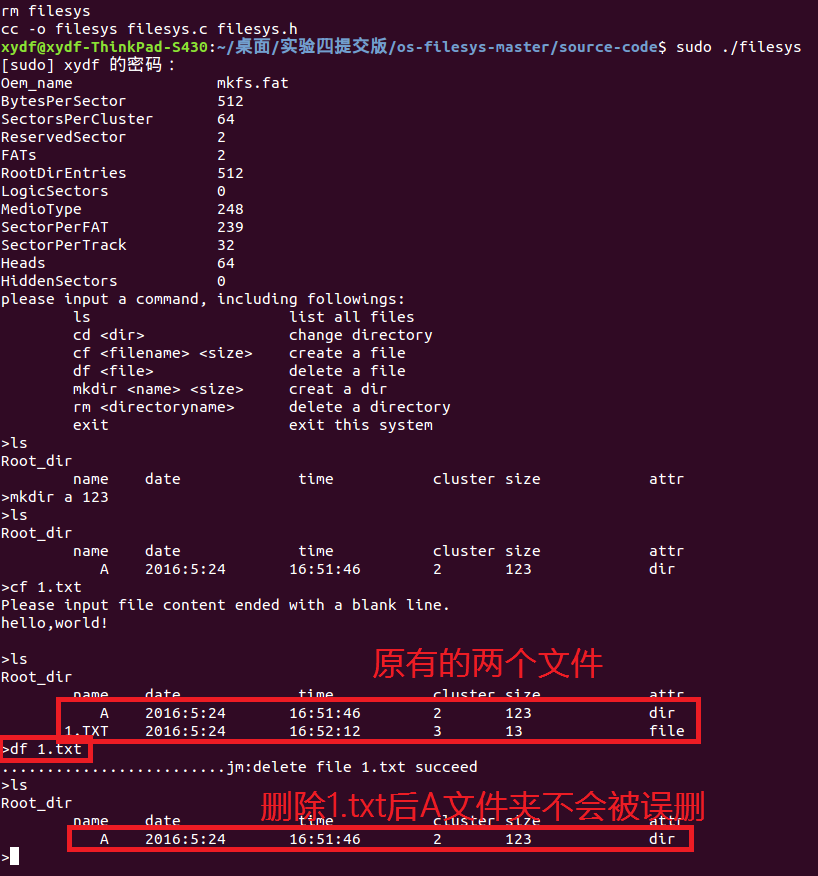
~~perror("write failed");~~

...

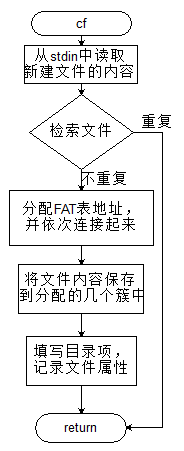
该部分是删除目录项的代码。代码中不仅删除了B文件的32字节的目录项，而且删除了B文件目录项之前的32字节目录项。这一怪异的做法使人联想到长文件名。通过直接对U盘文件的查看发现，自己创建的文件只有一个32位目录项，而挂载后由系统创建的文件会有两个32位目录项。前面的一个存储了长文件名。



看来df中清除目录项直接清除两项是有原因的。但此次实验不要求支持多级目录，故此处直接将标有删除线的代码去掉。



3.2.4创建文件函数cf

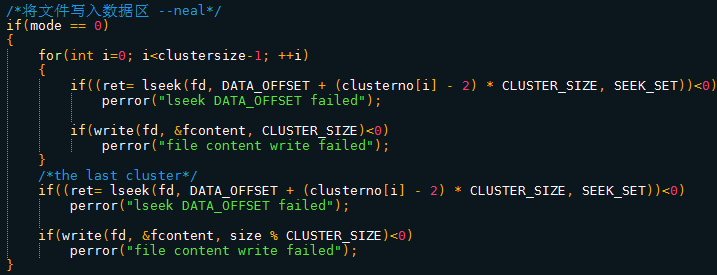


cf命令是创建文件。创建文件的操作与删除文件正好相反：要分配一定数量的簇，并且填写目录项。

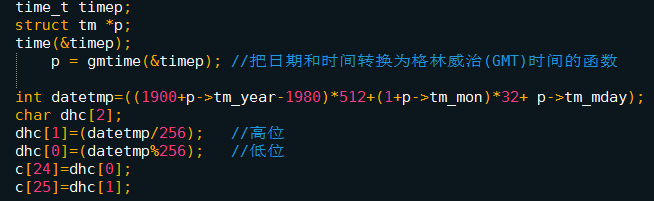
为了能够指定cf的输入，特编写以下getFileContent函数，从标准输入中读取输入的内容，以一个空白行结束，返回输入的字符串：



在fd\_cf函数中，调用以上函数获取文件内容，然后计算文件大小，得到需要分配的簇的个数。分配簇后便可将文件内容写入簇中：



原代码中在写文件属性时时间的格式有问题，在新的代码中已经修改。



修改后进行测试，程序能够成功创建文件，并设置正确的属性信息。



指定的内容会被正确地写入到文件中。具体详见问题及解决方案中挂载相关的部分。

3.3 提高要求实现说明

3.3.1新增创建目录命令mkdir

原来的程序中没有新建目录命令，在新的代码中，我们新增了mkdir指令。该指令的行为与创建文件指令cf类似。实际上，我们也是通过重用cf的代码实现的。

与创建文件不同，这里的mkdir在创建目录时不需要用户指定簇的内容，而且在目录项的属性中将此项标记为目录。



3.3.2新增删除目录命令rm

删除文件夹的rm指令也是新增加的指令，用于删除一个目录及其下的所有文件和目录。rm的实现是采用寻找目录项函数和删除文件函数fd\_df()，实现的重点是递归调用。以下是其实现在代码中的体现。



准备工作



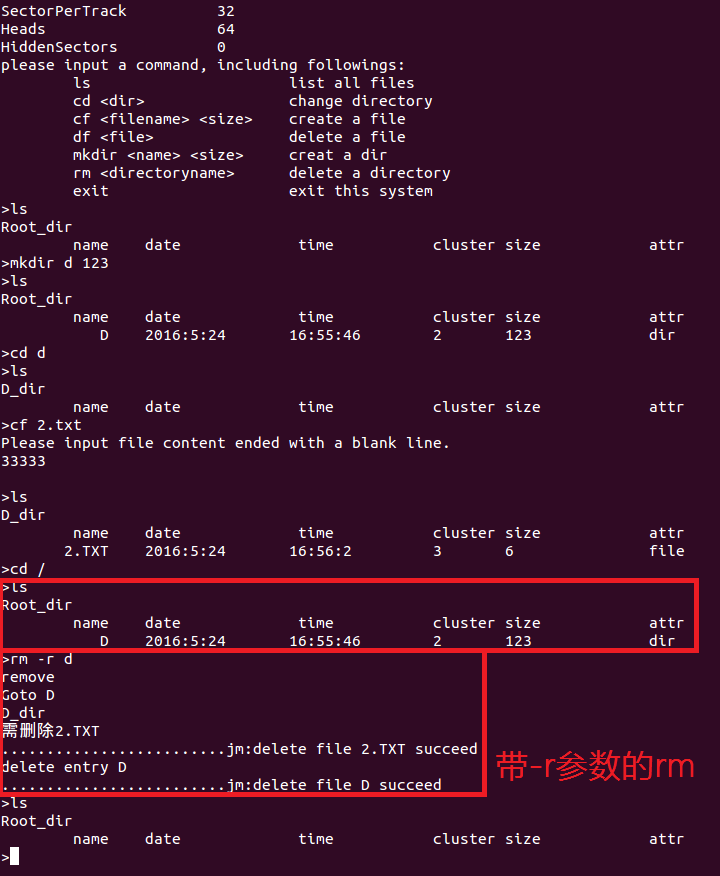
递归删除



收尾工作

rm的具体执行示例如下：





3.3.3绝对路径和多级目录的支持

绝对路径即从“/”开始的路径，多级目录即一次可以跳转不只一层。原代码中的指令不支持多级目录的操作，而新的代码中cd和基于其上的命令可以支持绝对路径与多级目录。

具体实现、结果展示详见3.2.2节。

3.3.4创建文件命令cf的改进

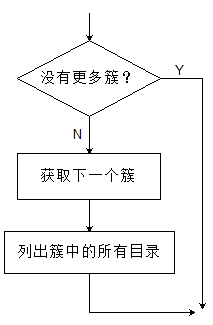
提高要求中要求新的cf可以支持用户输入文件内容，以此决定文件的大小和分配的簇的数量，并将内容写入这些簇中。

cf的流程图、实现与结果展示详见3.2.4节。关于指定的内容是否会被正确地写入到文件中，具体详见“问题及解决方案”中挂载相关的部分。

3.3.5列出目录项命令ls的改进

原代码中的ls可以列出根目录的所有目录项，可以列出子目录的一个簇中记录的目录项。在提高要求中，该命令将支持跨簇读取目录项。

下面的流程图和代码释义图展示了ls被修改的核心部分：如果当前处于非根目录下，则循环访问当前目录的所有簇，从中读出所有目录项。

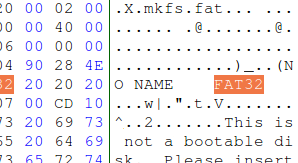




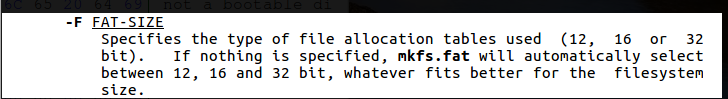
4.问题及解决方法

4.1 文件系统挂载过程中出现的问题

实际实验过程中，按照PPT上指出的操作步骤，mkfs对2G文件data操作生成的文件系统并未FAT16，而是FAT32。

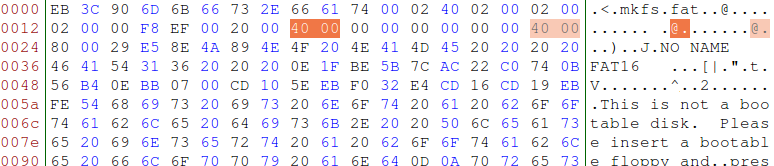


这是系统默认的配置。对于本实验来说，采用FAT16格式才能保证兼容性。所以此处格式化U盘文件时使用-F参数，指定16位FAT表格式。



在使用mount指令装载时，PPT中建议装载到/dev/sdb处。这样做有些不合理：dev是存储设备文件的地方，而不是用来存储挂载点，况且这样做会使得系统无法使用U盘。一个挂载点，应该是/mnt目录下的一个子目录。此实验中为了方便，直接在当前用户home目录中建立了一个挂载点。

挂载后，发现程序的读写与shell的读写互不影响，并且彼此不可见。经仔细查看数据文件内容时，才意识到应该修改特定的常数，如每扇区字节数、每簇扇区数、根目录个数等等。我们在此按照程序的行为修改了数据文件的定义，让系统使用与程序相同的配置。



此时，data可以被正常挂载并且读取filesys创建的文件。

使用filesys创建一个很长的文件。即用一下作为输入重定向给filesys：

cf clustest.txt

TestLine0000

...

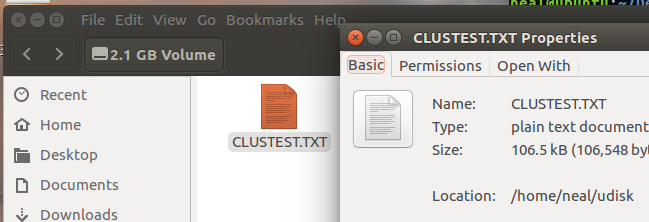
TestLine8193

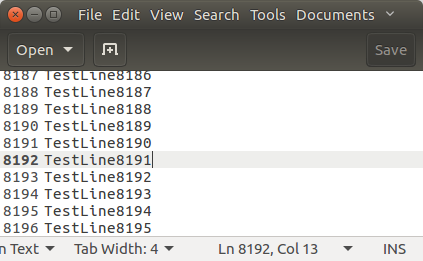
TestLine8194

TestLine8195

exit

这样将会在根目录中创建一个的文件并写入100+KB的内容。操作完成后将data文件挂载到系统中，可以从资源管理器中看到创建的文件，并可以正常读出其内容。

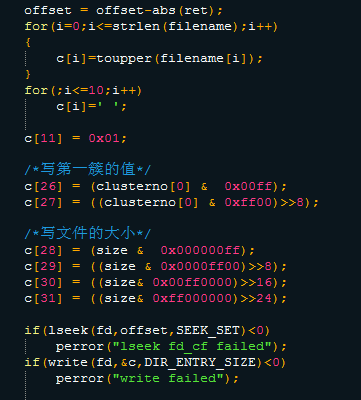




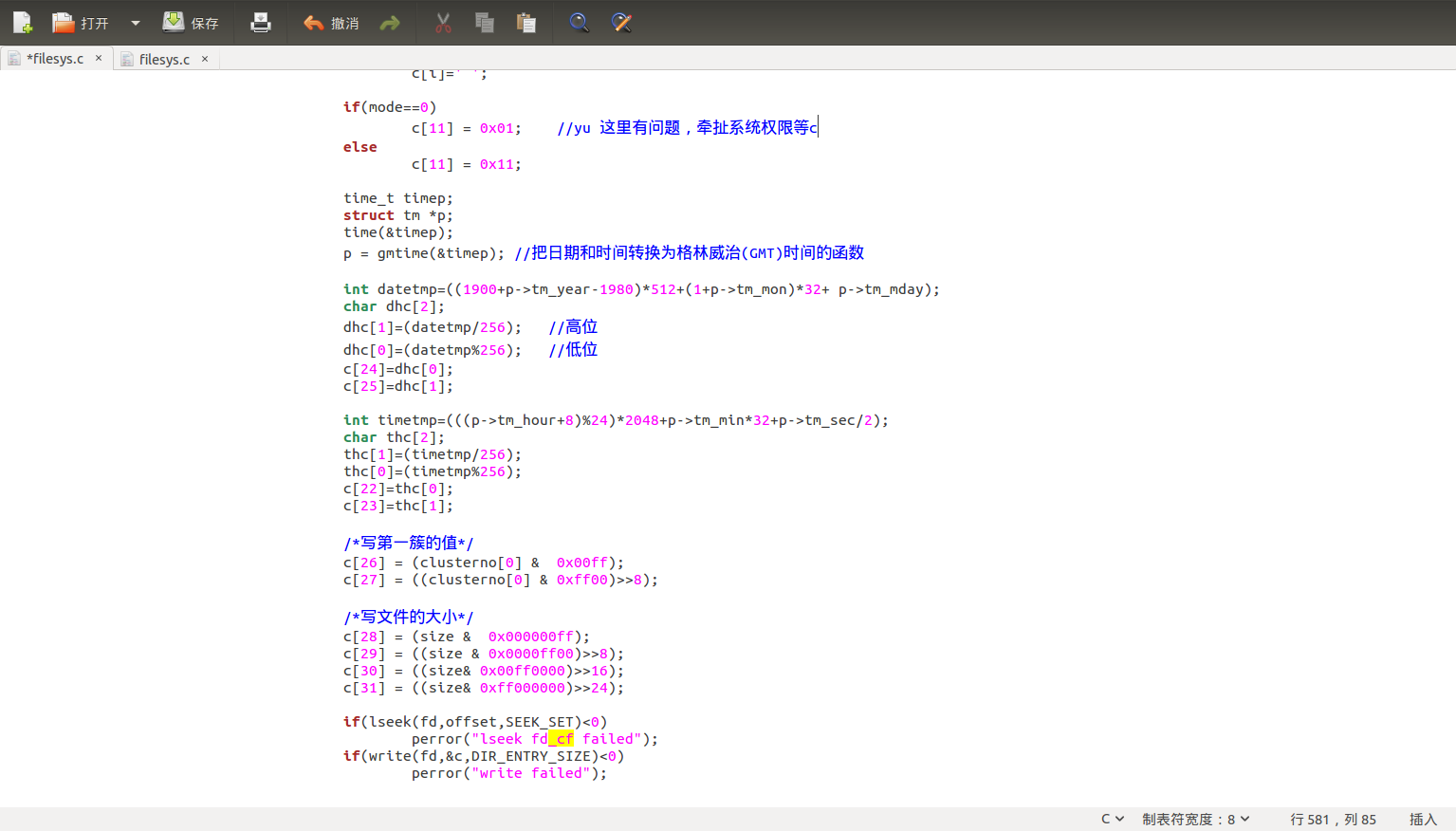
然而，由于代码历史原因，程序沿用了一套不太规范的标准。比如前文提到的长文件名的问题。这也是程序未来的修改方向。同时，这提醒我们，从一开始就布局代码，按照标准规范来处理问题，无疑会对之后的编程带来好处。

4.2创建文件的时间错误

错误定位：fd\_cf()函数



发现并没有关于创建时间部分的代码，于是代码需要添加以下部分：其中time\_t和struct tm均在系统包含的头文件<time.h>中。在修改时间的时候我们也遇到了一个bug，这种方法算出的时间是格林威治时间，与北京时间相差8小时。开始的时候，我们只是简单的将时间（小时部分）+8计算，但是我们晚上检查程序的时候偶然发现文件创建时间为25：32：4，而非1：32：4。问题出在，我们不该仅仅将小时+8，而还应该对24取模。



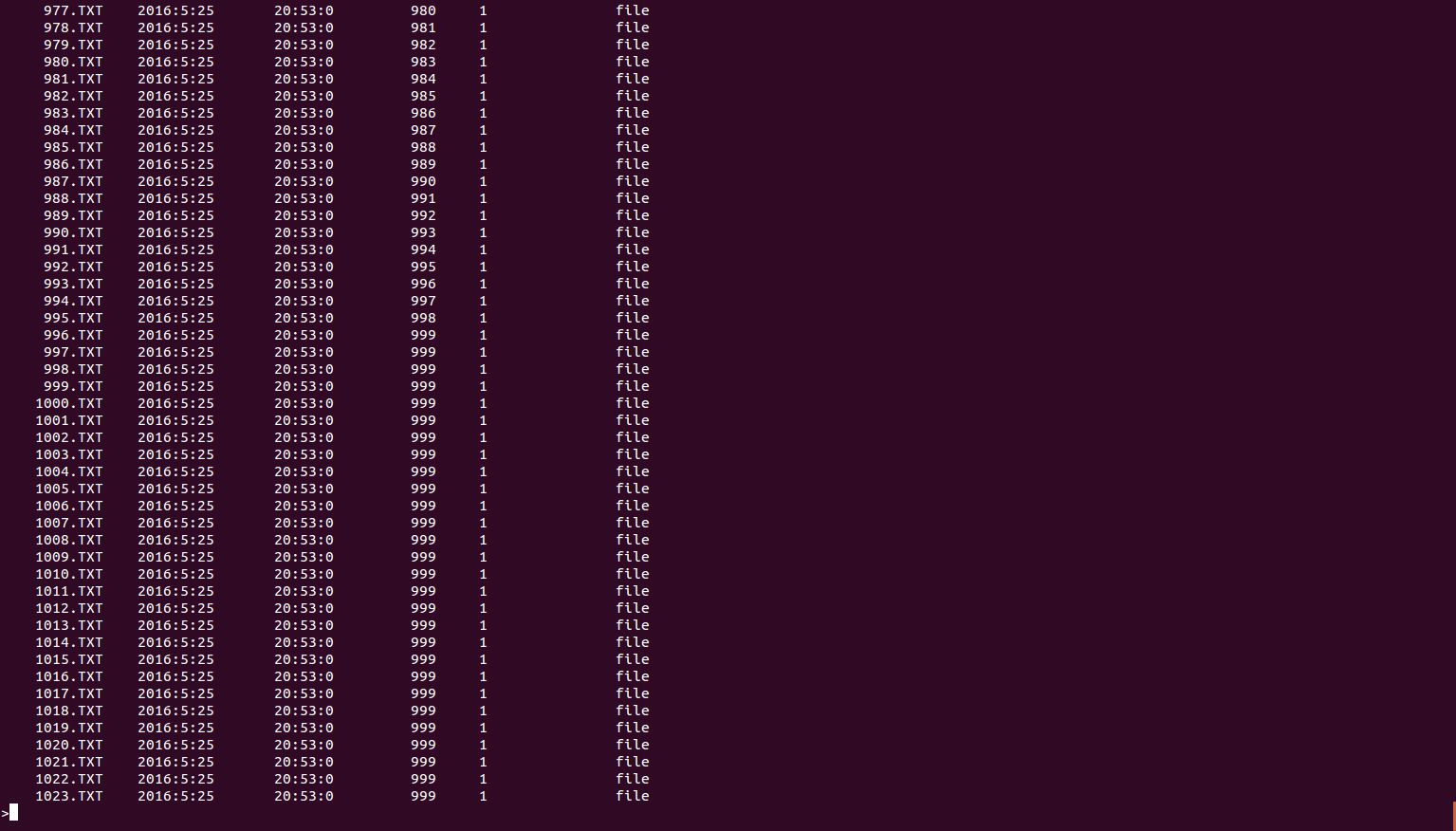
4.3 fd\_cf()创建大量文件时的错误

错误定位：fd\_cf()

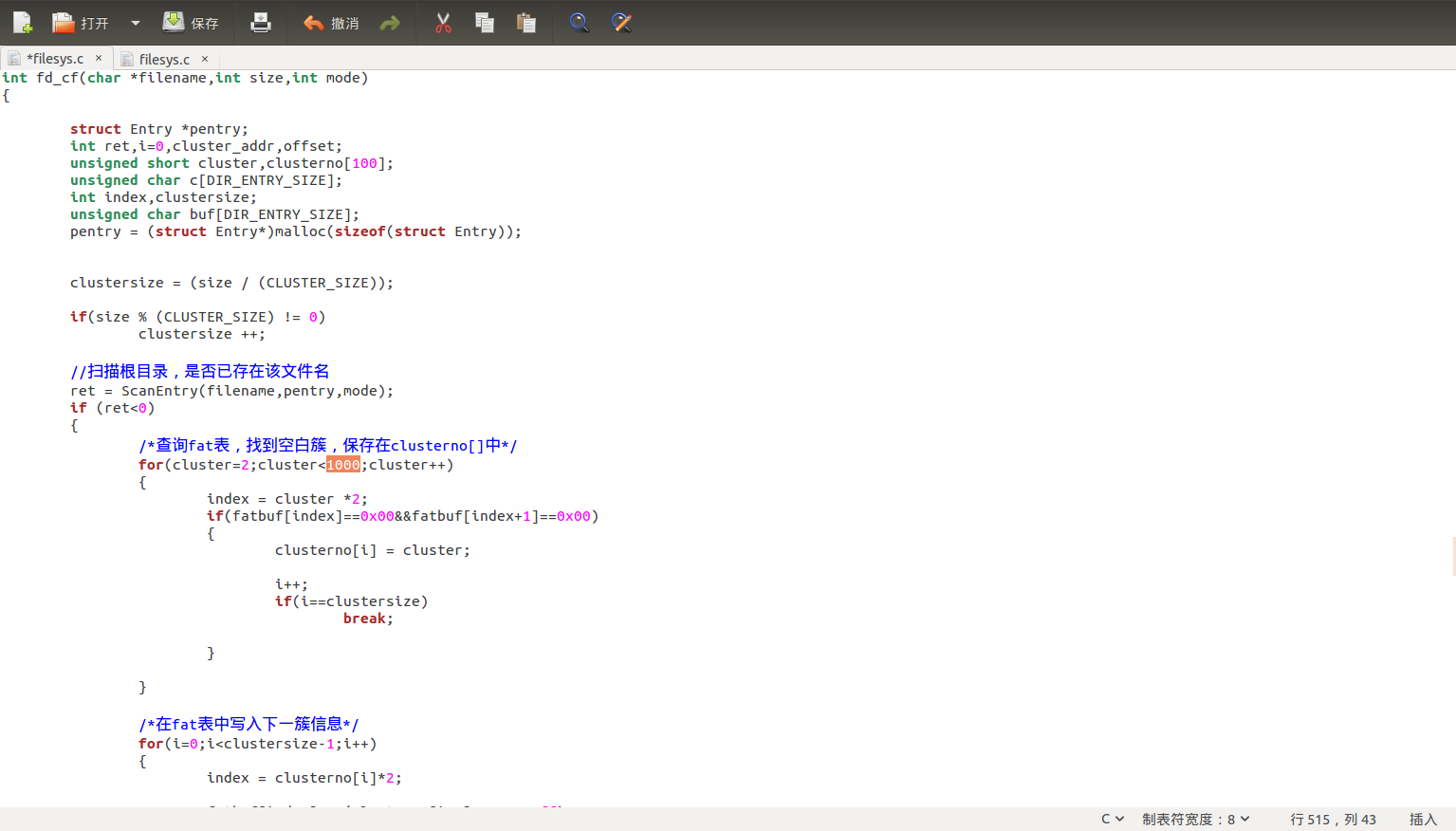
为了检测我们更改后的ls是否可以满足跳簇的要求，我们在文件夹abc下面创建了2035个txt格式文件（用python写一段小程序，然后将程序结果输出到in.txt；在执行sudo ./filesys时候输入重定向）。经过我们检查发现代码如下部分有问题：

1.显示的文件数量最多为1024个（0到1023），显然与跳簇问题有关

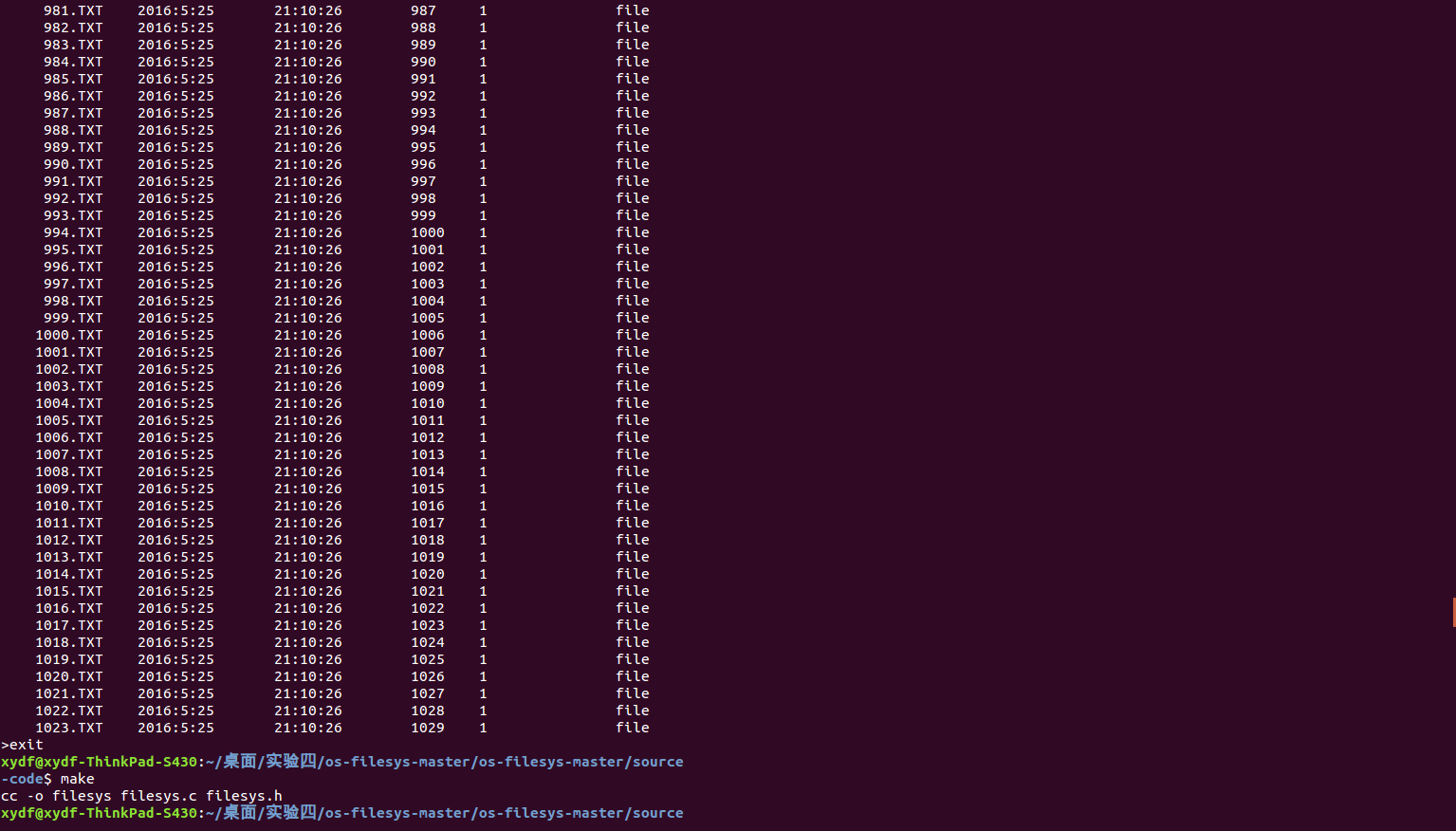
2.后面文件的簇号同为999（不同文件应该占有不同簇），显然与跳簇无关



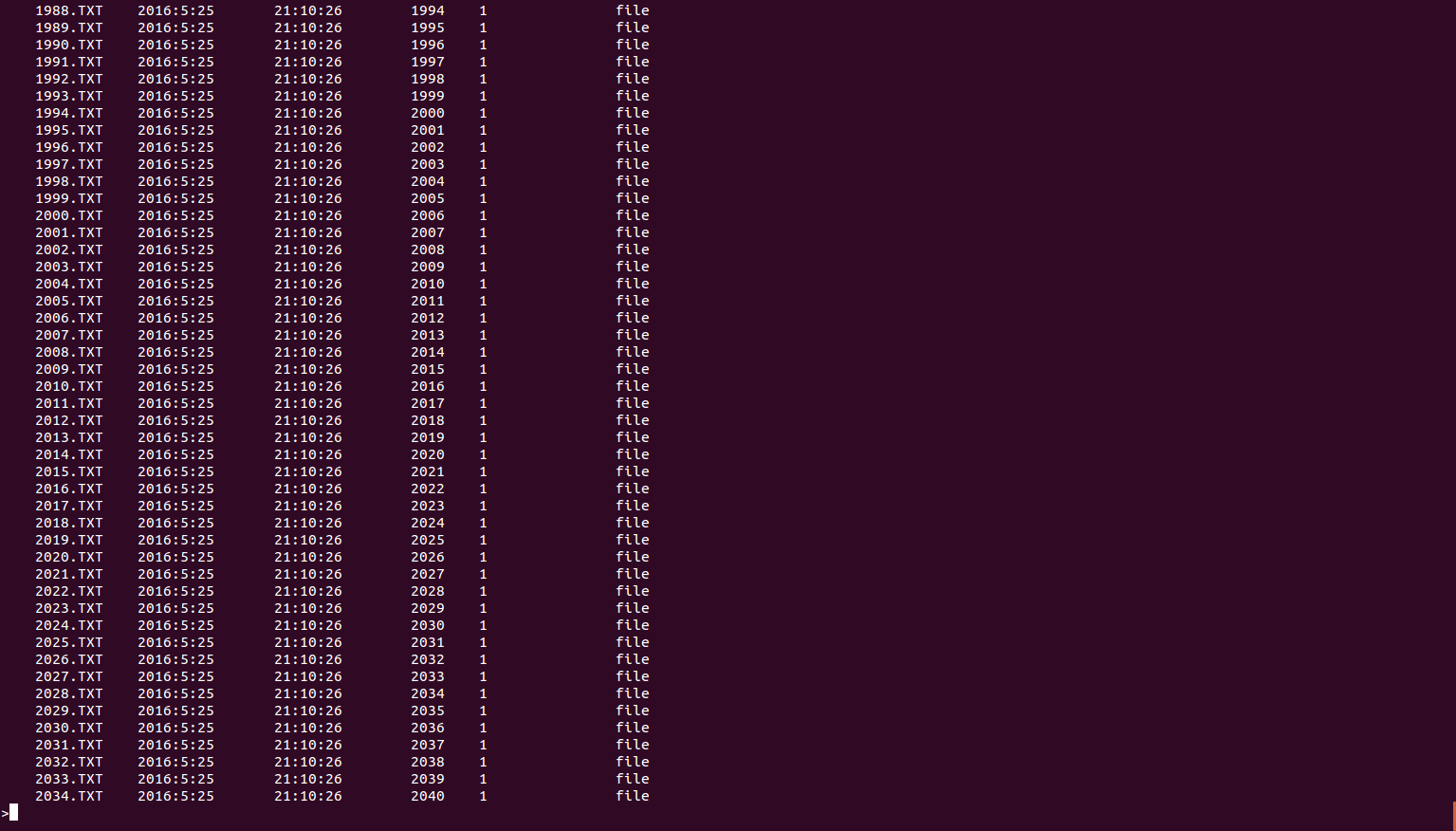
先解决与跳簇无关问题，查询代码发现：



果断将cluster<1000的限制条件更改，之后ls效果如下：



但仍与期望结果相距甚远（跳簇存在问题）。经过我们多次检查，发现ls部分并没有问题，于是我们想到cf函数部分并没有跳簇的设计。在增加cf部分的跳簇代码后，我们终于获得期望的结果：



4.4 删除文件夹过程中遇到的问题

我们采用递归的方式实现删除目录及目录下的文件。先调用函数fd\_cd将当前目录切换到要删除的目录，扫描该目录下的所有目录项，若遇到文件则调用fd\_df函数删除文件，若遇到目录则递归调用，进入子目录。如此循环，直至删除该目录下所有文件和目录。最后切换到该目录的父目录，删除该空目录。

最初在该函数执行的过程中，我们发现实际执行的路线并非如我们想象的那样，运行结果是只能删除一部分文件，而所有的目录都无法删除。通过在程序中添加输出语句跟踪程序递归的路线，我们发现是调用fd\_cd函数时会重新定位文件的读写位置，导致程序最好无法正常跳到当前目录的父目录，也就无法删除目录了。我们采用的解决方法是每次调用fd\_cd函数后都调用lseek函数重新定位到调用fd\_cd之前的位置，这一问题也就解决了。

5.收获和感想

首先，经过这次实验，我们接触到了建立在一个连续的存储模型上的文件系统FAT16，了解了一个文件系统所需要的数据结构，对文件系统的开发有了一个初步的认识。然而，据资料表明，FAT16是相对简单的一个文件系统，早已在储存量大增、新技术层出不穷的今天淘汰。现今Windows使用的NTFS，Linux使用的ext4和VFS，是在相近的基础上建立起来的复杂的文件系统，满足了压缩存储、安全访问等新的要求。这些文件系统的内容，还需我们进一步学习和掌握。同时，也希望之后的OS实践中涉及到这些内容，增进我们的视野。

其次，在学习过程中，我们还了解到Linux中对文件系统的看法。Linux中，设备可以被看作一个文件，一个文件也可以与回环设备关联从而被认为是一个设备。这一观点十分灵活，其设计之巧妙令人惊叹。在今后的学习中，还会接触到更多Linux本身，这些知识的作用也会彰显出来。另外，这也激发着我们的兴趣与灵感，让我们为编程之美更加倾心。

在实际实验过程中遇到了很多困难。此次实验使用的C几乎无法使用调试器帮助调试。这时，程序的设计（包括函数的功能、全局变量、大致流程等）便显得尤为重要。所以本次实验使我们明白，从一开始就要对整个程序拥有一个大致的了解。操作系统是个庞杂的体系，具体的开发需要具有一定的软件工程知识，并使用模块化、规格化思想，使代码尽量重用，使结构趋于合理。

在代码之外，我们也遇到其他的挑战，比如占用大量时间的OO，与OS之间的协调问题。一直在做一件事情会使得学习乏味，适度调换任务会增加工作效率。然而可能在时间规划上存在些不合理的现象，本来可以拿出更多时间，将OS做得更好。

OS的实验现已告一段落，而专业学习的道路仍在继续。我们会总结经验，不断向前。