

操作系统原理课程设计

**2019/2010(1)**



实验题目：模拟UNIX文件系统

学生姓名：高源

学生学号：201706062013

学生班级：软件工程1704

任课教师：李波

完成日期：2018年12月28日星期五

**计算机科学与技术学院**

目录

[一、实验介绍 3](#_Toc30165451)

[二、实验环境 4](#_Toc30165452)

[三、实验设计 5](#_Toc30165453)

[3.1系统流程 5](#_Toc30165454)

[3.2文件结构 6](#_Toc30165455)

[3.3实现命令操作 7](#_Toc30165456)

[四、数据结构 7](#_Toc30165457)

[4.1超级块数据结构 7](#_Toc30165458)

[4.2节点数据结构 8](#_Toc30165459)

[4.3文件数据结构 9](#_Toc30165460)

[4.4目录数据结构 10](#_Toc30165461)

[五、模块详解 10](#_Toc30165462)

[5.1文件操作 10](#_Toc30165463)

[5.2目录操作 13](#_Toc30165464)

[5.3主函数 15](#_Toc30165465)

[5.4核心函数 16](#_Toc30165466)

[六、实验演示 18](#_Toc30165467)

[6.1进入文件系统 18](#_Toc30165468)

[6.2文件相关操作 19](#_Toc30165469)

[6.3目录相关操作 24](#_Toc30165470)

[七、实验总结 28](#_Toc30165471)

# 一、实验介绍

实验要求大家完成一个 UNIX文件系统的子集的模拟实现。实验的提示与要求：

1. 文件卷结构设计

0#块可省略

I节点栈及空间为20项 i节点栈为保存在内存中的i节点数目吗，用栈的形式保存

或者是空闲节点栈

块大小为512字节

卷盘块数大于100

I节点盘块数大于10块

1. I节点结构设计

文件大小

文件联接计数

文件地址

文件拥有者

文件所属组

文件权限及类别

文件最后修改时间

其中文件地址为六项：四个直接块号，一个一次间址，一个两次间址

1. 目录结构

用16字节表示，其中14字节为文件名，2字节为I节点号

1. 用户及组结构

用户信息中包括用户名、口令，所属组，用户打开文件表

（文件树结构应与用户相对应：有每个用户的HOME目录）

组信息中可只包含组名

1. 文件树结构

除（4）要求外，适当考虑UNIX本身文件树结构

1. 实现功能如下命令

Ls 显示文件目录

Chmod 改变文件权限

Chown 改变文件拥有者

Chgrp 改变文件所属组

Pwd 显示当前目录

Cd 改变当前目录

Mkdir 创建子目录

Rmdir 删除子目录

Umask 文件创建屏蔽码

Mv 改变文件名

Cp 文件拷贝

Rm 文件删除

Ln 建立文件联接

Cat 连接显示文件内容

Passwd 修改用户口令

1. 可考虑构造一个简单的编辑器，用于创建文件
2. 需要创建“sbinfo”命令查看超级块里面数据块内容
   1. 空闲盘块栈
   2. Inode使用情况
   3. …..

# 二、实验环境

操作系统：windows 10 64位

开发工具：visio studio 2019

程序类型：win32控制台应用程序

# 三、实验设计

## 3.1系统流程

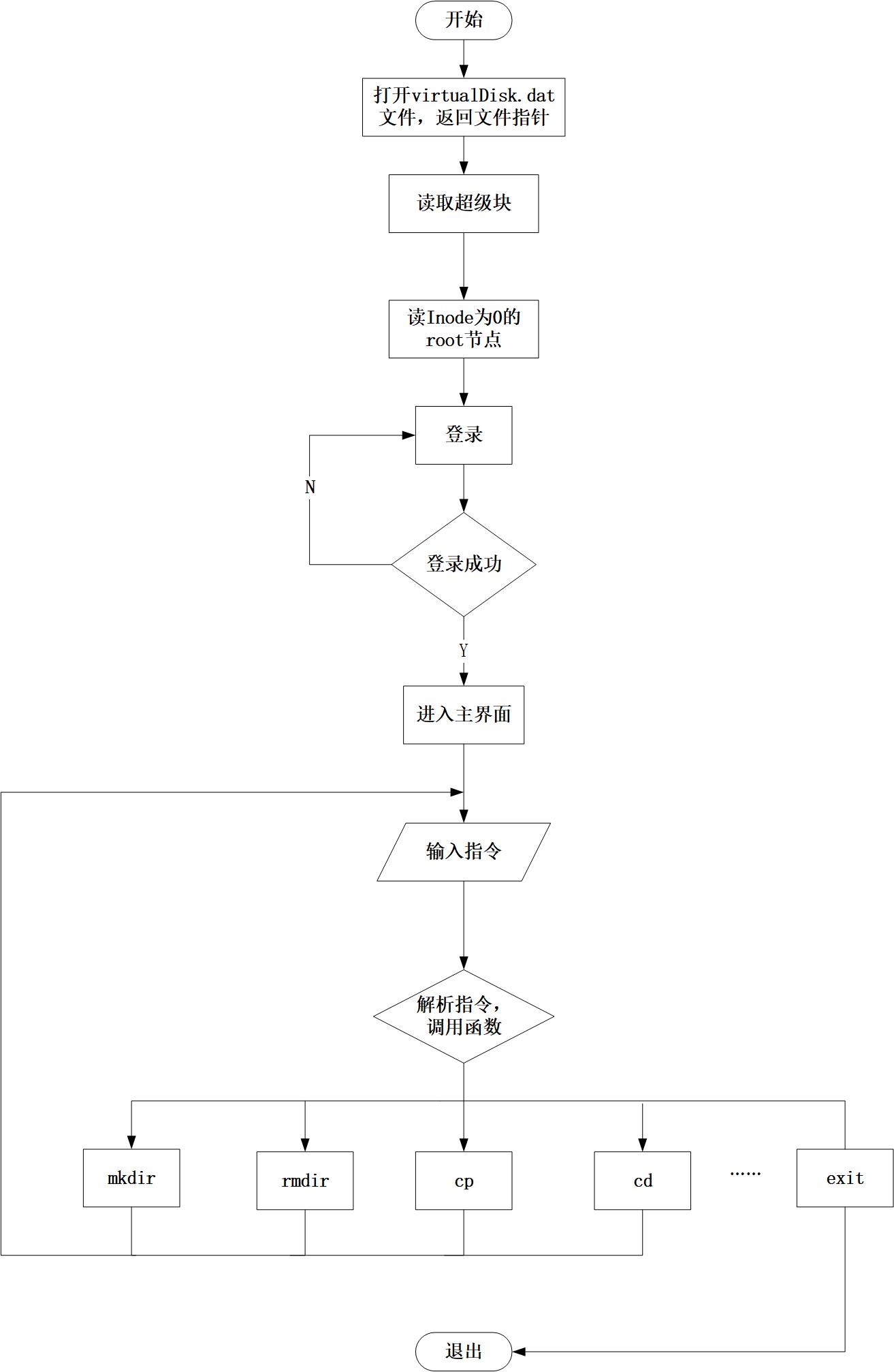


图3.1.1系统流程图

程序开始后先加载资源文件virtualDisk.da文件，将文件指针赋给全局变量virtualDisk（fstream）类型，以便后续文件的读写。

然后读取存储在盘块号为0中的superBlock的内容，赋给全局变量superBlock

然后读取InodeID为0的root节点，赋给全局变量root。

进行登录，系统首先读取root目录下第一个文件的内容，该文件文件名为user,里面存放了用户的数据结构，放在全局变量users数组，之后读取用户输入的用户名和密码，与users中的用户信息进行比对，若匹配则登录成功，若不匹配则显示用户名密码错误，重新登录。

登录成功后进入命令行界面，通过辨析用户输入的指令调用不同的函数，完成用户的请求。当用户输入exit时退出系统。

## 3.2文件结构

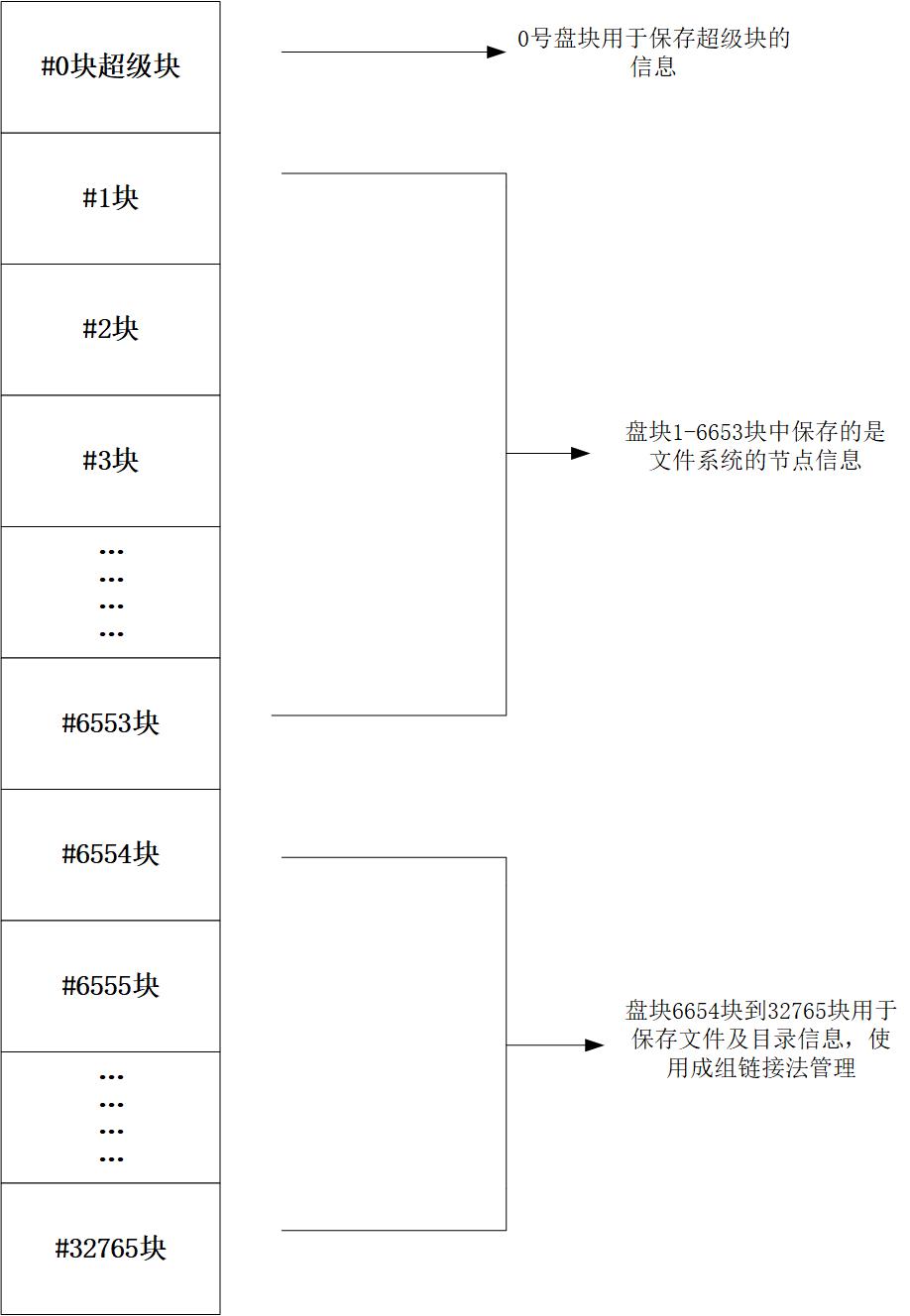


图3.2.1文件结构图

虚拟磁盘一共16MB，一个盘块512字节，一个Inode节点128字节，所以一个盘块里存放4个Inode,所以1/4的盘块(6653个盘块)用来存放Inode，3/4的盘块用来存放文件数据(26212个盘块)

## 3.3实现命令操作

本程序主要实现了以下命令操作

ls 显示文件目录

ll 显示文件目录里文件和文件的详细信息

mkdir 创建一个空目录

rmdir 递归删除目录里的内容以及目录本身

mv 移动目录及文件到指定目录，并指定文件名

cp 复制文件到指定目录

cat 连接显示文件内容，当后跟一个>时表示新建一个文件，

并输入内容，当后跟>>时，表示在现有文件后面追加内容

cd 改变当前目录

cpdir 复制目录及目录下的文件到指定目录下，并指定新的目录

名

chmod 修改文件权限

chown 改变文件拥有者

chgrp 改变文件所属组

pwd 显示当前目录

passwd 修改当前用户口令

ln 为文件创建硬链接

# 四、数据结构

## 4.1超级块数据结构

0#块为超级块。磁盘的回收和索引节点的分配与回收将涉及到超级块，超级块是用于专记录空闲盘块与空闲节点使用情况的一个盘块，其中含有以下字段

1.disksize:虚拟磁盘的大小，以字节为单位

2.freeBlock[FREE\_BLOCK\_NUM]：空闲盘块栈，记录当前可用空闲盘块号的栈，用成组链接法管理。

3.freeBlockIndex：空闲盘块栈的指针，记录下一个可用空闲盘块的下标

4.freeTotalBlock：当前文件系统全部可用空闲盘块的数目

5.freeInode[FREE\_INODE\_NUM]：空闲节点栈，记录当前可用空闲节点编号的栈

6.freeInodeInde：空闲节点栈的指针，记录下一个可用空闲节点的下标

7.freeTotalInode：当前文件系统全部可用空闲节点的数目

8.lastLoginTime:上次登录的时间

9.umask：权限掩码，创建文件或目录时他们的权限为默认权限减去umask

|  |
| --- |
| class SuperBlock {  public:  unsigned int diskSize;  unsigned int freeBlock[FREE\_BLOCK\_NUM];  int freeBlockIndex;  unsigned int freeTotalBlock;  unsigned int freeInode[FREE\_INODE\_NUM];  int freeInodeIndex;  int freeTotalInode;  long int lastLoginTime;  char umask[UMASK\_SIZE];  }; |

## 4.2节点数据结构

关于索引节点，程序中有两种结构的定义，分别是内存索引节点和磁盘索引节点，当磁盘索引节点加载到内存时为其增加一个InodeID（索引节点号）属性。

磁盘索引节点：

1.fileSize:文件大小，以字节为单位

2.fileLink:文件连接数

3.addr[6]:盘块地址数组

4.owner:所属用户名

5.group:所属用户组

6.mode:文件类型及权限

7.modifiedTime:上次修改时间

结构定义：

|  |
| --- |
| class Finode {  public:  long int fileSize;  int fileLink;  int addr[6];  char owner[OWNER\_MAX\_NAME];  char group[GROUP\_MAX\_NAME];  int mode;  long int modifiedTime;  char blank[48];  }; |

内存索引节点：

1.finode:磁盘索引节点结构，保存从磁盘读出的索引节点信息

2.inodeID:索引节点号

结构定义

|  |
| --- |
| class Inode {  public:  Finode finode;  unsigned short int parentInodeID;  unsigned short int inodeID;  }; |

## 4.3文件数据结构

文件目录项：

1.inodeID:索引节点号

2.directEntryName[DIRECT\_MAX\_NAME]:目录项名

结构定义

|  |
| --- |
| class DirectEntry {  public:  unsigned short int inodeID;  char directEntryName[DIRECT\_MAX\_NAME];  }; |

## 4.4目录数据结构

目录结构：

1.directEntryNum:目录项数目

2.directEntry[ENTRY\_NUM\_IN\_BLOCK]

结构定义：

|  |
| --- |
| class Direct {  public:  int directEntryNum;  DirectEntry directEntry[ENTRY\_NUM\_IN\_BLOCK];  }; |

# 五、模块详解

## 5.1文件操作

关于文件，本系统主要实现了以下操作：

1. chgrp 改变文件所属组：
2. 命令格式：

chgrp [组名] [文件名]

1. 命令功能：

通过chgrp命令可以改变文件的所属用户组名，此用户组名必须是已经存在的用户组。使用权限是超级用户。

1. 命令参数：

[组]：用户组名称

[文件]：当前目录下的文件名称

1. chown 改变文件拥有者：
2. 命令格式：

chown [用户名][文件名]

1. 命令功能：

通过chown命令改变文件的所属用户名，此用户名必须是已经存在的用户名。使用权限是超级用户

1. 命令参数：

[用户名]：要更改的新的用户名

[文件]：当前目录下的文件名

1. chmod 改变文件权限：
2. 命令格式：

chmod [mode] [文件名]

1. 命令功能：

chmod命令用于不同用户和用户组对文件或目录的访问权限。用它控制文件或目录的访问权限。

1. 命令参数：

[mode] 文件权限码

r=4，w=2，x=1  
若要rwx属性则4+2+1=7  
若要rw-属性则4+2=6；  
若要r-x属性则4+1=5。

mode权限码一共有三位数字分别代表所属用户，所属用户组和其他用户对其拥有的权限。

例如764则代表该文件所属用户对 该文件拥有读写执行的权限，与该文件所属用户组相同的用户对其拥有读写权限，其他用户对其拥有读权限

[文件名] 当前目录下要更改的文件名称

1. cat 连接显示文件内容：
2. 命令格式：

cat [选项] [文件名]

1. 命令功能：

cat命令的用途有连接文件并打印，或者新建一个文件，或者是向已有的文件中追加内容。

1. 参数说明：

[选项]选项中可以什么都不填，也可以填> 或>>

当选项一栏无参数时，表示连接并显示文件内容，例如 cat file1

当填>时，代表新建一个文件，例如 cat > file2

当填>>时，代表向一个已有文件中追加内容，例如cat >> file3

[文件名]当前目录下的文件名，或一个新的文件名

1. cp 文件拷贝：
2. 命令格式：

cp [源文件名] [目的目录路径]

1. 命令功能：

cp命令用来复制文件到指定的目录下。

1. 命令参数：

源文件名：当前目录下的源文件名

目的目录路径：要复制到哪个目录下，支持相对路径和绝对路径。例如cp file1 ../dir1/dir2或cp file1 /dir3/dir4

1. mv 移动文件

1) 命令格式：

mv [文件或目录名] [新的文件路径]

2) 命令功能：

用于移动文件或目录到指定目录下并指定新的文件或目录名，支持绝对路径和相对路径

3) 命令参数

[文件名或目录名]当前目录下要移动的文件或目录名

[新的文件路径]指定要移动的目录路径和新的文件或目录名，例如mv file1 ../dir/dir/newfilename

1. ln 创建连接

1) 命令格式：

ln [文件名] [新的文件路径]

2) 命令功能：

为某一个文件在另外一个位置建立一个同步的链接，支持绝对路径和相对路径

3) 命令参数

[文件名]当前目录下要创建链接的文件名

[新的文件路径]在指定的目录下创建一个新的链接并指定链接名，例如 ln file1 ../dir/dir/newfilename

1. rm 删除文件

1) 命令格式：

rm [文件名]

2) 命令功能：

删除当前目录下的文件

3) 命令参数

[文件名]当前目录下的文件名

## 5.2目录操作

关于目录，本模拟系统主要实现以下几个指令：

1. ls显示文件目录：
2. 命令格式：

ls

1. 命令功能：

ls命令是UNIX下最常用的命令。ls命令就是list的缩写，用来打印出当前目录的清单

1. pwd 显示当前目录：
2. 命令格式：

pwd

1. 命令功能：

 pwd 命令来查看”当前工作目录“的完整路径。 简单得说，每当你在终端进行操作时，你都会有一个当前工作目录。

1. cd 改变当前目录：
2. 命令格式：

cd [目标目录名]

1. 命令功能：

cd指令的功能是进入目录，支持相对路径和绝对路径。

1. 命令参数：

[目标目录名]指定的目录路径，例如 cd dir1/dir

1. mkdir 创建空目录：
2. 命令格式：

mkdir [目录名]

1. 命令功能：

mkdir指令是用于在当前目录下创建子目录的指令。

1. rmdir 删除子目录：

1） 命令格式：

rmdir [目录名]

2） 命令功能：

该命令从一个目录中删除一个目录项，目录项下的所有文件和目录也将被删除

1. cpdir 复制目录

1） 命令格式：

cpdir [目录名] [目标目录名]

2） 命令功能：

该命令将一个目录复制到指定的目录下，并指定新的目录名，支持绝对路径和相对路径

3) 命令参数：

[目标目录名]在指定的目录下复制目录，并指定新的目录名，例如 cpdir dir1 dir2/dir3/newdir

7．mv 移动文件

1) 命令格式：

mv [文件或目录名] [新的文件路径]

2) 命令功能：

用于移动文件或目录到指定目录下并指定新的文件或目录名，支持绝对路径和相对路径

3) 命令参数

[文件名或目录名]当前目录下要移动的文件或目录名

[新的文件路径]指定要移动的目录路径和新的文件或目录名，例如mv file1 ../dir/dir/newfilename

8. ll 显示目录项信息

1） 命令格式：

ll

2） 命令功能：

显示目录项的详细信息

## 5.3主函数

主函数中最重要的是readCommend()函数，读取用户的输入并解析指令调用对应的函数,通过使用strtok()函数以空格分割字符串，通过比较分割空格得到的第一个字符串来判定用户调用的是什么命令，来调用不同的函数并传递对应的参数。流程图如下。

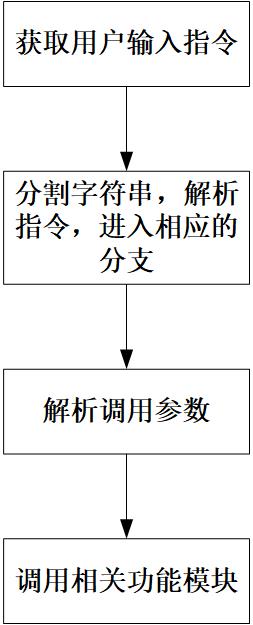


图5.3.1主函数流程图

## 5.4核心函数

以下介绍一下本系统中的一些基础函数，其他命令函数的实现是建立在这些函数的基础上。

1. 获取Inode在磁盘里的起始地址

inline int getInodePos(int inodeID) ;

根据inodeID获取inode在磁盘里的起始地址；

2.获取block在磁盘里的起始地址。

inline int getBlockPos(int blockNumber) ;

根据盘块号获取盘块在磁盘里的起始地址。

3.同步inode函数

int synchronizationInode(Inode\* inode)

将在内存中修改过的inode写回磁盘

4.同步superBlock函数

int synchronizationSuperBlock()

将在内存中修改过的superBlock写回磁盘。

5.获取空闲盘块块号函数

int getFreeBlockID();

从superBlock的空闲盘块栈中获取一个空闲盘块号，若superBlock

中的空闲盘块栈为空，则再从磁盘中读取一组空闲盘块号

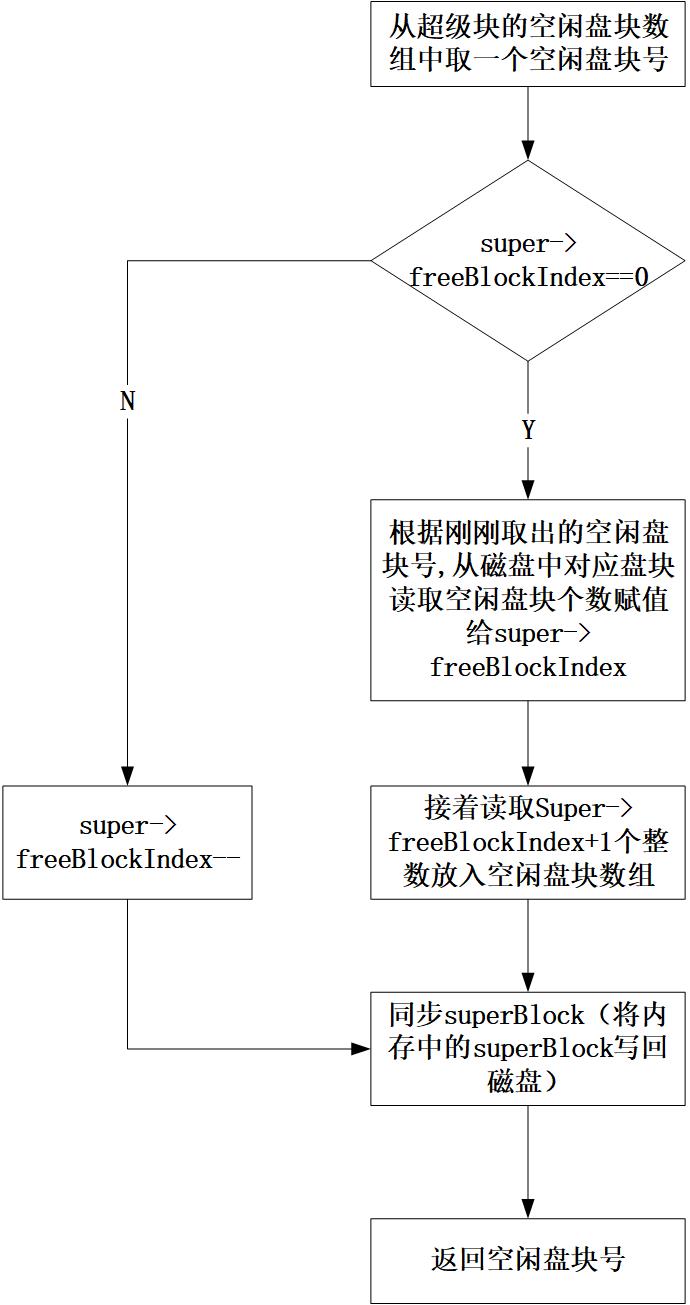


图5.4.1获取空闲盘块节点流程图

6.获取空闲节点编号

int getFreeInodeID();

每次从超级块空闲节点栈中取一个空闲节点号，若已经取完，则遍历 磁盘的所有节点，取一组放入空闲节点栈

7.释放盘块函数

void blockFree(int freeBlockID);

将被释放的盘块号放入超级块的空闲盘块数组，若数组已满，则将

超级块中的整个空闲盘块数组写入被释放的盘块中，令super->

freeBlockIndex=0,super->freeBlock[0]=被释放的盘块号。

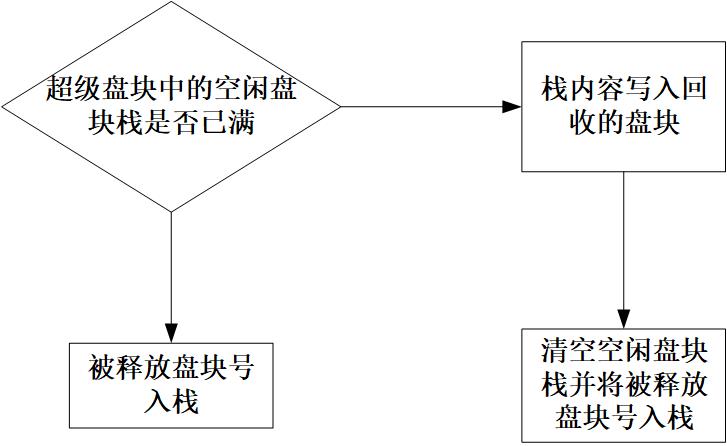


图5.4.2获取空闲节点函数流程图

8.查找目录项函数

int findFileInDir(Inode givenInode, char\* filename,int

&targetFreeBlockID)

遍历一个目录里所有目录项，并找到和给定的文件名匹配的目录

项，返回他的inodeID

# 六、实验演示

## 6.1进入文件系统

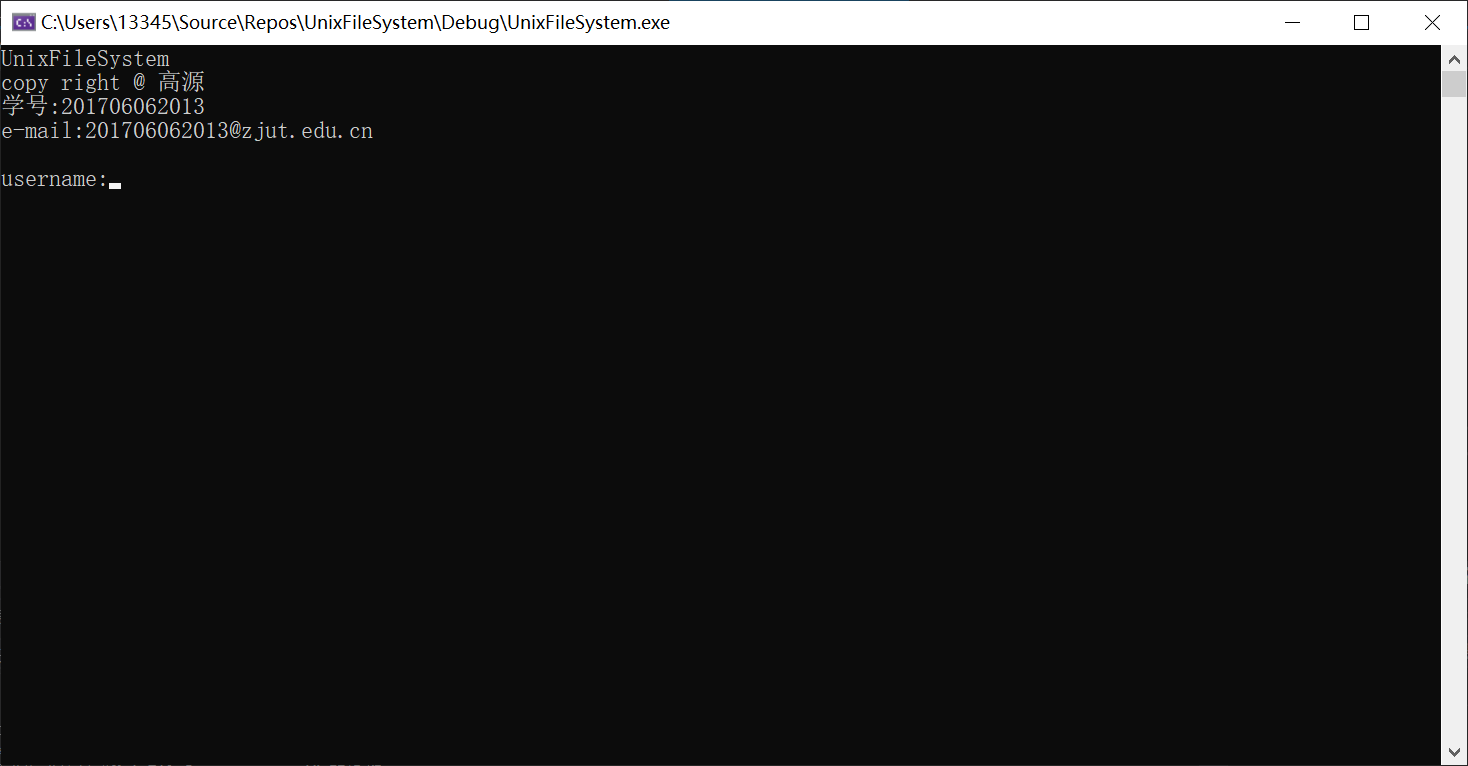


图6.1.1登录界面

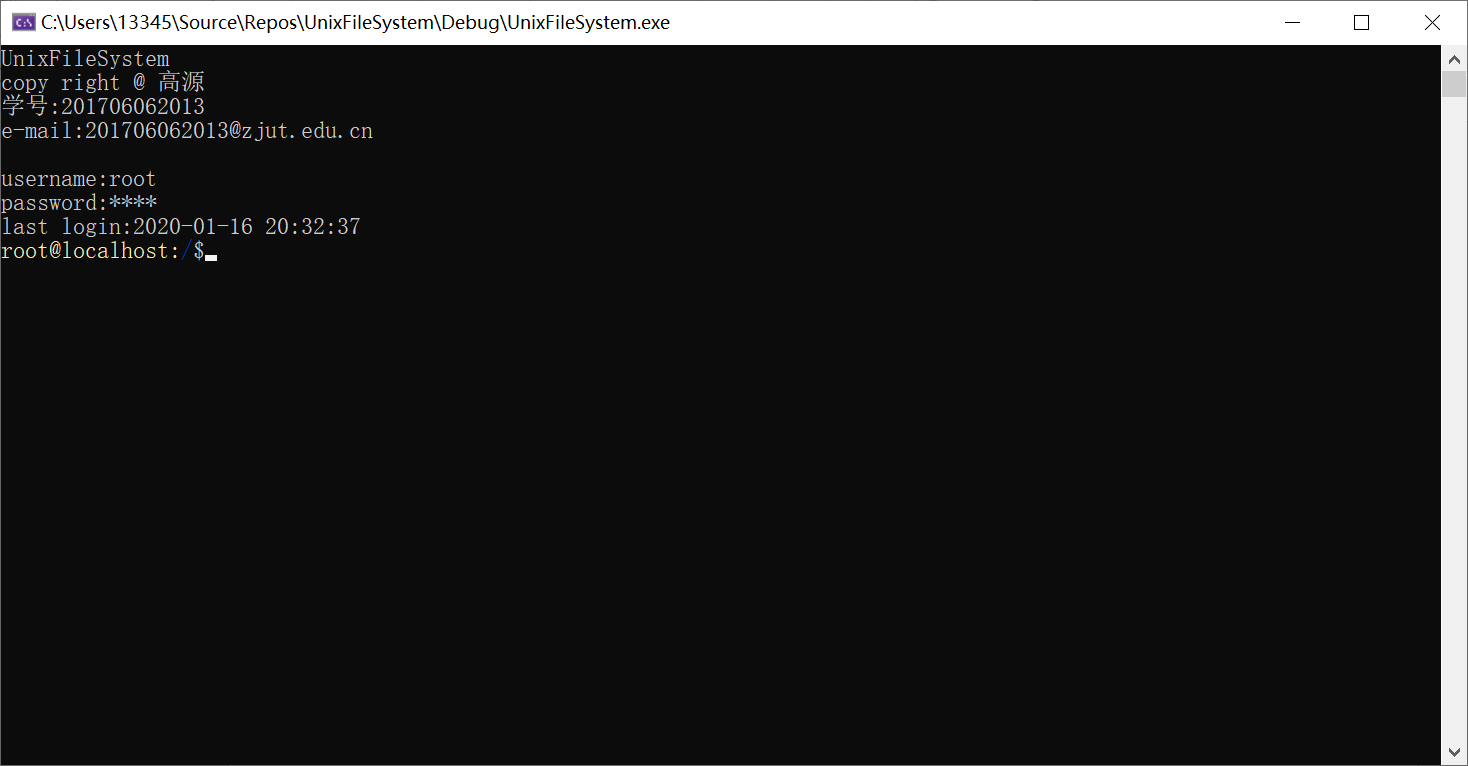


图6.1.2登录成功界面

## 6.2文件相关操作

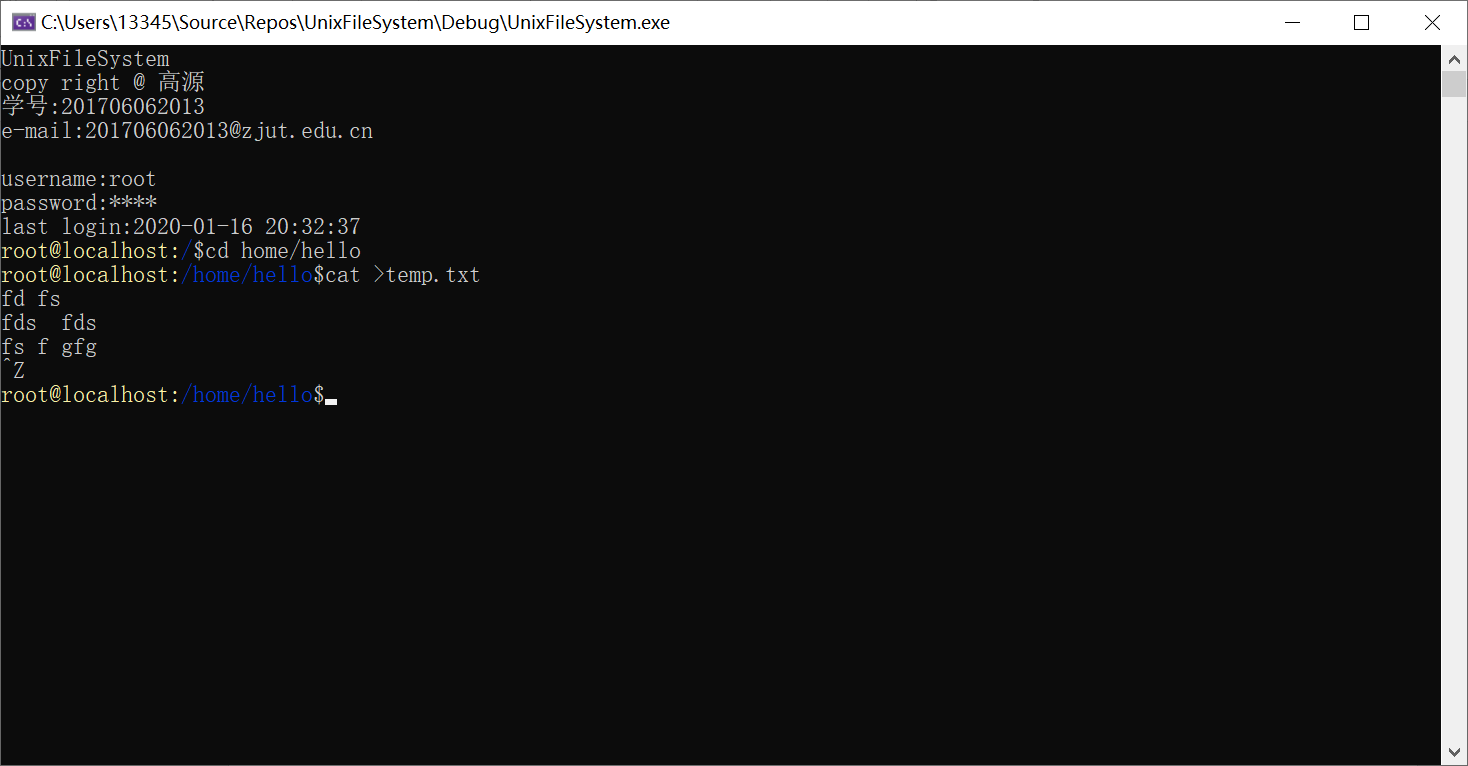


图6.2.1创建一个文件，并添加内容

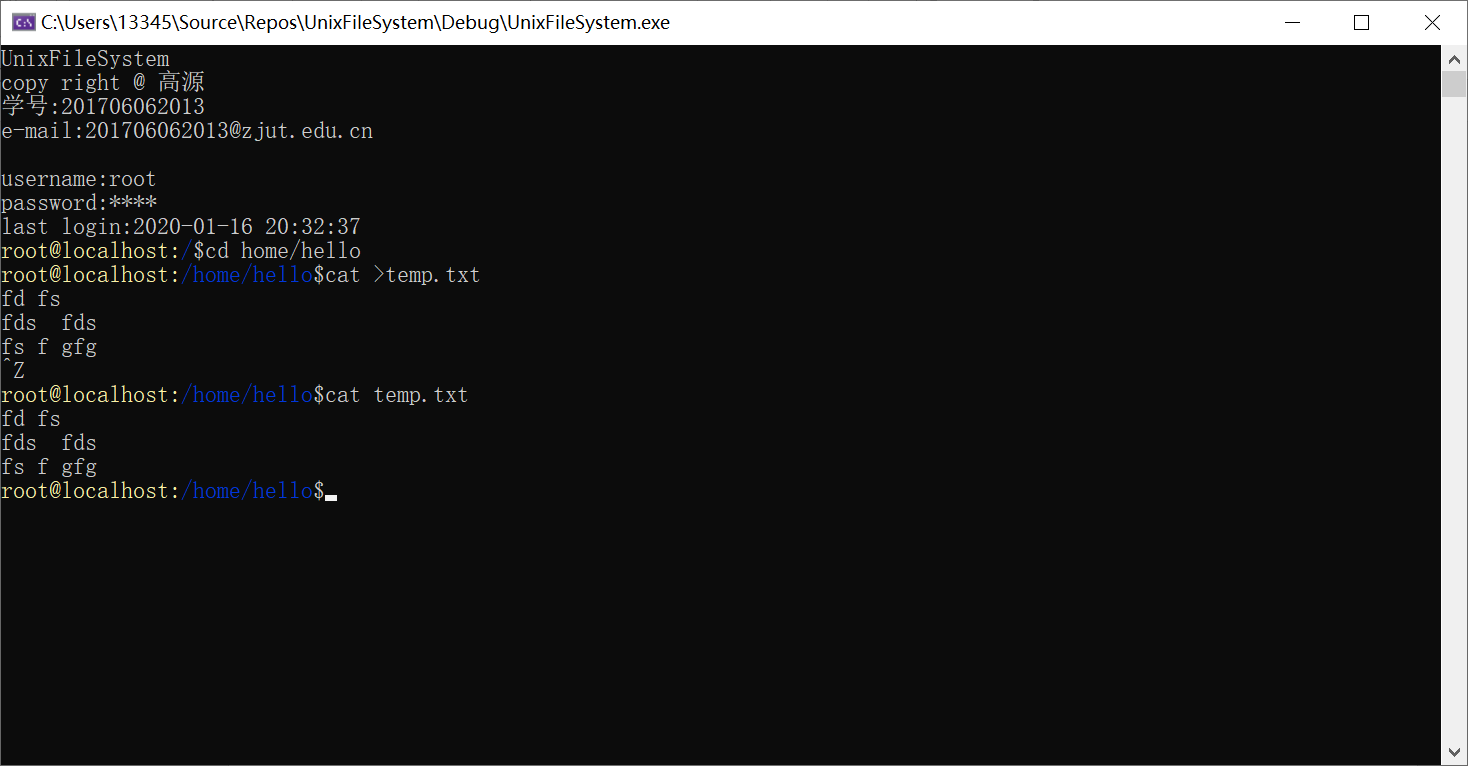


图6.2.2打印文件

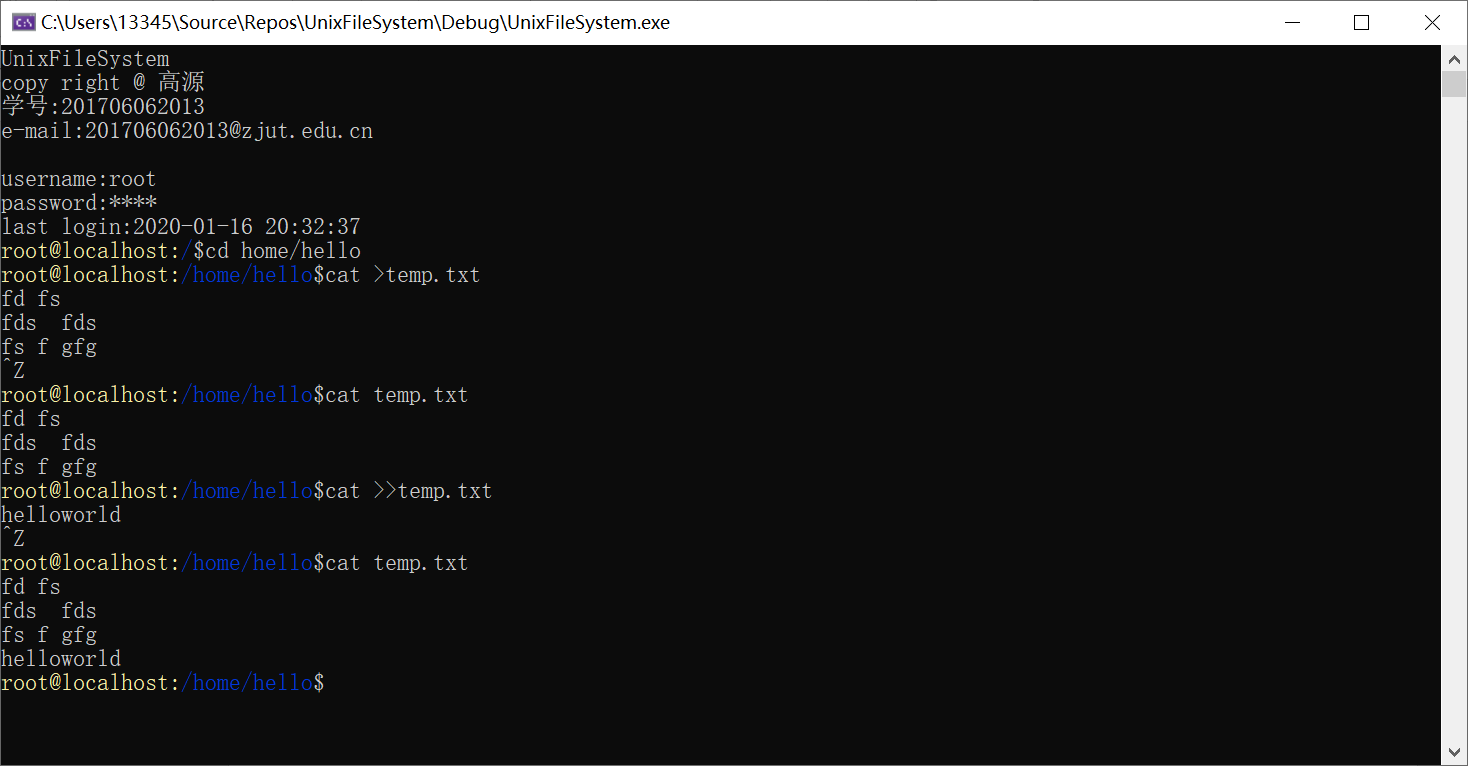


图6.2.3向文件中追加内容并打印

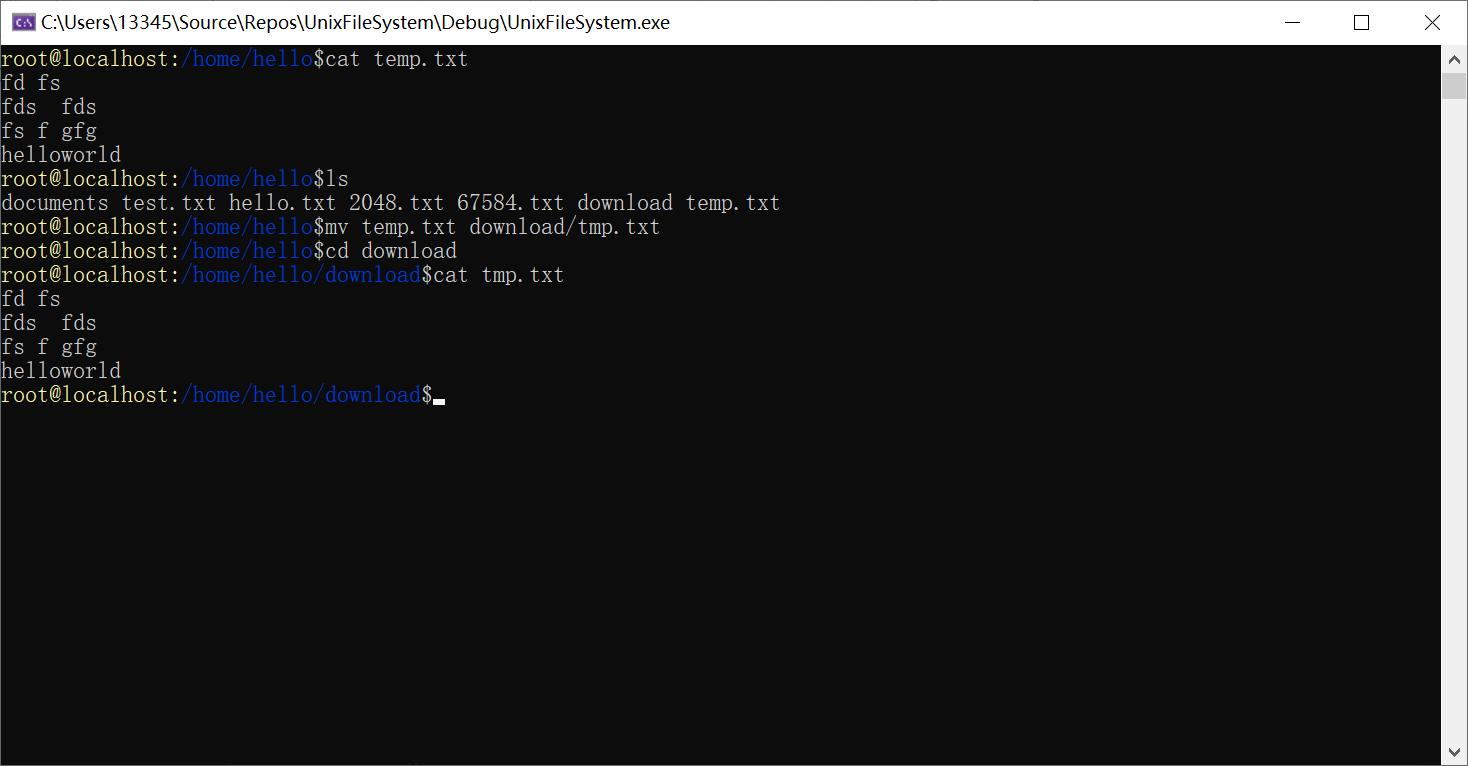


图6.2.4将文件移到另一个目录里,并重新赋予名字

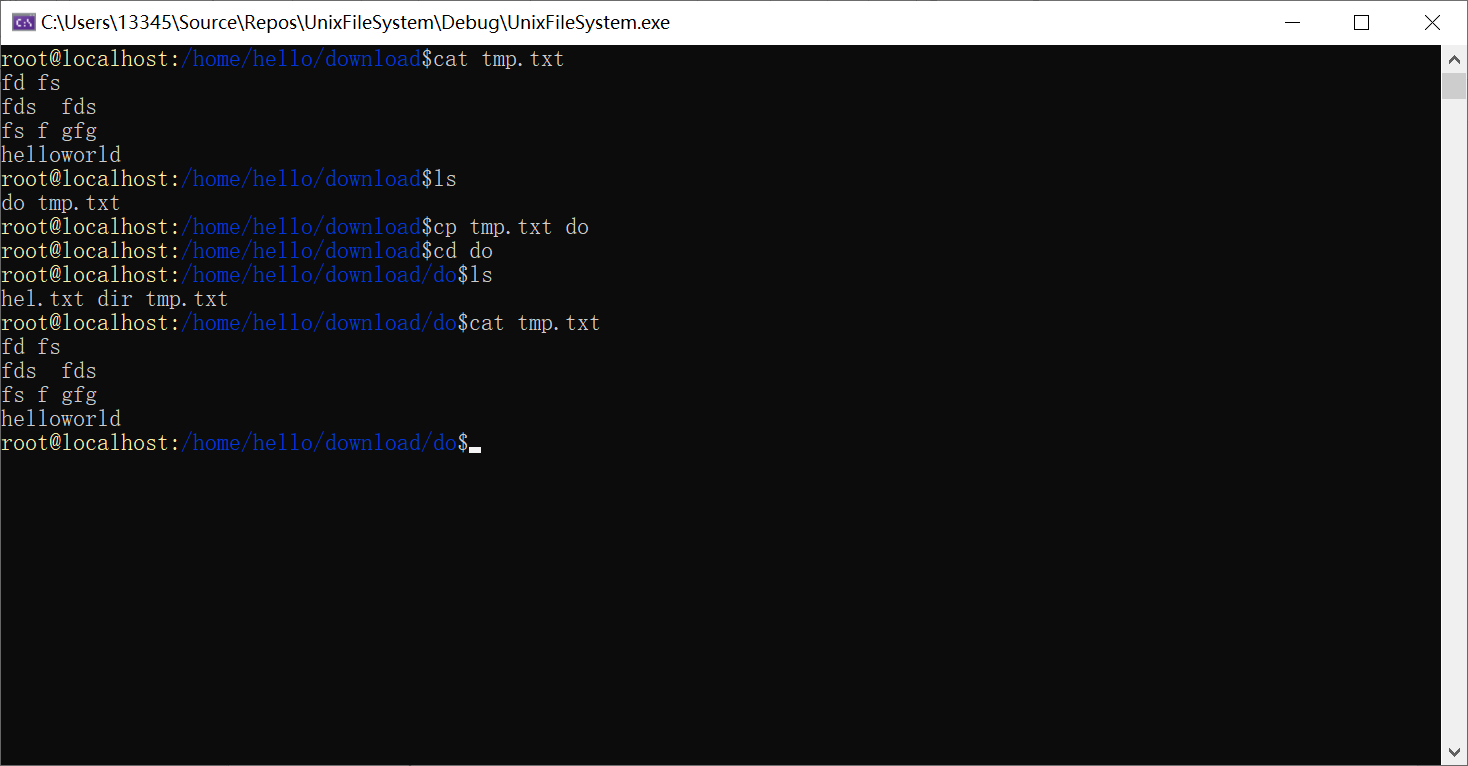


图6.2.5将文件复制到另一个目录中，并显示被复制的目录里的内容

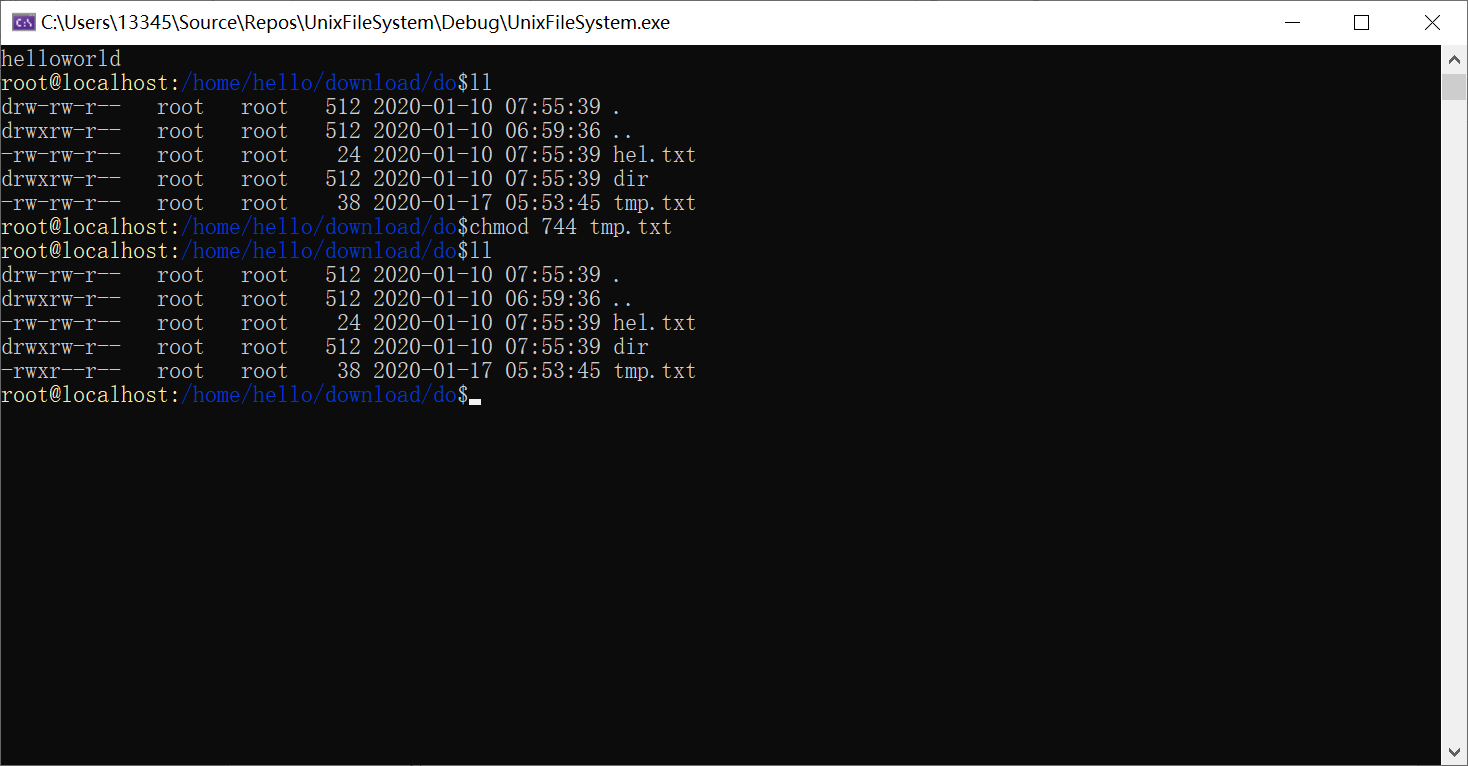


图6.2.6改变文件的访问权限，并显示文件的详细信息

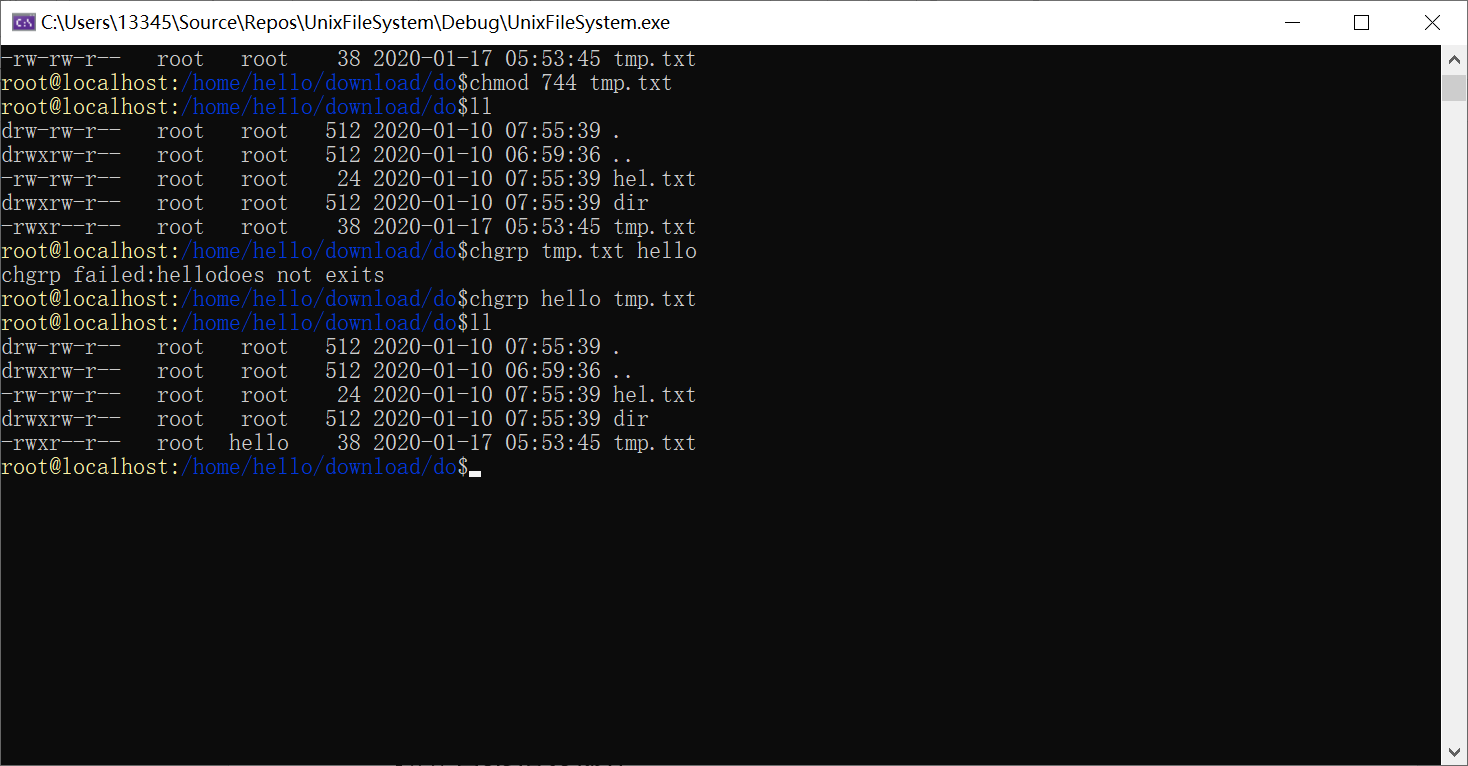


图6.2.7改变文件的所属用户组，并显示文件的详细信息

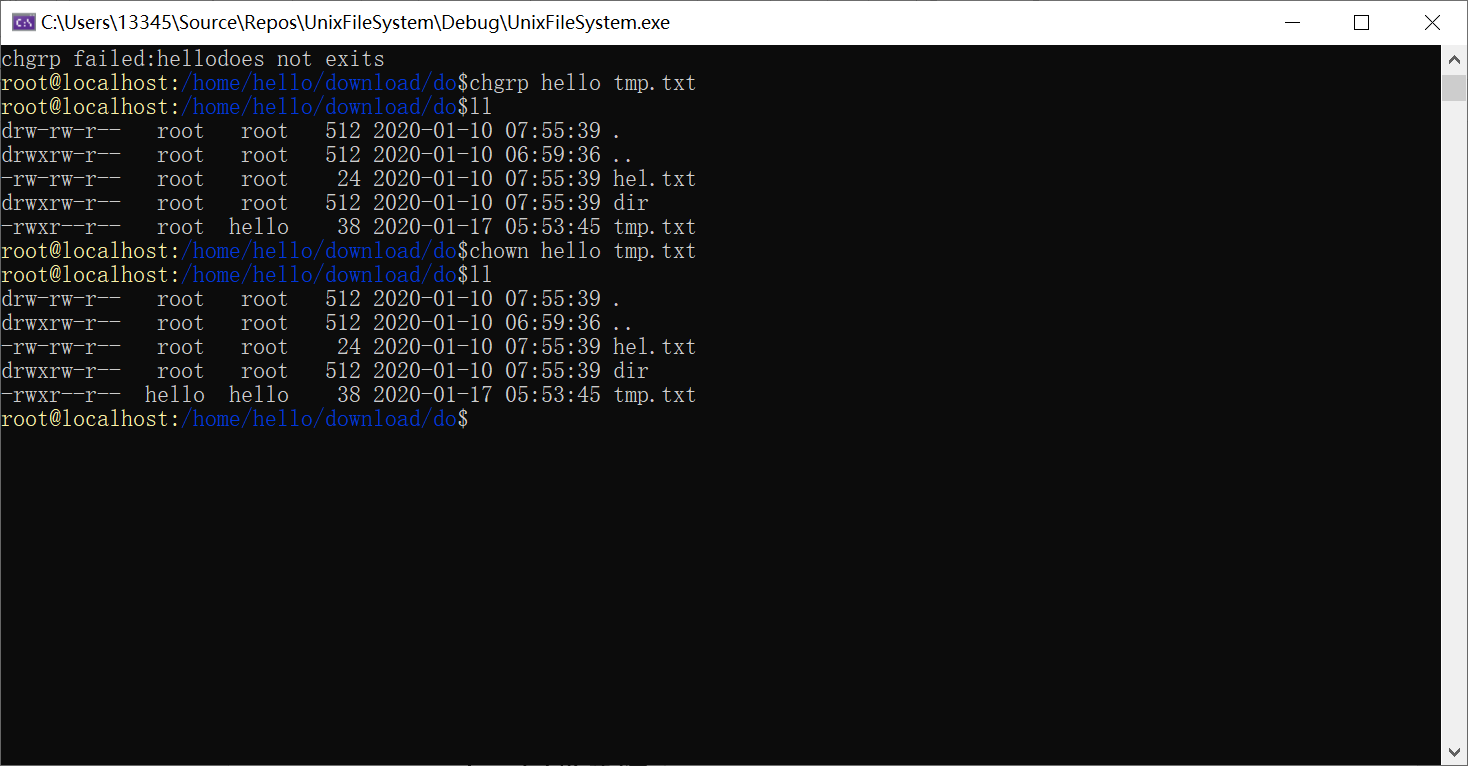


图6.2.8改变文件的拥有者，并显示文件的详细信息

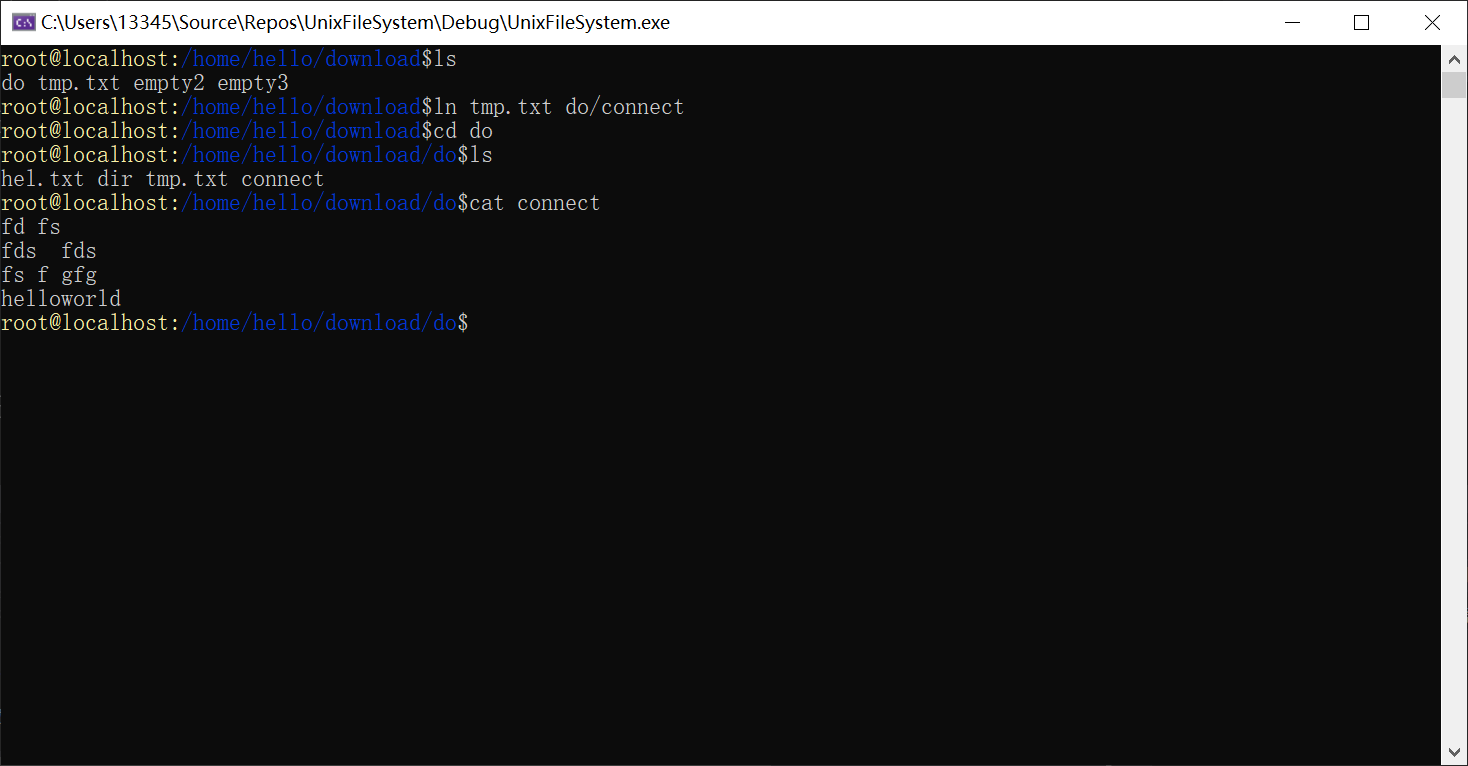


图6.2.9为文件创建硬链接

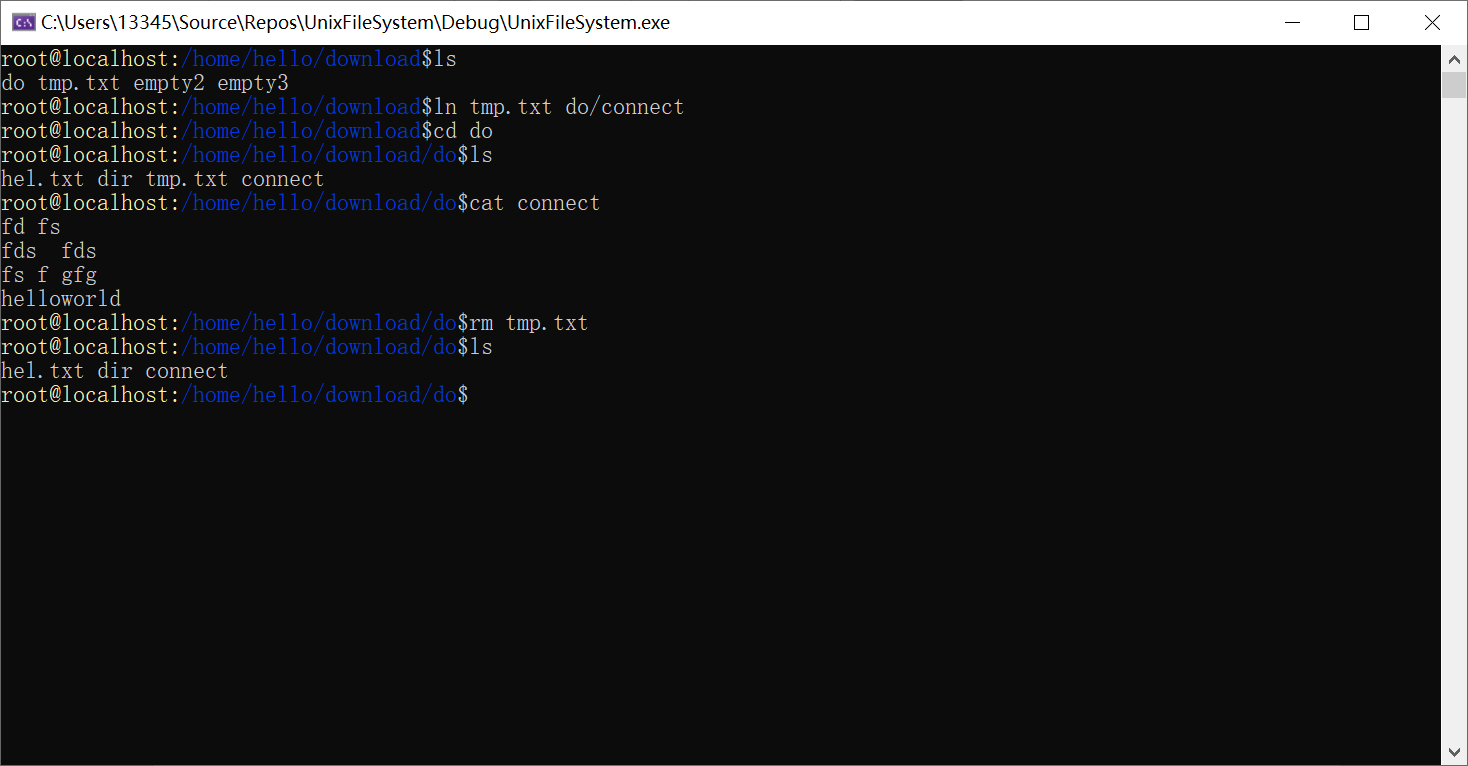


图6.2.10删除文件

## 6.3目录相关操作

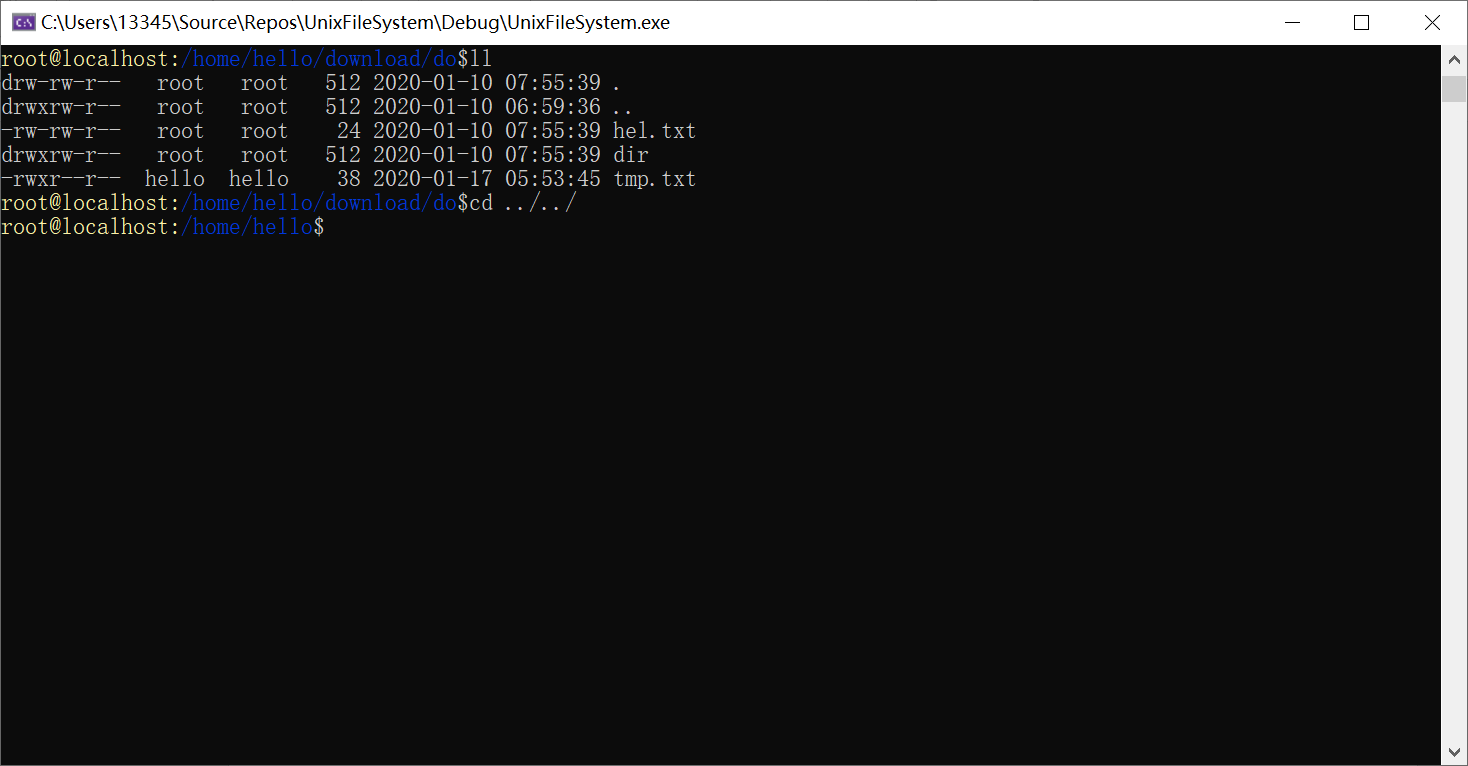


图6.3.1cd改变工作目录

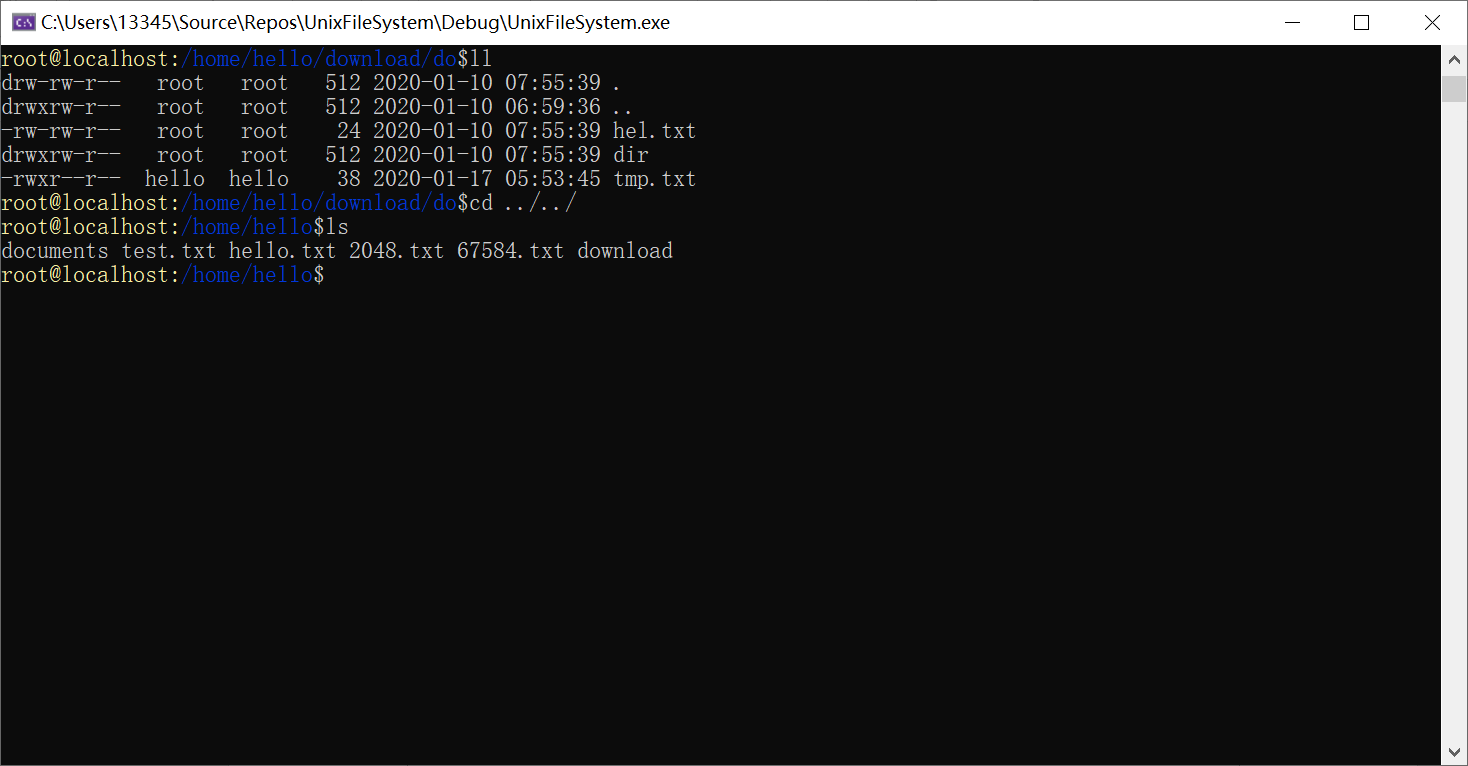


图6.3.2显示当前目录里的子目录以及文件

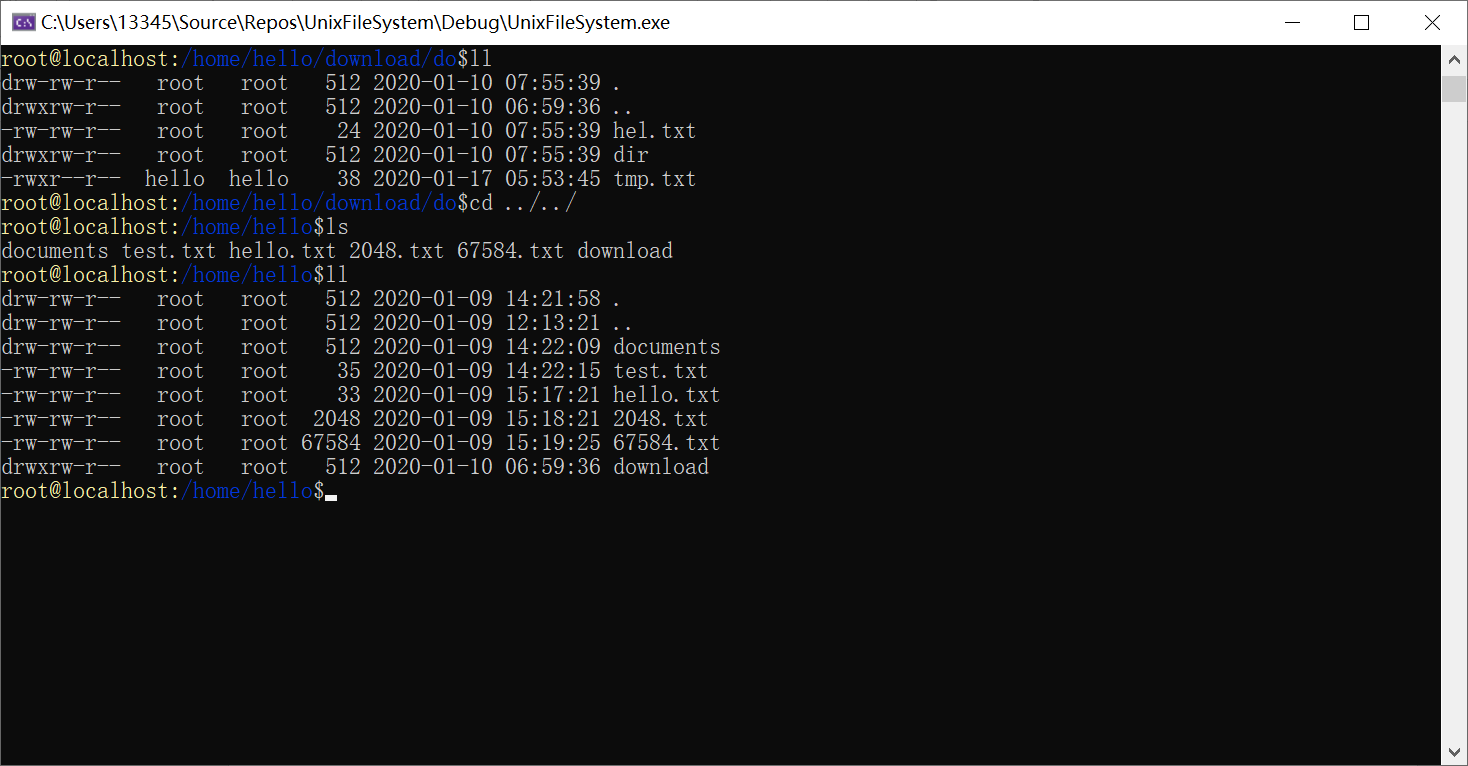


图6.3.3显示当前目录里子目录及文件的详细信息

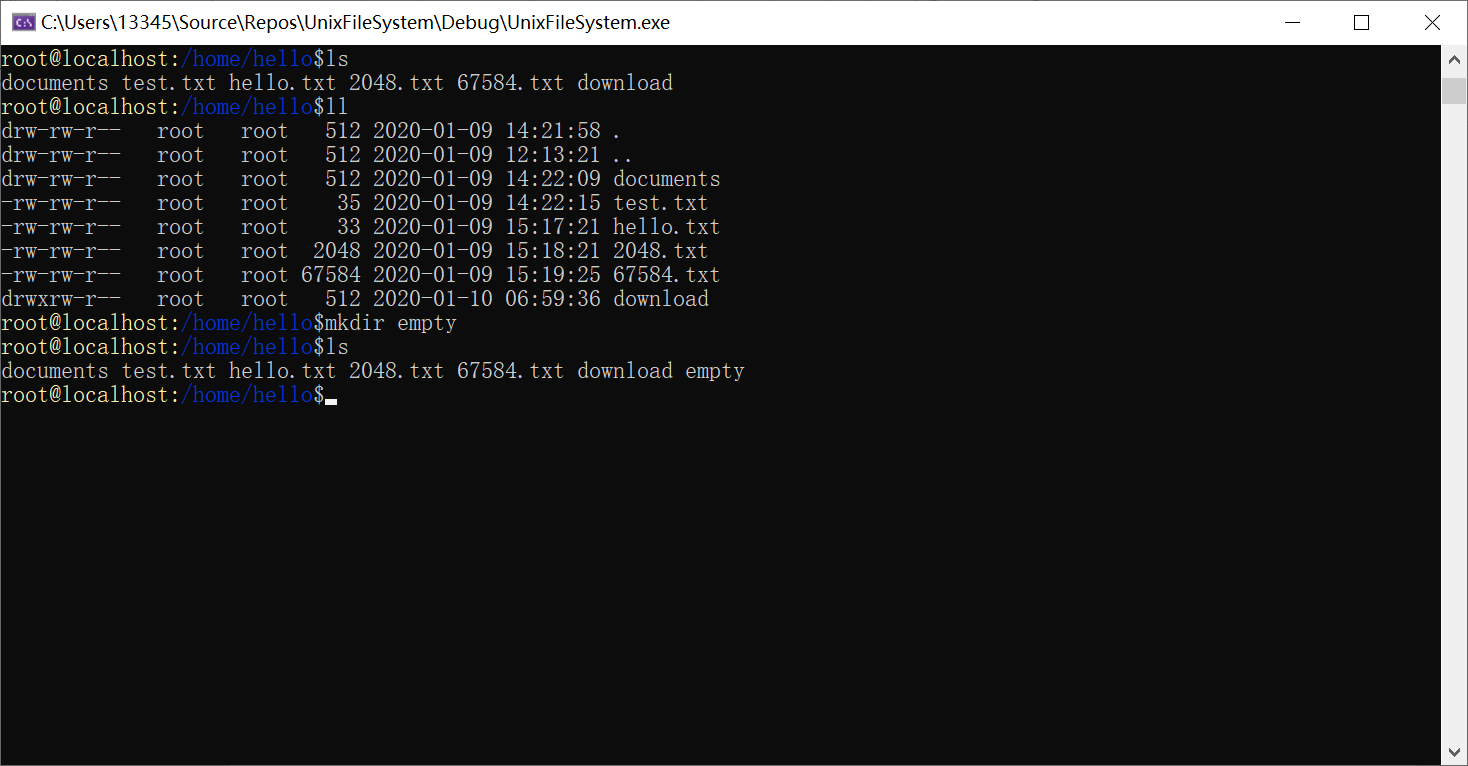


图6.3.4创建一个空目录

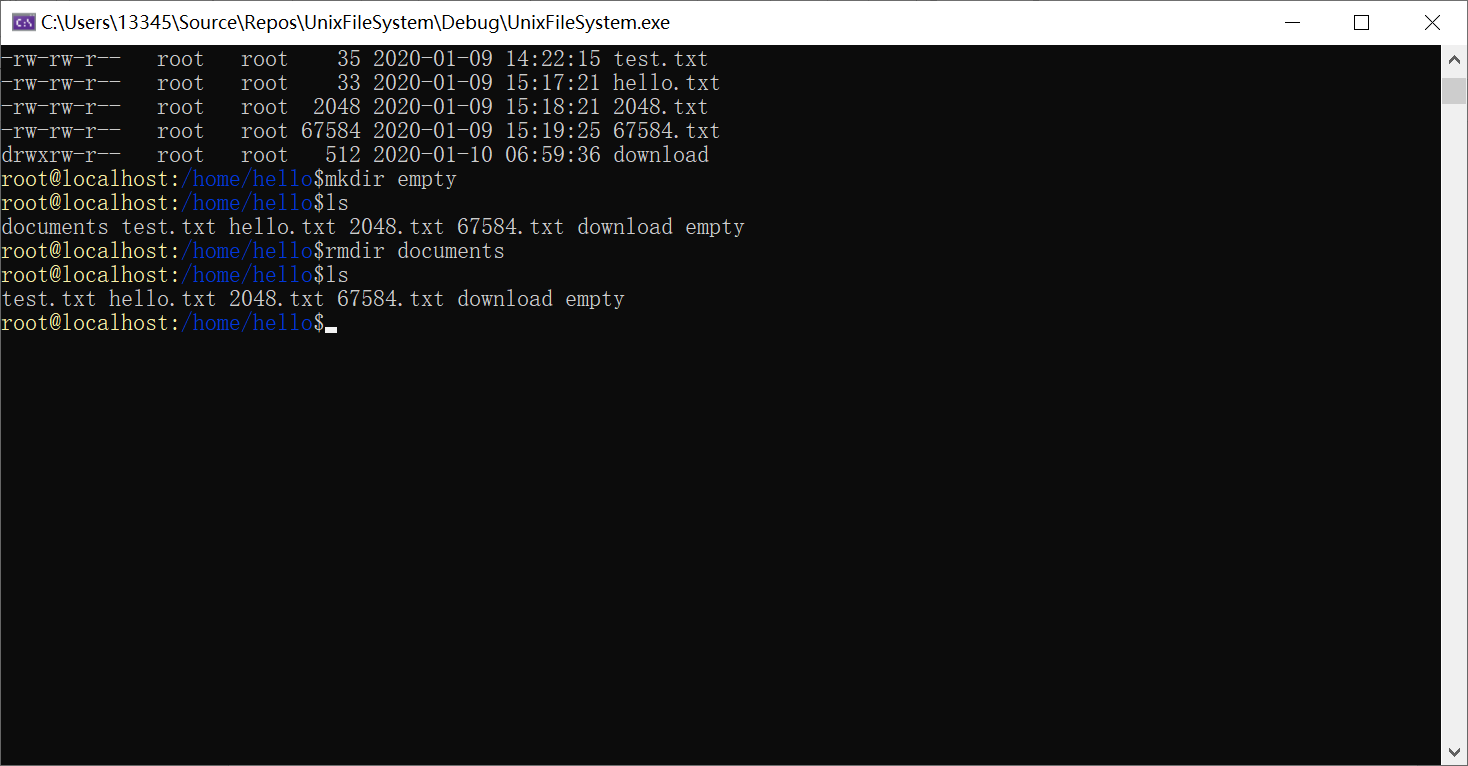


图6.3.5递归删除一个目录

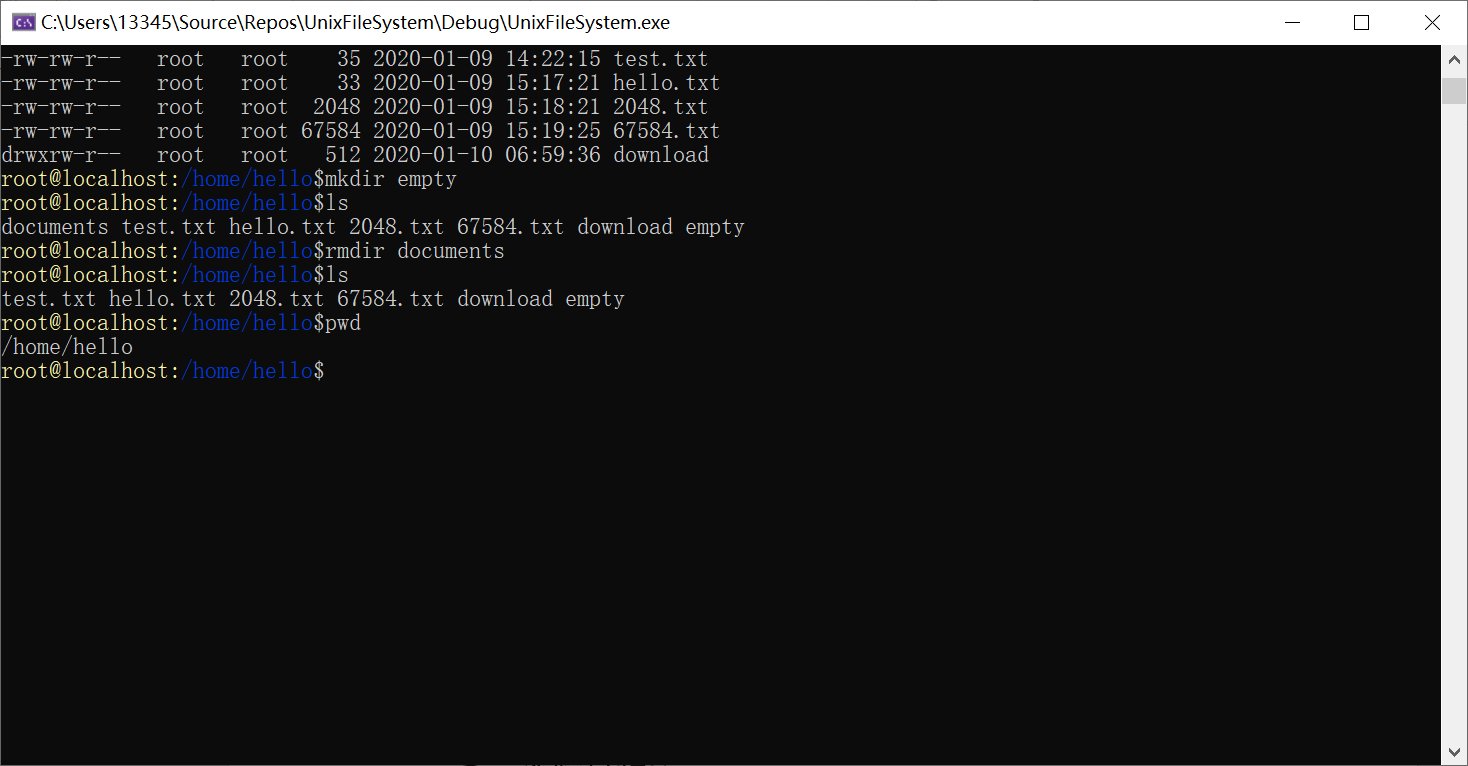


图6.3.6显示当前的工作目录路径

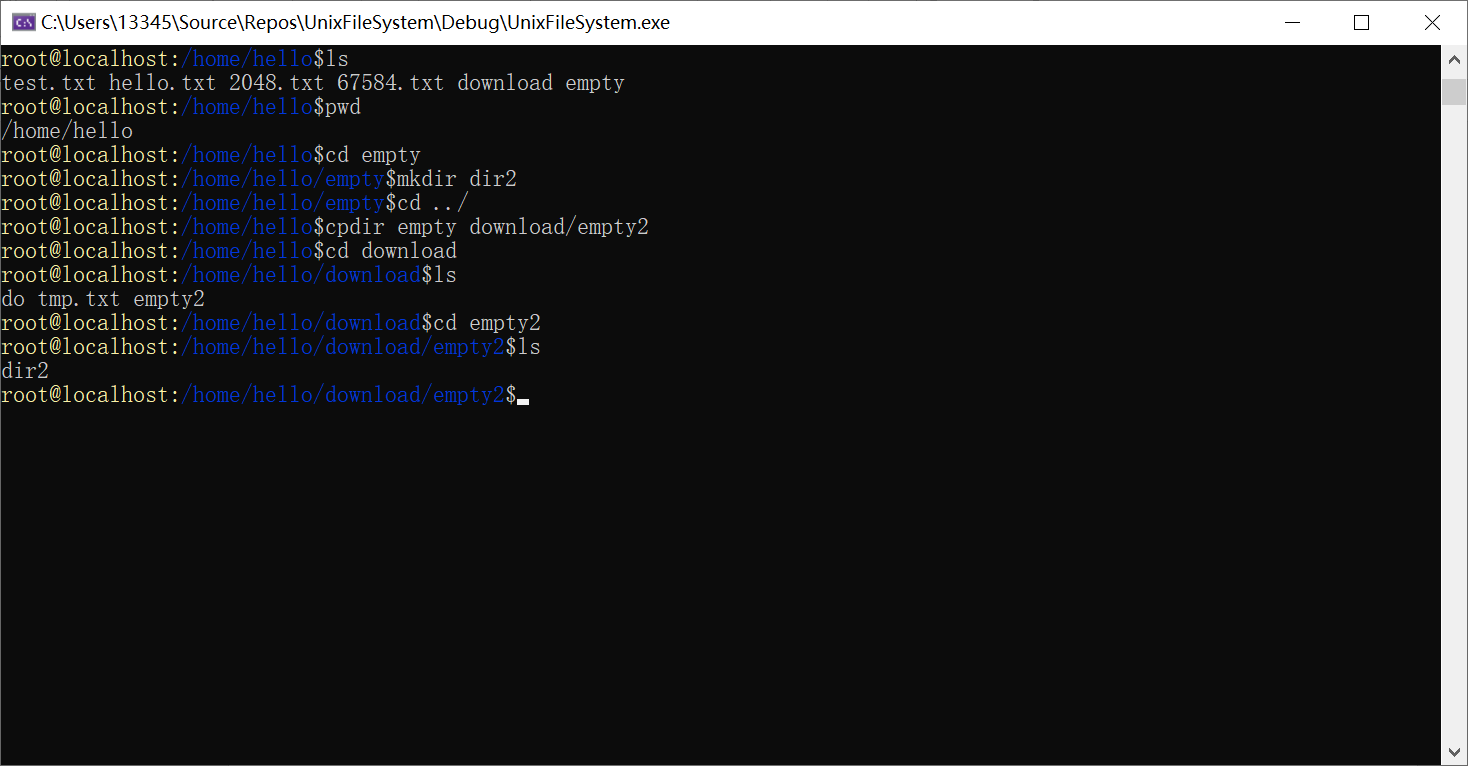


图6.3.7拷贝目录到另一个目录中，并显示被拷贝目录里的内容

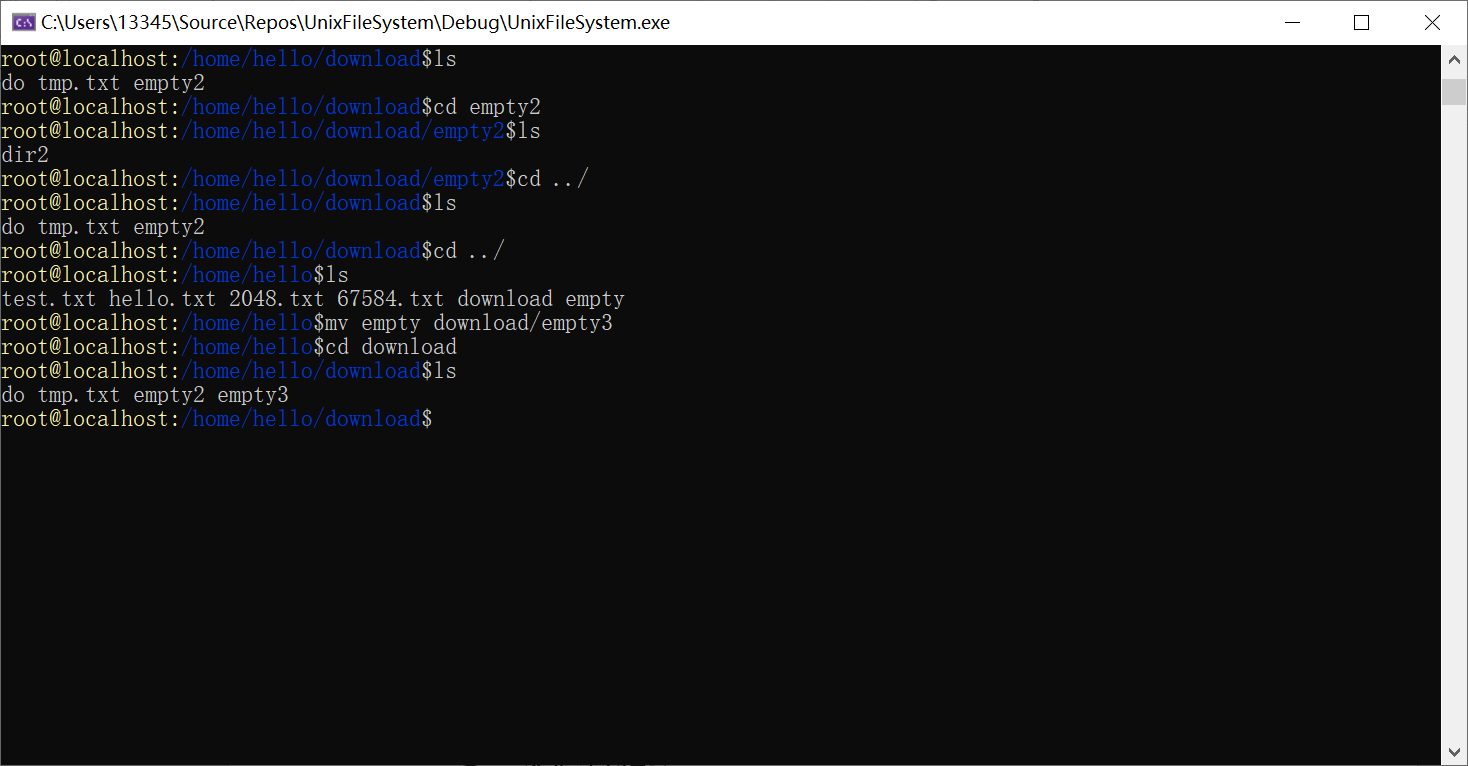


图6.3.8移动目录到另一个目录中，并指定新的目录名

# 七、实验总结

都说实践出真知。在做了这个文件系统后我对unix文件系统的认知可以说是更上一层楼。

在最初开始做的时候可以说是完全一头雾水，光在网上查资料，理解unix文件系统的结构就花了我三天时间，当我以为自己已经完全掌握开始做后，又频频遇到难题，光是初始化虚拟磁盘和实现mkdir命令就又花我三天时间，不过当基础打好后，后面进行得就越来越快了，加上自己模块化了一些常用得操作。比如代码里有一个findFileInDir函数，作用是遍历一个目录里所有目录项，并找到和给定的文件名匹配的目录项，返回他的inodeID,就是我自己设计的函数，实在减少了我很大的代码量。

经过这次课程设计，我不仅对文件系统的结构更加清楚，了解了盘块和节点的概念，而且也锻炼了我写代码的能力，收获颇丰。