

操作系统课程设计

**2019/2020(1)**



实验题目 模拟UNIX文件系统

学生姓名 潘伟航

学生学号 201706061714

学生班级 软件工程1701班

任课教师 李 波

提交日期 2020-01-17

**计算机科学与技术学院**

**目录**

[一、 实验内容 4](#_Toc30184009)

[二、 实验环境 5](#_Toc30184010)

[三、 实验设计 6](#_Toc30184011)

[3.1系统流程 6](#_Toc30184012)

[3.2 盘块的分配与回收流程 7](#_Toc30184013)

[3.3文件结构 9](#_Toc30184014)

[3.4实现命令操作 9](#_Toc30184015)

[3.5部分命令操作的流程图 11](#_Toc30184016)

[3.5.1 新建文件操作：touch 11](#_Toc30184017)

[3.5.2 新建文件操作：mkdir 12](#_Toc30184018)

[3.5.3 往文件写入操作：>> 13](#_Toc30184019)

[四、 数据结构 14](#_Toc30184020)

[4.1 超级块 14](#_Toc30184021)

[4.2磁盘INode 14](#_Toc30184022)

[4.3 内存INode 15](#_Toc30184023)

[4.4 目录项结构 16](#_Toc30184024)

[4.5 目录结构 16](#_Toc30184025)

[4.6 用户结构 16](#_Toc30184026)

[4.7用户组结构 17](#_Toc30184027)

[4.8 组结构 17](#_Toc30184028)

[4.9 组集合 17](#_Toc30184029)

[五、 模块详解 18](#_Toc30184030)

[5.1 文件操作 18](#_Toc30184031)

[5.2 目录操作 21](#_Toc30184032)

[5.3 主函数 23](#_Toc30184033)

[六、 实验演示 26](#_Toc30184034)

[6.1 登录 26](#_Toc30184035)

[6.2 help命令 26](#_Toc30184036)

[6.3 ls命令 27](#_Toc30184037)

[6.4 cd命令 27](#_Toc30184038)

[6.5 sp命令 28](#_Toc30184039)

[6.6 touch命令 29](#_Toc30184040)

[6.7 >>命令与cat命令 30](#_Toc30184041)

[6.8 cp命令 30](#_Toc30184042)

[6.9 umask命令 31](#_Toc30184043)

[6.10 mv命令 31](#_Toc30184044)

[6.11 mkdir命令 32](#_Toc30184045)

[6.12 在新建的目录中使用cd与touch命令 32](#_Toc30184046)

[6.13 rmdir命令 33](#_Toc30184047)

[6.14 ln命令 33](#_Toc30184048)

[6.15 pwd命令 33](#_Toc30184049)

[6.16 关于隐藏文件现实的命令 34](#_Toc30184050)

[6.17 输入的文件大小需要间址的情况 34](#_Toc30184051)

[七、 实验总结 36](#_Toc30184052)

[7.1实验过程遇到的问题 36](#_Toc30184053)

[7.2实验体验总结 37](#_Toc30184054)

**模拟UNIX文件系统**

# 实验内容

本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

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. …..
3. 上述提示仅供参考，欢迎同学们提出更好的建议，这一点将作为成绩的一个依据，但务必完成（6）中提出的功能要求。

要求完成大型实验的方案设计，编程测试，写出实验总结报告。

# 实验环境

系统操作环境：Win10

系统开发工具：Visual Studio 2017

# 实验设计

## 3.1系统流程

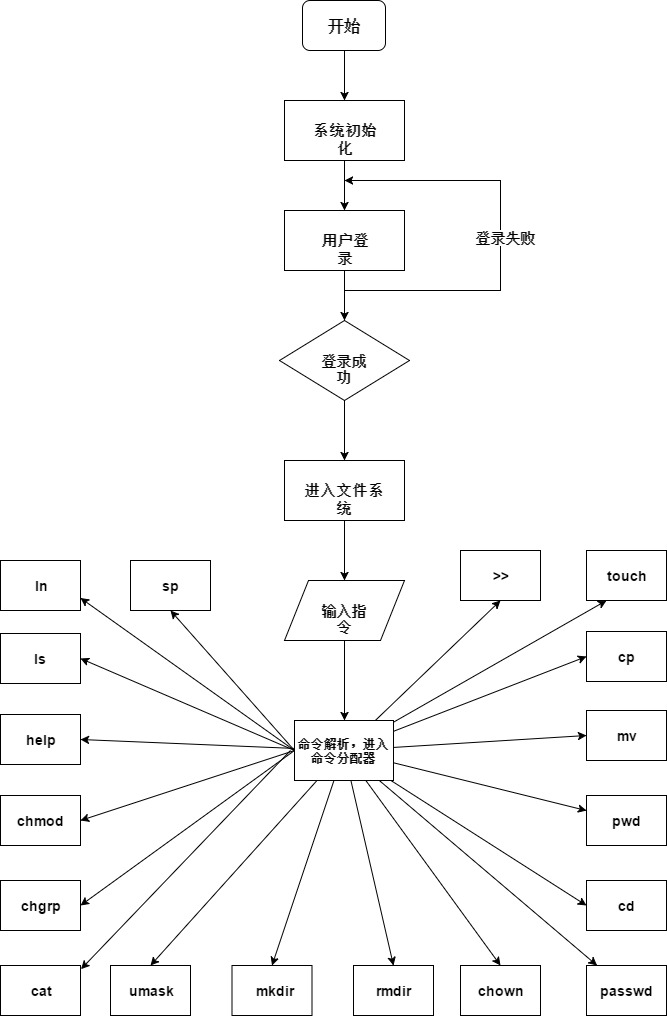


图3.1.1 系统流程图

注：每当一次命令执行完成之后都返回到命令界面。

说明：系统开始后进行初始化操作，首先判断模拟磁盘文件是否存在，如果不存在则先初始化一个模拟磁盘，存在则从模拟磁盘中读入超级块、用户、组、初始化当前目录。

接着是用户登录，登录成功后进入系统主界面，输入各项命令即可进行各项操作，但是要注意的是文件权限的操作。

## 3.2 盘块的分配与回收流程

盘块的分配与回收在UNIX文件系统中采用的是成组链接的方式来进行的。

首先是关于成组链接的介绍：

计算机上的文件是记录在磁盘上的，而磁盘空间的分配是以盘块为单位的，那么如何管理磁盘中已经被使用的块和未被使用的块是操作系统必须要考虑的问题。下面将介绍比较实用又有点复杂的成组链接法，看它是如何把磁盘中所有的空闲盘块都记录起来，又不耗费太多的内存空间。

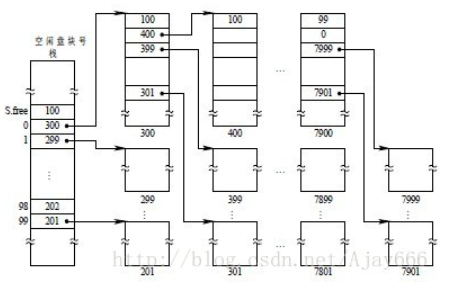


图3.2.1 成组链接法

关于成组链接法，我们可以用一个流程图来表示它的分配过程：

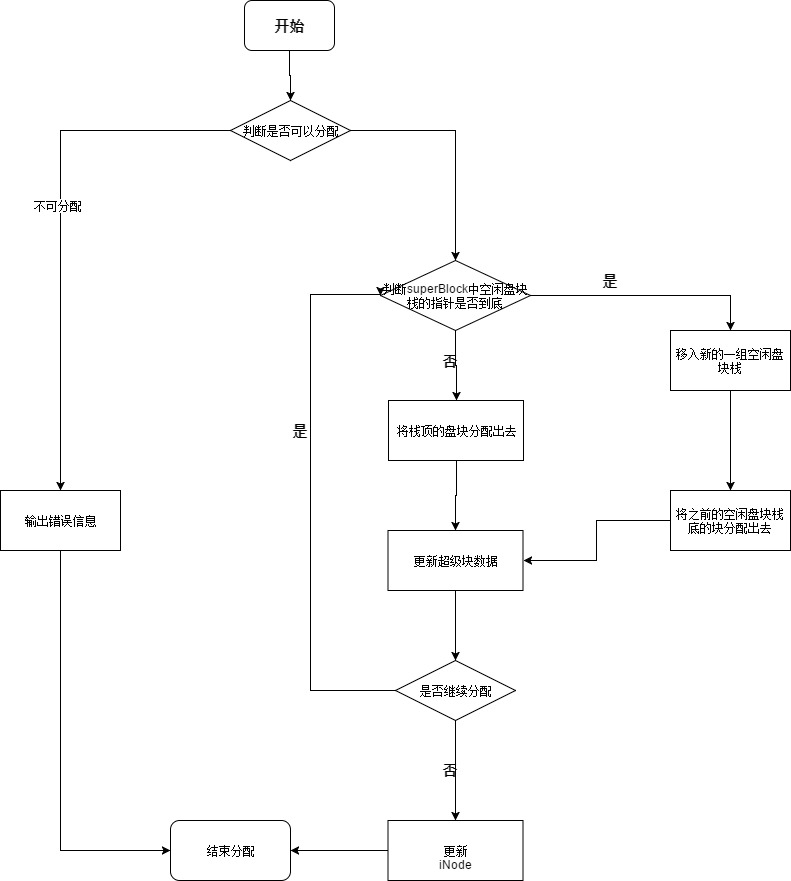


图3.2.2 分配过程

关于成组链接法，我们也可以用一个流程图来表示它的回收过程：

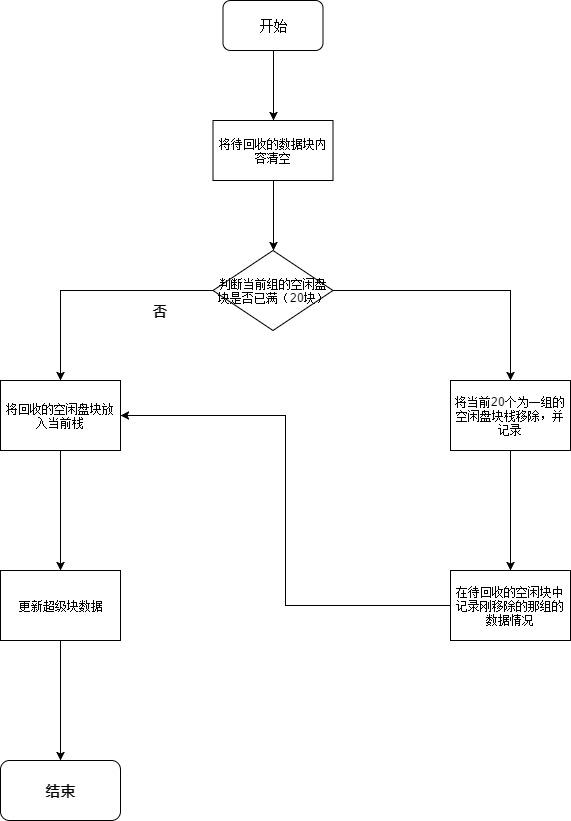


图3.2.3 回收过程

## 3.3文件结构



图3.3.1 文件卷结构

说明：模拟磁盘的大小为200kb(申请时在200kb的位置写入一个“?”，每一个Block的大小为512个字节，一个设计了400个盘块，其中0#号盘块省略，1号盘块为超级块，保存了一些关于INode、Block的信息，2~41号盘块为INode节点，共40个，42号至399号盘块为Block，共400个。)

## 3.4实现命令操作

本系统主要实现了以下命令：

ls 显示目录文件

help 显示命令提示

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

mv 改变文件名

cp 文件拷贝

rm 文件删除

cat 连接显示文件内容

ln 建立文件的硬链接

passwd 修改用户口令（密码）

umask 显示或修改创建文件及目录默认权限码

touch 创建一个文件

>> 文本内容追加

ls -a 显示包括隐藏文件在内的所有文件

sp 输出显示超级块内容

## 3.5部分命令操作的流程图

### 3.5.1 新建文件操作：touch

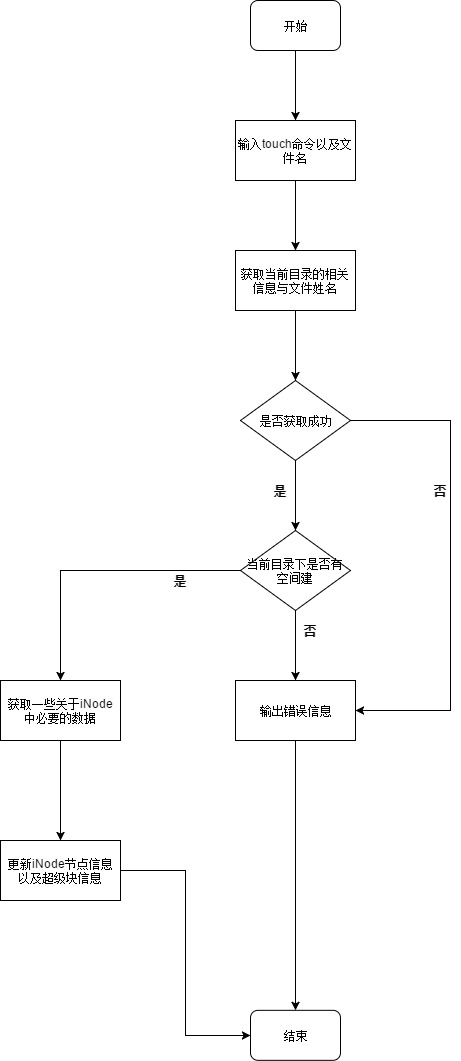


图3.5.1 touch操作过程

### 3.5.2 新建文件操作：mkdir

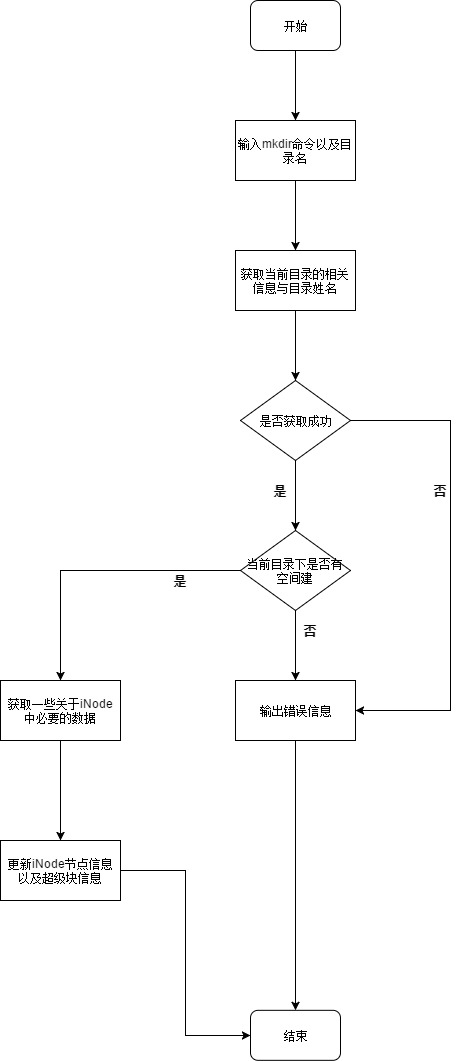


图3.5.2 mkdir操作过程

### 3.5.3 往文件写入操作：>>

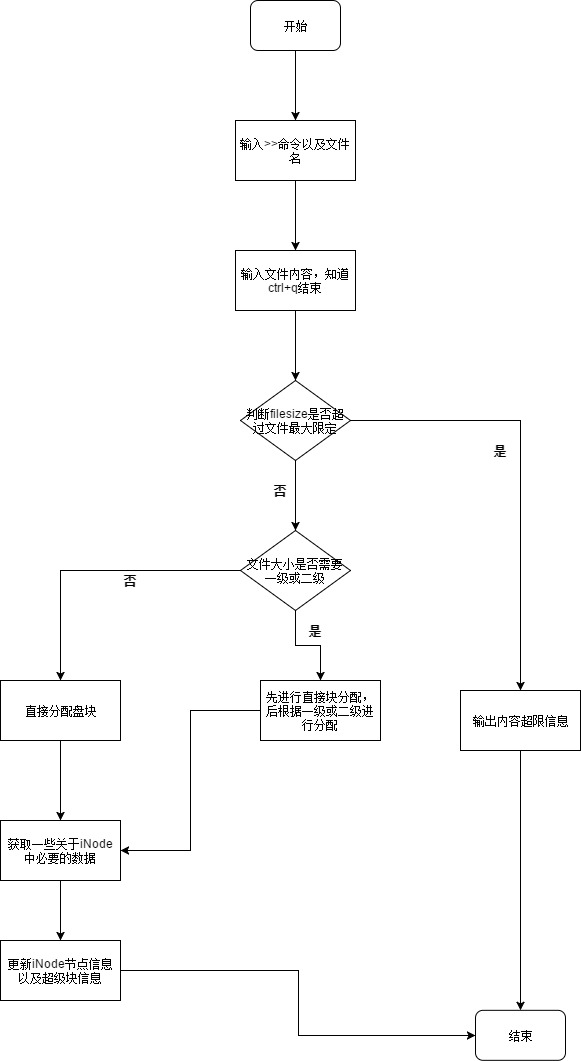


图3.5.3 文件写入操作过程

# 数据结构

## 4.1 超级块

**struct superBlock //超级块**

**{**

**unsigned int Fimod1;//文件权限码**

**unsigned int Fimod2;//文件夹权限码**

**unsigned int size; //磁盘大小**

**unsigned int freeINodeNum; //空闲INode数**

**unsigned int freeINode[INODE\_NUM]; //空闲INode栈**

**unsigned int nextFreeINode; //栈中下一个INode**

**unsigned int freeBlockNum; //空闲盘块数**

**unsigned int freeBlock[BLOCK\_GROUP\_SIZE]; //空闲盘块栈**

**unsigned int nextFreeBlock; //栈中下一个盘块**

**};**

**说明：Fimod1为文件的权限码，Fimod2为文件夹的权限码，刚开始时默认Fimod1为7，也就是rwx，Fimod2为14，也就是rw，以及dir的8（这个在后面详细解释）。空闲的盘块栈以及空闲的INode栈皆用数组的形式来表示，并且通过nextFreeINode和nextFreeBlock的指向来实现栈的弹出和弹入。当nextFreeBlock指向0且用户又申请盘块存储数据时，系统将通过读freeBlock[0]指向的盘块读入下一组盘块刷新freeBlock和nextFreeBlock。**

## 4.2磁盘INode

**struct dINode //磁盘iNode**

**{**

**unsigned int fileSize; //文件大小**

**unsigned int linkNum; //文件的链接数**

**unsigned int addr[6]; //文件地址:四个直接块号，一个一级间址，一个二级间址**

**unsigned short ownerId; //文件拥有者Id**

**unsigned short groupId; //文件所属组Id**

**unsigned int mod; //文件权限和类型**

**long int createTime; //文件创建时间**

**long int modifyTime; //文件最后修改时间**

**long int readTime; //文件最后访问时间**

**};**

**说明：**

**fileSize不仅提供了计算机文件地址的依据，也是控制数据读写的关键参数。**

**LinkNum为实现ln命令提供了方便，链接方和被链接方共享一个INode，链接数也相应增加，进行删除命令删除一个链接文件时，只需使链接数减1并删除在父亲节点留下的目录信息即可，除非链接数为1时才真正删除该文件。**

**add[6]中前四个为直接寻址，剩余的两个分别为一级间址与二级间址。**

**ownerId为权限管理提供了依据。**

**GroupId表示的文件所属组**

**Mod决定了文件的访问权限。**

**剩余的三个Time分别表示的是文件创建的时间、文件最后修改时间、文件最后访问时间。**

## 4.3 内存INode

**struct iNode //内存iNode**

**{**

**dINode dINode;**

**iNode \*parent; //所属的目录i节点**

**unsigned int nodeId; //i节点Id**

**unsigned int users; //引用计数**

**};**

**说明：**

**dINode 表示的是磁盘Inode**

**\*parent提供了cd相对路径的解决办法。**

**Users表示的是引用计数。**

## 4.4 目录项结构

**struct direct //目录项结构**

**{**

**char name[MAX\_NAME\_SIZE]; //文件或目录的名字**

**unsigned short iNodeId; //文件或目录的i节点号**

**};**

**说明：**

**Name表示的是文件或目录的名字**

**iNodeId表示的是文件或目录的i节点号**

## 4.5 目录结构

**struct dir //目录结构**

**{**

**unsigned short dirNum; //目录项数**

**direct direct[MAX\_DIRECT\_NUM]; //目录项数组**

**char padding[DIR\_PADDING\_SIZE]; //目录结构填充字符组**

**};**

**说明：**

**DirNum表示的是目录项数。**

**Padding数组中进行填充使总体占用512个字节方便读写。**

## 4.6 用户结构

**struct owner //用户**

**{**

**unsigned short ownerId; //用户Id**

**unsigned short groupId; //组Id**

**char ownerName[MAX\_NAME\_SIZE]; //用户名**

**char ownerPassword[MAX\_NAME\_SIZE];//用户密码**

**};**

**说明：**

**用户结构里包括了组ID来表示改用户的所属组**

## 4.7用户组结构

**struct owners //用户组**

**{**

**unsigned short ownerNum; //用户项数**

**owner os[MAX\_OWNER\_NUM]; //用户项组**

**};**

**说明：**

**用户组结构中包含了用户项，用数组的形式来保存用户。**

## 4.8 组结构

**struct group //组**

**{**

**unsigned short groupId; //用户组Id**

**char groupName[MAX\_NAME\_SIZE];//组名**

**};**

**说明：**

**组结构中包含了组id与组名**

## 4.9 组集合

**struct groups //组集合**

**{**

**unsigned short groupNum; //组数**

**group gs[MAX\_GROUP\_NUM];**

**};**

**说明：**

**组集合中包含了组数，并利用数组的形式储。**

# 模块详解

## 5.1 文件操作

**1． chown**

**1）命令格式：**

**chown [用户][文件名]**

**2）命令功能：**

**改变文件拥有者。**

**3）命令参数：**

**[用户名]：用户ID**

**[文件名]：文件名称**

**2． chmod**

**1）命令格式：**

**chmod [用户][文件名]**

**2) 命令功能：**

**改变文件所属的权限和类型。**

**3）命令参数：**

**[mod] 文件权限码**

**dir=8,r=4,w=2,x=1**

**#这里的dir指的是目录，r指的是可读、w指的是可写、x指的可执行**

**文件mode**

**若权限为rwx，则4+2+1=7**

**若权限为rw-，则4+2=6**

**若权限为r-x，则4+1=5**

**若权限为r--，则为4**

**若权限为-wx，则为2+1=3**

**若权限为-w-，则为2**

**若权限为--x，则为1**

**目录mode**

**若为rw-，则为8+4+2=14**

**若为r--，则为8+4=12**

**若为-w-，则为8+2=10**

**#目录文件在本修通内除上之外无其他权限模式**

**[文件名]文件名称**

**3． chgrp**

**1）命令格式：**

**chgrp [组][文件名]**

**2）命令功能：**

**改变文件所属组**

**3）命令参数：  
 [组] : 用户组ID**

**[文件名] : 文件名称**

**4． touch**

**1）命令格式：**

**touch [文件名]**

**2）命令功能：**

**创建一个新的文件**

**3）例：touch pwh.txt**

**5． >>**

**1）命令格式：**

**>> [文件名]**

**2）命令功能：**

**向已存在的文件中追加内容。（输入ctrl+q，回车结束输入）**

**3）例：>> pwh.txt**

**Hello world**

**Ctrl + q 结束**

**6． cat**

**1）命令格式：**

**cat [文件名]**

**2）命令功能：**

**连接显示文件内容**

**3）例：cat pwh.txt**

**此时就会输出：Hello world**

**7． cp**

**1）命令格式：**

**cp [源] [目的]**

**2）命令功能：**

**复制源文件到目的文件，若目的文件不存在则创建一个，**

**若存在则目的文件被覆盖**

**复制源目录到目的目录，若目的文件不存在则创建一个，**

**若存在则目的文件被覆盖**

**3）命令参数：**

**[源] ： 源文件或源目录**

**[目的] ： 目的文件或目录**

**8． mv**

**1）命令格式：**

**mv [文件][新文件名]**

**2）命令功能：**

**更改文件的文件名**

**3）例：mv pwh.txt a（此时文件名pwh.txt已更改为a）**

**9． rm**

**1）命令格式：**

**rm [文件]**

**2）命令功能：**

**文件删除**

**10． ln**

**1）命令格式：**

**ln [源][目的]**

**2）命令功能：**

**将源文件链接到目的文件（硬链接的方式）**

**11． umask**

**1）命令格式：**

**第一种：umask**

**第二种：umask [mod][文件或者目录]**

**2）命令功能：**

**第一种功能为显示当前默认的文件或目录的权限码**

**第二种功能为修改当前文件或目录的权限码**

**3）例：umask 12 dir**

## 5.2 目录操作

**1． ls**

**1）命令格式：**

**ls**

**2）命令功能：**

**打印出当前目录清单但不包括隐藏文件。**

**2． ls -a**

**1）命令格式：**

**ls -a**

**2）命令功能：**

**打印出当前目录清单并且包括隐藏文件。**

**3． pwd**

**1）命令格式：**

**pwd**

**2）命令功能：**

**查看当前目录路径。**

**4． cd**

**1）命令格式：**

**cd [目录名]**

**2）命令功能：**

**改变当前目录位置。**

**5． mkdir**

**1）命令格式：**

**mkdir [目录名]**

**2）命令功能：**

**在当前目录下创建子目录。**

**6． rmdir**

**1）命令格式：**

**rmdir [目录名]**

**2）命令功能：**

**删除指定目录和该目录下所有内容。**

**7． ln**

**1）命令格式：**

**ln [源][目的]**

**2）命令功能：**

**将源目录链接到目标目录。**

## 5.3 主函数

|  |  |
| --- | --- |
| **核心的函数表** | |
| **函数名** | **描述** |
| **Void initGlobal(FILE \*f)** | **初始化全局变量、超级块、用户、用户组、和成组链接信息等，且均写入磁盘，当且仅当模拟磁盘文件不存在时调用。** |
| **Void initSystem()** | **初始化文件系统：读入超级块、用户、用户组、当前目录** |
| **Bool readINode(iNode \*r)** | **读取磁盘Inode信息，成功返回true，失败返回false** |
| **Bool writeINode(iNode \*w)** | **写入磁盘INode 信息，成功返回 true，失败返回 false** |
| **bool writeSuperBlock();** | **写入超级块信息，成功返回true，失败返回false** |
| **bool writeDir(unsigned int blockId, dir\* d);** | **写入目录结构，成功返回true，失败返回false** |
| **bool readDir(unsigned int blockId, dir\* d);** | **读取目录结构，成功返回true，失败返回false** |
| **bool readNextBG();** | **读取下一组盘块组信息，成功返回true，失败返回false** |
| **int mkdir(iNode\* parent, char name[MAX\_NAME\_SIZE]);** | **创建目录，返回状态码** |
| **int getFreeBlock();** | **获取空闲盘块，成功返回空闲盘块号，失败返回状态码** |
| **int getFreeINode();** | **获取空闲Inode节点，成功返回空闲Inode节点，失败返回状态码** |
| **int login();** | **登录，返回状态码** |
| **void commandDispatcher();** | **命令分派器** |
| **void displayCommands();** | **显示所有命令** |
| **string pwd();** | **返回当前路径字符串** |
| **string trim(string s);** | **去除字符串两侧空格** |
| **int readCurDir();** | **读取当前目录，返回状态码** |
| **string ls();** | **返回当前目录清单字符串** |
| **string ls2();** | **返回当前目录清单字符串（包括隐藏文件）** |
| **int umask1();** | **显示系统权限码** |
| **int umask1();** | **显示系统权限码** |
| **void changeumask(string name,int n);** | **修改系统权限码** |
| **Bool checkFileName(char name[MAX\_NAME\_SIZE]);** | **检查当前目录下是否存在改名字，存在成功返回true，失败返回false** |
| **int cd(char name[MAX\_NAME\_SIZE]);** | **改变当前目录，返回状态码** |
| **int superMkdir(iNode\* parent, char name[MAX\_NAME\_SIZE], unsigned short ownerId, unsigned short groupId);** | **创建用户目录，返回状态码** |
| **int chmod(char name[MAX\_NAME\_SIZE], unsigned int mod);** | **改变文件权限，返回状态码** |
| **int chown(char name[MAX\_NAME\_SIZE], unsigned short ownerId);** | **改变文件拥有者，返回状态码** |
| **int chgrp(char name[MAX\_NAME\_SIZE], unsigned short groupId);** | **改变文件所属组，返回状态码** |
| **int passwd();** | **更改当前用户密码，返回状态码** |
| **bool writeOS();** | **写入用户组信息，成功返回true，失败返回false** |
| **int mv(char oldName[MAX\_NAME\_SIZE], char newName[MAX\_NAME\_SIZE]);** | **修改文件名，返回状态码** |
| **int touch(iNode\* parent, char name[MAX\_NAME\_SIZE]);** | **创建新文件，返回状态码** |
| **int writeText(iNode\* temp, string text);** | **写入文件内容，返回状态码** |
| **int textAppend(char name[MAX\_NAME\_SIZE]);** | **文件内容追加操作，返回状态码** |
| **int cat(char name[MAX\_NAME\_SIZE]);** | **显示文件内容，返回状态码** |
| **int readText(iNode \*temp);** | **读取文件内容，返回状态码** |
| **int rm(char name[MAX\_NAME\_SIZE]);** | **删除文件，返回状态码** |
| **int returnFreeBlock(unsigned int blockId);** | **回收空闲盘块，返回状态码** |
| **int returnFreeINode(unsigned int iNodeId);** | **回收空闲Inode节点，返回状态码** |
| **int ln(char source[MAX\_NAME\_SIZE], char des[MAX\_NAME\_SIZE]);** | **链接源文件至目标文件，返回状态码** |
| **void rmIter(unsigned short iNodeId);** | **迭代删除目录下所有内容（级联删除）** |
| **int rmdir(char name[MAX\_NAME\_SIZE]);** | **删除目录操作，返回状态码** |
| **int cp(char source[MAX\_NAME\_SIZE], char des[MAX\_NAME\_SIZE]);** | **复制源文件至目标文件，返回状态码** |
| **string getText(iNode\* temp);** | **获取文件内容字符串** |

# 实验演示

## 6.1 登录

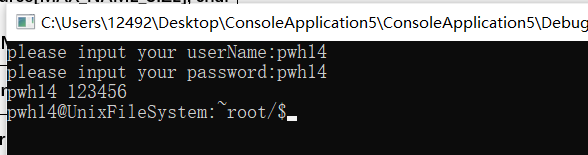


图6.1.1 登陆界面

## 6.2 help命令

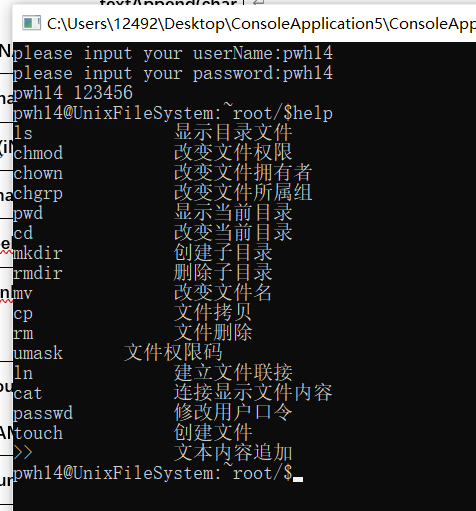


图6.2.1 help命令

## 6.3 ls命令

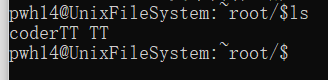


图6.3.1 ls命令

**说明：**

**登录之后我们进入到主命令界面，输入命令ls，我们可以看到有coderTT与TT两个目录项。**

## 6.4 cd命令

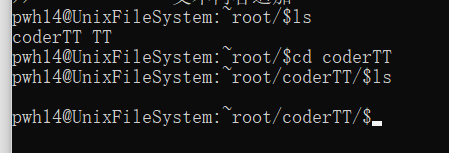


图6.4.1 cd命令

**说明：**

**通过cd命令我们可以改变当前的目录位置，输入cd coderTT，我们就进入到了该目录项下，再输入ls命令，我们可以看到此时并没有任何目录或文件在此目录下。**

## 6.5 sp命令

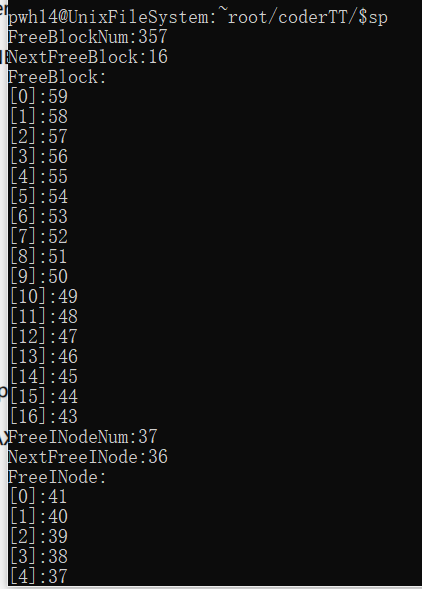


图6.5.1 sp命令(1)

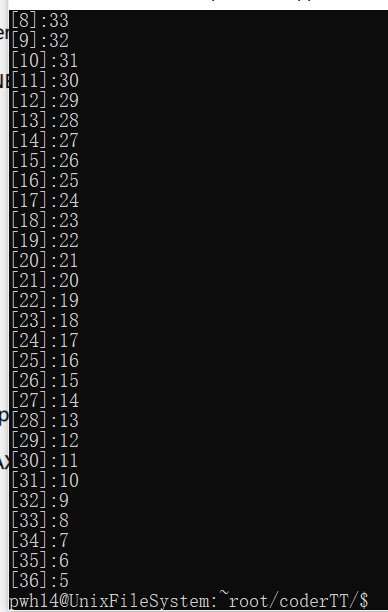


图6.5.2 sp命令(2)

**说明：**

**此时我们输入sp，屏幕上就会打印出超级块中的信息，我们可以看到此时剩余的块数有357个，下一个盘块号为43.**

## 6.6 touch命令

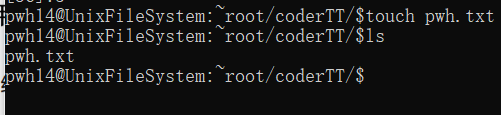


图6.6.1 touch命令

**说明：**

**在coderTT目录下，我们创建一个文件，名字叫做“pwh.txt”，再通过ls命令，我们看到的确这时候已经有一个文件名为pwh.txt的文件在该目录下了。**

## 6.7 >>命令与cat命令

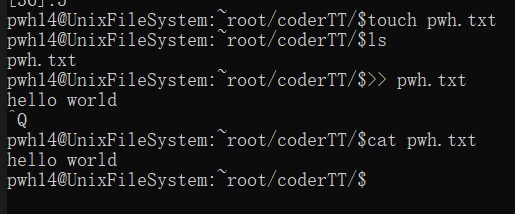


图6.7.1 >>命令与cat命令

**说明：**

**我们可以利用>>命令，向pwh.txt中输入文字，然后通过cat命令将pwh.txt中的信息读取出来。**

## 6.8 cp命令

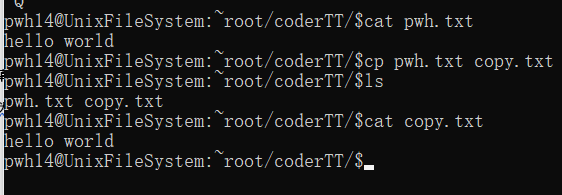


图6.8.1 cp命令

**说明：**

**我们可以利用cp命令，拷贝文件，如图6.8.1，我们可以看到，通过语句cp pwh.txt copy.txt，我们成功将pwh.txt拷贝，因为原本没有copy.txt这个文件存在，因此在这个拷贝过程中，我们先创建一个copy.txt到该目录下，然后复制内容到该文件中。通过ls语句与cat语句，我们可以知道该命令结果是正确的。**

## 6.9 umask命令

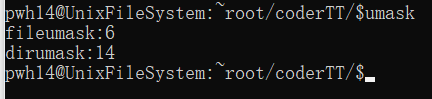


图6.9.1 umask命令

**说明：**

**使用umask可以将当前系统的默认权限码打印在屏幕上。**

## 6.10 mv命令

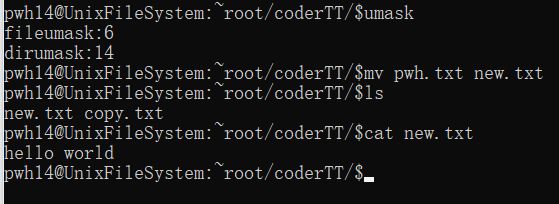


图6.10.1 mv命令

**说明：**

**使用mv命令将pwh.txt文件的名称改为new.txt，然后通过ls命令与cat命令验证了其正确性。**

## 6.11 mkdir命令

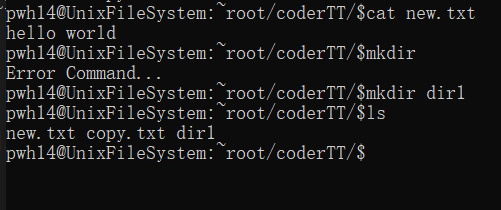


图6.11.1 mkdir命令

**说明：**

**使用mkdir命令将在当前目录下创建出一个新的目录，输入命令mkdir dir1，然后输入ls，可以看到当前目录下已经创建出一个名为dir1的新目录。**

## 6.12 在新建的目录中使用cd与touch命令

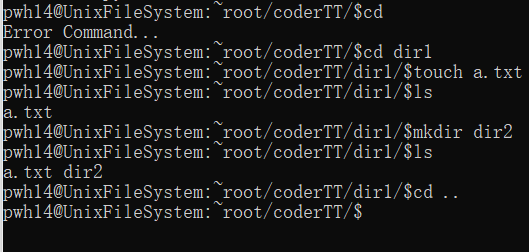


图6.12.1在新建的目录中使用cd与touch命令

**说明：**

**这一步里，我们进入到新创建的dir1目录中，然后创建出一个新的目录dir2与一个新的文件a.txt。通过ls判断该操作的正确性，最后通过cd ..回到上一级目录。**

## 6.13 rmdir命令

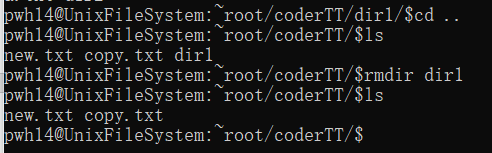


图6.13.1 rmdir命令

**说明：**

**回到coderTT目录下，我们删除dir1目录，删除的方式是级联删除，通过ls查看发现，操作正确。**

## 6.14 ln命令

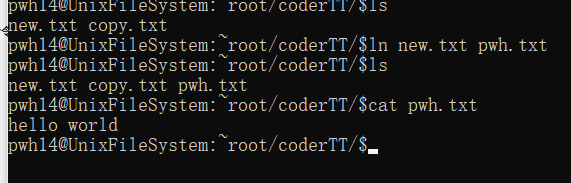


图6.14.1 ln命令

**说明：**

**使用ln命令，对文件进行硬链接操作，输入命令ln new.txt pwh.txt，此时已经对new.txt建立了硬链接，通过ls，cat命令判断操作的正确性。**

## 6.15 pwd命令

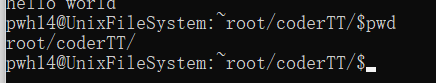


图6.15.1 pwd命令

**说明：**

**使用pwd命令，当前所在目录的路径被打印在了屏幕上。**

## 6.16 关于隐藏文件现实的命令

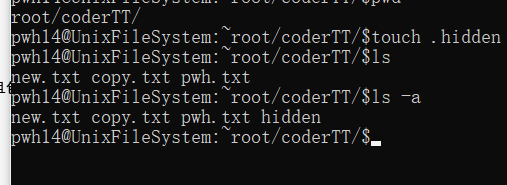


图6.16.1 ls -a命令

**说明：**

**隐藏文件在UNIX文件系统中一般是不显示的，我们可以通过命令ls -a将所有的文件显示出来，包括隐藏文件。如图6.16.1，我们先创建一个隐藏文件（开头为.的文件就为隐藏文件），然后分别输入ls与ls -a两个指令，我们可以对比发现，隐藏文件只在ls -a命令中出现。**

## 6.17 输入的文件大小需要间址的情况

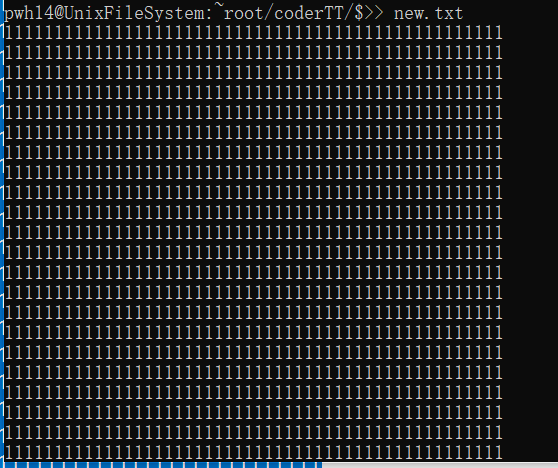


图6.17.1 >>命令（1）

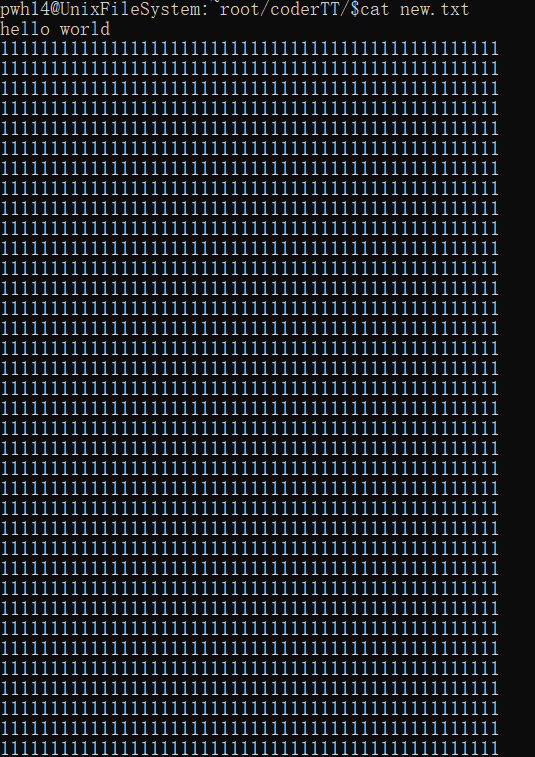


图6.17.2 >>命令（2）

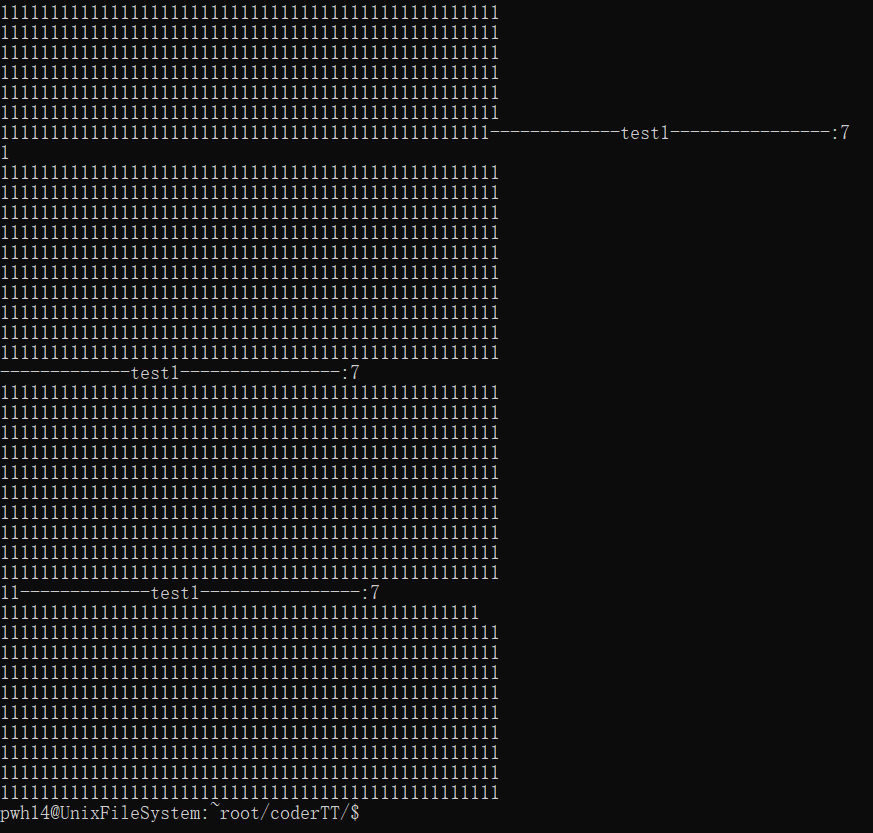


图6.17.3 >>命令（3）

**说明：**

**当输入的文件超过2048B时，此时就需要用到一级间址，如图6.17.3，可以知道此时文件内容已经到第七块，用到了一级间址。正确**

# 实验总结

## 7.1实验过程遇到的问题

1. 关于一些字符串的操作不是很了解，例如输入的命令是一串字符串，那么如何分开读取时一个很重要的字符串处理问题，通过查询并实际测试过后较为顺利地解决了这个问题，但是后面又碰到怎么将分开获取出来的字符串形式的数字转化为int类型又伤到了脑筋，后来查询得知，可以利用atoi这个函数解决。遂问题解决。
2. 在最开始做这个系统的时候遇到了很多问题，比如对文件系统本身的不熟悉，文件卷的结构设计问题，空闲盘块的回收与分配问题，以及间址问题等等。但在接近半个月的资料翻阅，对整体系统有了进一步了解之后，系统的编写就稍微地顺利了一些，但是还是卡在了数据结构上设计，写了多个版本，每次写着写着就发现一些功能没法在当前地结构上实现，后询问了身边的人才有了进一步的改进。
3. 设计间址的时候出现了一些细节上的错误，比如计算的时候只因为极小的差错就导致了最终的结果出现了大差错。经过多次计算后才得以正确。
4. 最终的版本有一个问题，在cd相对路径实现上，UNIX的文件系统是采用在创建目录一开始就生成“.”以及“..”两个隐藏文件，分别表示当前目录以及上一级目录。但是我采用的是用parent指针来回到上一级。在实现上有一定的差错。虽然在目前的数据结构上，修改也比较容易进行，但由于时间比较紧张，遂最终提交没有修改，计划回家继续完善该模拟UNIX版本。

## 7.2实验体验总结

本次实验最主要的是对于UNIX文件系统的理解更加深刻了一些，UNIX文件系统作为一个较完善的文件系统，有很多地方都值得我们去理解学习，例如成组链接法，位示图法等等。

这次实验老师也帮助了很多，例如在一些命令的实现上给了我很多思路，最后验收的时候老师对于我对原先间址问题的解决办法又有了更好的解决方案让我开扩了思维。在这点上，整个文件系统的编写让我对于C++在一些字符串、字符数组的处理上有了进一步的理解。例如cat出现乱码问题是因为从.dat文件中获取换块输出时，最后的判断是靠‘\0’来判断的（老师提出采用‘EOF’的方式更好）等等。

总而言之，操作系统是一门偏向于实际操作的课程，整个系统的编写让我对于文件系统的认识更加深刻了。