《操作系统》课程设计实验报告

类 UNIX 文件系统

学　　号 1953072

姓　　名 肖鹏飞

专　　业 计算机科学与技术

日　　期 2022.5.31

一．实验概述

1. 实验目的

作为操作系统课程设计的组成部分，构造一个类 UNIX 二级文件系统。

2. 实验内容

构造一个类 UNIX 二级文件系统，使用一个普通的大文件（如 c:\myDisk.img ，称之为一级文件） 来模拟 UNIX V6++的一个文件卷（把一个大文件当一张磁盘用）。

2.1 磁盘文件结构：

- 定义自己的磁盘文件结构
- SuperBlock 结构
- 磁盘 Inode 节点结构，包括：索引结构
- 磁盘 Inode 节点的分配与回收算法设计与实现
- 文件数据区的分配与回收算法设计与实现

2.2 文件目录结构：

- 目录文件结构
- 目录检索算法的设计与实现

2.3 文件打开结构

2.4 磁盘高速缓存：选作

2.5 文件操作接口：

- fformat： 格式化文件卷
- ls： 列目录
- mkdir： 创建目录
- fcreat： 新建文件
- fopen： 打开文件
- fclose： 关闭文件
- fread： 读文件
- fwrite： 写文件
- flseek： 定位文件读写指针
- fdelete：删除文件
- 。。。

2.6 主程序：

- 格式化文件卷；

装 订 线

- 用 mkdir 命令创建子目录，建立如图所示的目录结构；
- 把你的课设报告，关于课程设计报告的 ReadMe.txt 和一张图片存进这个文件系统，分别放在/home/texts ，/home/reports 和/home/photos 文件夹；
- 图形界面或者命令行方式，等待用户输入；
- 根据用户不同的输入，返回结果。

2.7 通过命令行方式测试：

- 新建文件/test/Jerry，打开该文件，任意写入 800 个字节；
- 将文件读写指针定位到第 500 字节，读出 500 个字节到字符串 abc。
- 将 abc 写回文件。

3. 实验报告内容

需求分析（10%）：说明程序任务，包括：输入、输出形式，程序功能。

概要设计（10%）：任务分解；数据结构定义；模块间的调用关系，算法说明等。

详细设计（30%）：重点函数的重点变量需说明，重点功能部分要绘制清晰的程序流程图。画出函数调用关系。

运行结果分析（35%）：

程序运行结果展示说明；

测试命令及输出结果，结果分析。

用户使用说明（5%）

答辩（10%）：PPT 介绍设计内容，现场回答老师问题。

4. 实验环境

操作系统：Windows 10 1903

开发语言：c++

开发环境及具体版本：Visual Studio 2019 (16.0)

装

订

线

二. 需求分析

1. 输入形式

在本文件系统中，使用控制台的命令行形式来实现用户与文件系统的具体交互，构建了一个用于交互的 shell 界面。具体支持可在 shell 界面输入的指令及输入格式如下：

1.1 基本目录相关指令

支持指令	指令格式	指令功能
help		提示信息
cd	<文件路径>	切换当前目录至指定路径

ls			展示当前目录下内容
mkdir	<文件路径>		在指定路径下创建目录
rm	<文件路径>		删除指定路径下的目录或文件
chmod	<权限>	<文件路径>	修改目录或文件的权限

1.2 文件读写相关指令

支持指令	指令格式		指令功能
create	<文件路径>		在指定路径下创建文件
open	<文件路径>		打开指定文件
close	<文件路径>		关闭指定文件
cat	<文件路径>		打印指定文件所有内容
read	<文件路径>	<读长度>	读文件
write	<文件路径>	<写内容>	写文件
lseek	<文件路径>	<新的文件指针位置>	移动文件指针
writein	<文件路径>	<外部文件路径>	从外部导入文件进入文件系统
writeout	<文件路径>	<外部文件路径>	从文件系统向外部导出文件
opened			查看打开文件信息

1.3 多用户管理相关指令

支持指令	指令格式		指令功能
adduser	<用户名>	<用户密码>	增加用户
deluser	<用户名>		删除用户
addusertogroup	<用户名>	<用户组名>	增加用户进用户组
deluserfromgroup	<用户名>	<用户组名>	从用户组中删除用户
whoami			打印当前用户
showalluser			打印所有用户信息
updatepassword	<用户名>		修改用户密码
su	<用户名>		切换当前登录用户
exit			退出文件系统

1.4 其他非指令操作

除了以上 1.1-1.3 中所列出指令以外，本系统还有以下额外功能：

对文件卷的格式化：只有在每次重新打开文件系统时才能够选择格式化文件。

```
是否重新构建磁盘？（y：重新构建，n：不重新构建）：n
```

登录：在刚进入系统，尚未登录的状态下，输入用户名和密码进行登录。

```
登录
用户名: root
密码: root123
欢迎登录，可以通过输入help查看本文件系统支持的指令
[root ~]#help
```

2. 输出形式

在构建的 shell 界面中，通过命令行交互的方式输出文件内容或者相关操作的提示信息，具体的各项提示信息在（五）中有详细展示。

```
[root ~]#help
支持指令          指令格式                              指令功能
help                                                    提示信息
cd                <文件路径>                            切换当前目录至指定路径
ls                                                      展示当前目录下内容
mkdir             <文件路径>                            在指定路径下创建目录
rm                <文件路径>                            删除指定路径下的目录或文件
chmod             <权限>      <文件路径>                修改目录或文件的权限
create            <文件路径>                            在指定路径下创建文件
open              <文件路径>                            打开指定文件
close             <文件路径>                            关闭指定文件
cat               <文件路径>                            打印指定文件所有内容
read              <文件路径> <读长度>                   读文件
write             <文件路径> <写内容>                   写文件
lseek             <文件路径> <新的文件指针位置>         移动文件指针
writein           <文件路径> <外部文件路径>             从外部导入文件进入文件系统
writeout          <文件路径> <外部文件路径>             从文件系统向外部导出文件
opened                                                  查看打开文件信息
adduser           <用户名>   <用户密码>                 增加用户
deluser           <用户名>                              删除用户
addusertogroup    <用户名>   <用户组名>                 增加用户进用户组
deluserfromgroup  <用户名>   <用户组名>                 从用户组中删除用户
whoami                                                  打印当前用户
showalluser                                             打印所有用户信息
updatepassword    <用户名>                              修改用户密码
su                <用户名>                              切换当前登录用户
exit                                                    退出文件系统
```

3. 程序功能

在本文件系统中，实现了类 UNIX 文件系统相关的部分功能，具体列表如下：

3.1 系统设置与交互功能

文件系统的格式化与初始化

文件系统的登录与退出

Shell 界面的实现与指令的读取分析

打印系统说明文档

3.2 目录相关基础功能
目录的创建与删除
进入目录和展示目录内容
目录的权限修改

3.3 文件相关功能
文件的创建与删除
文件在内存中的打开与关闭
文件系统与外部的交互
文件的读写与文件指针的操作
文件的权限修改与操作权限检测

3.4 用户相关功能
用户的创建和删除
用户的登录和退出
修改用户密码
展示当前用户身份和系统中用户列表
将用户加入用户组
从用户组中删除用户

装 订 线

三. 概要设计

1.任务分解
在本文件系统的整体模块设计中，划分为以下几个功能模块：
顶层文件系统模块：作为所有模块中的最顶层模块，直接提供用户与文件系统进行交互的接口。具体功能包括构建 shell 界面，创建新的文件系统、加载已有文件系统、处理用户输入指令并返回交互信息等。

内存管理模块：负责管理本次登录时的内存，在每次登录文件系统中时从磁盘的对应位置载入。需要注意的是，内存的生命周期仅限于本次登录期间。具体功能有将内存中修改后数据同步到磁盘、对物理块和逻辑块之间的映射转换、对内存中登录的用户、打开的文件等进行实时更新等。

磁盘管理模块：对 IMG 磁盘文件进行管理，以 512 字节大小的 block 作为管理的基本单位，需要维护磁盘中的 superblock、inode、数据区 block 等。具体功能有内存中修改后数据同步到磁盘、对磁盘 inode 和 block 块的分配与释放、路径与 inode 块的转换等。

目录管理模块：对目录结构进行管理，在本系统中没有使用额外的目录区块存储，而是将其结构作为文件，直接存储在数据区 block 中。具体功能有查看当前目录信息、创建目录、删除目录、修改目录权限、进入目录等。

文件管理模块：对系统中文件进行管理，主要负责管理除目录以外的文本文件。具体功能有打开/关闭文件、读文件、写文件、移动文件指针、修改文件权限、展示系统中打开文件、从系统外部加载文件等。

多用户管理模块：对系统中的所有用户进行统一管理，用户信息存储在磁盘 block 一个特定区域，在每次登录时维护一个当前登录的用户信息。具体功能有创建用户、删除用户、用户的登录和退出、展示用户信息、管理用户组、修改用户密码等。

2.数据结构定义

顶层文件系统类：

```
/*系统整体，注意使用单例模式！！！！！！*/
// 局部静态变量实现的懒汉式单例，C++11 后线程安全
/*C++11 规定了 local static 在多线程条件下的初始化行为，要求编译器保证了内部静态变量的线程安全性。在 C++11 标准下，《Effective C++》提出了一种更优雅的单例模式实现，使用函数内的 local static 对象。这样，只有当第一次访问 getInstance()方法时才创建实例。这种方法也被称为 Meyers’ Singleton。*/
class FileSystem
{
private:
	FileSystem() { };
	~FileSystem() { };
	FileSystem(const FileSystem&);
	FileSystem& operator=(const FileSystem&);

	/*以下为文件系统自定义内容，与单例无关*/
	UserInfo user_now;//当前系统中登录的用户

	/*指令，实现指令集的目的是为了实现指令增减的自动化*/
	/*通过函数指针的方式，绑定一组函数与 string 的对应关系*/
	unordered_map<string, int(FileSystem::*)(vector<string>)> inst; //指令集
	SuperBlock my_superblock;                                        //内存 superblock

	/*打开文件相关*/
	int inode_open_num;                              //已经打开的 inode 数(只有文件)
	Inode my_inode_map[MAX_OPEN_FILE_NUM];    //内存 inode 数组
	File my_file_map[MAX_OPEN_FILE_NUM];      //打开的文件结构

	/*路径相关*/
```

```
	Directory now_dir;		//当前位于的 direction

public:
	static FileSystem& getInstance()//单例模式唯一外界可用——获取实例
	{
		static FileSystem instance;
		return instance;
	}
};
```

BlockGroup 类，成组链接法过程中使用：

```
//成组链接的 block 块
class BlockGroup {
public:
	int s_nfree;
	int s_free[BLOCK_FREE_NUM];
};
```

Superblock 类：

```
/*本项目尝试构建的文件系统是：
superblock——inode——文件数据区的三段存储层次结构*/
/*superblock：占用 0、1 两个 block*/
class SuperBlock {
public:
	/*inode 为文件总数的限制，处于简化考虑，不对其使用成组链接法，写死为 100*/
	int disk_block_size;//磁盘的 block 总数
	int s_isize;			//inode 占用的 block 盘块数
	int s_ninode;			//直接管理的空闲 inode 数
	int s_inode[INODE_NUM]; //管理的 inode 栈，里面存编号

	/*实际 block：数量较多，对其使用成组链接法*/
	int s_fsize;			//数据区使用的 block 总数
	int s_nfree;			//直接管理的空闲盘块数
	int s_free[BLOCK_FREE_NUM]; //直接管理的空闲盘块索引表

	/*其他相关项*/
	int s_fmod;		//内存中 superblock 被修改，写回
	int s_ronly;		//文件系统只能读出
	time_t s_time;	//最近一次更新时间
	int padding[66];//填充，使 superblock 大小为 1024
};
```

Inode 类：

```
/*inode，占用 10 个 block。单个 inode 结构大小为 64，1 个 block 能放 8 个 inode，系统共 80
个可用 inode，即可创建文件上限为 80*/
```

```
/*一个 inode 代表一个文件*/
/*inode 目前考虑在 superblock 中使用位示图进行管理*/
class Inode {
public:
	/*10 位有效位，drwxrwxrwx*/
	enum INODEMODE {
		OTHER_x = 0x1,
		OTHER_w = 0x2,
		OTHER_r = 0x4,
		GROUP_x = 0x8,
		GROUP_w = 0x10,
		GROUP_r = 0x20,
		OWNER_x = 0x40,
		OWNER_w = 0x80,
		OWNER_r = 0x100,
		DIRECTORY = 0x200
	};

	unsigned int i_mode;		//enum INODEMODE 中定义的文件权限(暂定为 10 位)
	int i_nlink;				//文件在不同目录树中的索引数量
	int i_uid;					//文件所有者用户 id
	char usergroup[12];			//用户组名，只有一个主用户组
	int i_size;					//文件大小（字节数）
	int i_addr[10];				//逻辑块号和物理块号转换索引表
	int i_no;					//文件的 inode 序号(便于内存中 inode 写回磁盘查找位置)
	time_t	i_atime;			//最后访问时间
	time_t	i_mtime;			//最后修改时间
	int padding[9];				//填充，使 inode 节点大小为 128
};
```

File 类，打开文件：

```
/*是文件，block 数据区中存的是文件内容
是目录，block 数据区中存的是 directory 结构 */
/*文件的打开结构，一个 file 对应着某个用户的一次打开
注意，一个用户最多只能打开一个相同文件。在打开文件之前首先查询
使用一个内存中的文件管理表来进行管理
单次打开 cmd 相当于开机，关闭 cmd=关机*/
class File {
public:
	string file_name;			//文件名
	string user_name;			//用户名
	int inode_num;				//文件对应的 inode 号
	int offset;					//文件指针（偏移量位置），每次打开后被重置
	int openuser_id;			//打开该文件的用户 id
};
```

DirectoryEntry 类，目录项：

```
/*注意，需要写在 block 中的内容不能使用 string，因为大小不确定。向 block 中写东西是非常底层的内存操作，而 string 是一个很高级语言的封装类，两者显然会产生冲突*/
/*目录项：可能是目录，也可能是文件，取决于 inode 中 mode 设置
DIRSIZ=28，DirectoryEntry=32*/
class DirectoryEntry {
public:
	int n_ino;				//目录项中 Inode 编号部分
	char m_name[MAX_NAMELEN];//目录项中文件名部分
};
```

Directory 类，目录：

```
/*目录，存目录项，有上限（暂时未实现实时扩充）
MAX_SUBDIR=16，一个 Directory 为 32*16=512 字节，占一个 block
存储是摊平的，目录是树形的，因此需要对两者做一个转换*/
class Directory {
public:
	DirectoryEntry child_filedir[MAX_SUBDIR];	//子目录/文件
	char m_name[MAX_NAMELEN];			//改进，在目录中写自己的文件名
	int d_inode;						//填充
};
```

UserInfo 类，单个用户信息：

```
/*用户，数据存在一个单独的文件里*/
/*5 个属性，自由设置*/
class UserInfo {


public:
	int id;				//用户序号，0 为 root
	char name[12];		//用户名
	char usergroup[12]; //用户组名，每个用户可以有很多个用户组，但只有一个主用户组
	char password[16];	//注意是 md5 算法加密后的用户密码，private 封装,MD5 最长为 128bit=16byte

	const char * Func_Encryption(const char password1[12]);//存储密码：将明文密码加密为密文

};
```

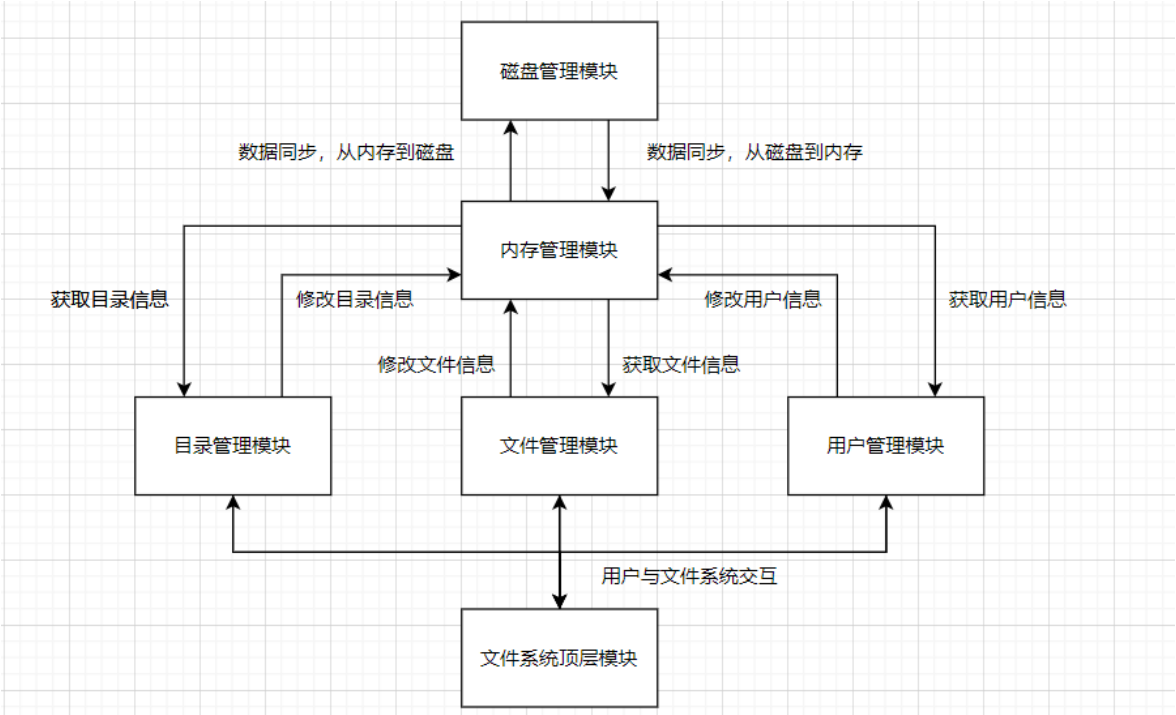
Users 类，所有用户信息：

```
/*系统中用户的所有信息，存在某个 block 中，不在内存中常态存在*/
class Users {
public:
	int user_num;			//系统存储的用户总数（上限为 10，一个 block 中）
	int user_now;			//当前用户累计数（分配给下一个用户的 uid 号）
```

```
    UserInfo user[MAX_USER_NUM];//存储用户信息
    int padding[16];                //填充字节
};
```

3.模块间的调用关系

模块间具体的调用关系如下：



4.算法说明

以下为本系统实现的几个较为典型的算法，具体实现细节及相关流程图在（四、详细设计）中有说明。

4.1 逻辑块与物理块的地址映射转换算法

在文件的 Inode 结构中，有 int i_addr[10]数组用于从文件逻辑块号查找物理块号。其中 i_addr[0:5]为直接索引，i_addr[6:7]为一级索引，i_addr[8:9]为二级索引。具体查找方法如下：

4.1.1 逻辑块号 < 6：直接索引，查找 i_addr[0:5]获取对应的物理块号。

4.1.2 6 <= 逻辑块号 < 6+128*2：一级索引，先查找 i_addr[6:7]获取一级索引块 block 号，再根据一级索引块 block 查找(logic_num-6)%128 位置，此处为对应的物理块号。

4.1.3 6+128*2 <= 逻辑块号 < 6+128*2+128+128*2：二级索引，先查找 i_addr[8:9]获取二级索引块 block 号，再根据二级索引块 block 查找(logic_num-6+128*2)/128 位置，获取一级索引块 block 号。再根据一级索引块 block 查找(logic_num-6+128*2)%128 位置，此处为对应的物理块号。

分配和释放的算法与查找类似，使用同样的算法映射，此处不再详细说明。

4.2 block 数据块成组链接法

对于数据块 block 来说，在 superblock 中只有 100 个块直接管理，绝大多数 block 数据块是通过成组链接法进行管理的。每 100 个数据块作为一组，链接块数据结构在 BlockGroup 中，s_nfree 为当前连接块管理的空白块数目，s_free[100]为管理的数组，其中[0]号位置指向下一个链接块，[1-99]指向普通的数据块，只有[0:s_nfree-1]为有效的空白块。

4.2.1 空白块的分配

在分配空白块时，查看当前 superblock 中 s_nfree 是否为 1，若为 1，则将[0]指向的链接块读入 superblock 中，同时将最后一个连接块分配出去；否则将 s_free[--s_nfree]分配出去。

4.2.2 空白块的释放

在释放空白块时，查看当前 superblock 中 s_nfree 是否为 100，若为 100，则将原来的链接块移出 superblock，在 superblock 中创建一个新的链接块，将[0]指向刚刚移出的链接块，同时将最后一个链接块分配出去，将 s_nfree 设为 1；否则在 s_free[s_nfree++]位置释放，填入释放块号。

4.3 根据路径查找对应文件/目录算法

在输入某个路径时，假设该路径合法。首先判断路径是绝对路径还是相对路径，若为绝对路径，则起始目录为根目录，若为相对路径，则起始目录为当前目录。之后设置 vector<string> 来存储每一级目录名，以/为分隔符对一长串目录进行分解。当目录分解完成后依次按名称查询 vector 中对应的目录 inode，查看其子目录是否有下一个，若有则进入下一级目录，若没有则返回查找失败。

4.4 文件操作权限检测算法

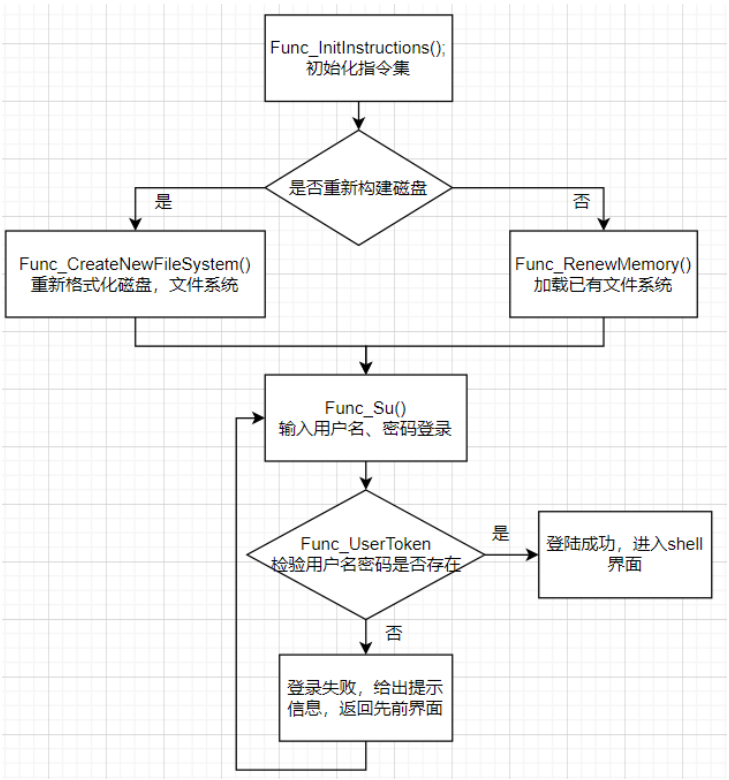
在对文件进行读写等操作时，需要当前在系统中登陆的用户拥有对应的权限才能进行操作。因此对所有的文件相关操作，在实际执行操作之前进行统一的权限检测。给出对应的 10bit 文件权限位（drwxrwxrwx）作为检测信息，首先检测当前用户是否为文件所有者，若不是再检测当前用户是否为所有者同一组用户，若不是则为 OTHER。之后根据读/写的不同需求，检测对应的权限位。

四．详细设计

1.文件系统顶层模块

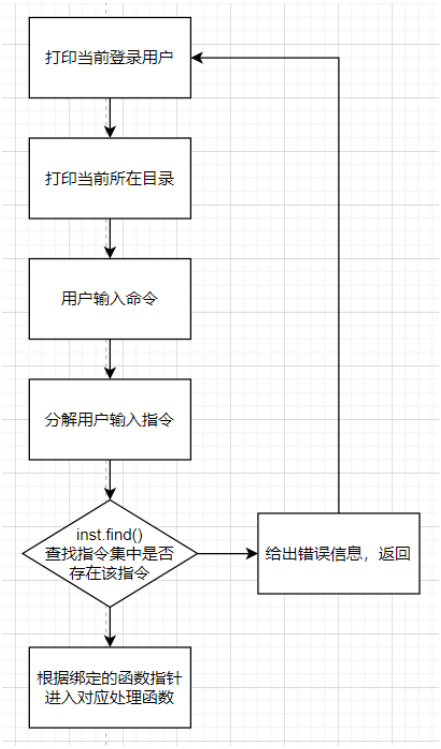
1.1 登录循环 Func_LoginLoop()

启动系统的登录过程，先选择是否构建新文件系统，之后循环输入用户名和密码进行登录。



1.2 shell 界面实现 Func_ShellLoop

实现了系统的 shell 界面，界面功能包括展示当前登录的用户名和所在目录，分析用户输入的指令并转到对应的调用函数等。需要注意的是，在根据指令选择调用函数的过程中，使用了一组依赖于本文件系统类的函数指针，指令集在 Func_InitInstructions()中初始化。



2. 磁盘与内存管理模块

由于磁盘与内存作为二级文件系统的基础层级，两者之间有着密切的数据转换关系，因此在一起说明。对内存与磁盘的管理提供了一系列重要的组件函数，是本文件系统最基础也是最重要组成部分。

2.1 内存与磁盘的数据同步

2.1.1 磁盘数据同步到内存 Func_RenewMemory()

在每次需要从磁盘读取数据到内存是需要使用该函数，使用场景为对初始化刚登陆的文件系统或修改磁盘数据后需要执行该函数。

2.1.2 内存数据同步到磁盘 Func_RenewDisk()

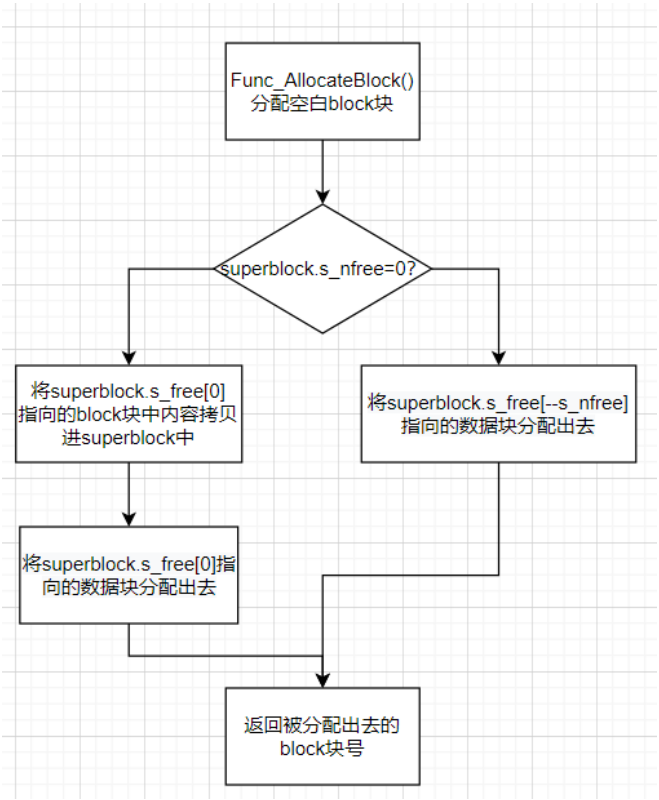
在每次对内存中数据操作完成后，将内存中数据同步到磁盘持久性保存。例如在分配 block 和 inode 块、写入文件、修改用户信息等操作最后需要执行该函数。

2.2 磁盘 block 数据块的分配与释放

对于数据块 block 来说，在 superblock 中只有 100 个块直接管理，绝大多数 block 数据块是通过成组链接法进行管理的。每 100 个数据块作为一组，链接块数据结构在 BlockGroup 中，s_nfree 为当前连接块管理的空白块数目，s_free[100]为管理的数组，其中[0]号位置指向下一个链接块，[1-99]指向普通的数据块，只有[0:s_nfree-1]为有效的空白块。

2.2.1 空白块的分配 int Func_AllocateBlock()

在分配空白块时，查看当前 superblock 中 s_nfree 是否为 1，若为 1，则将[0]指向的链接块读入 superblock 中，同时将最后一个连接块分配出去；否则将 s_free[--s_nfree]分配出去。



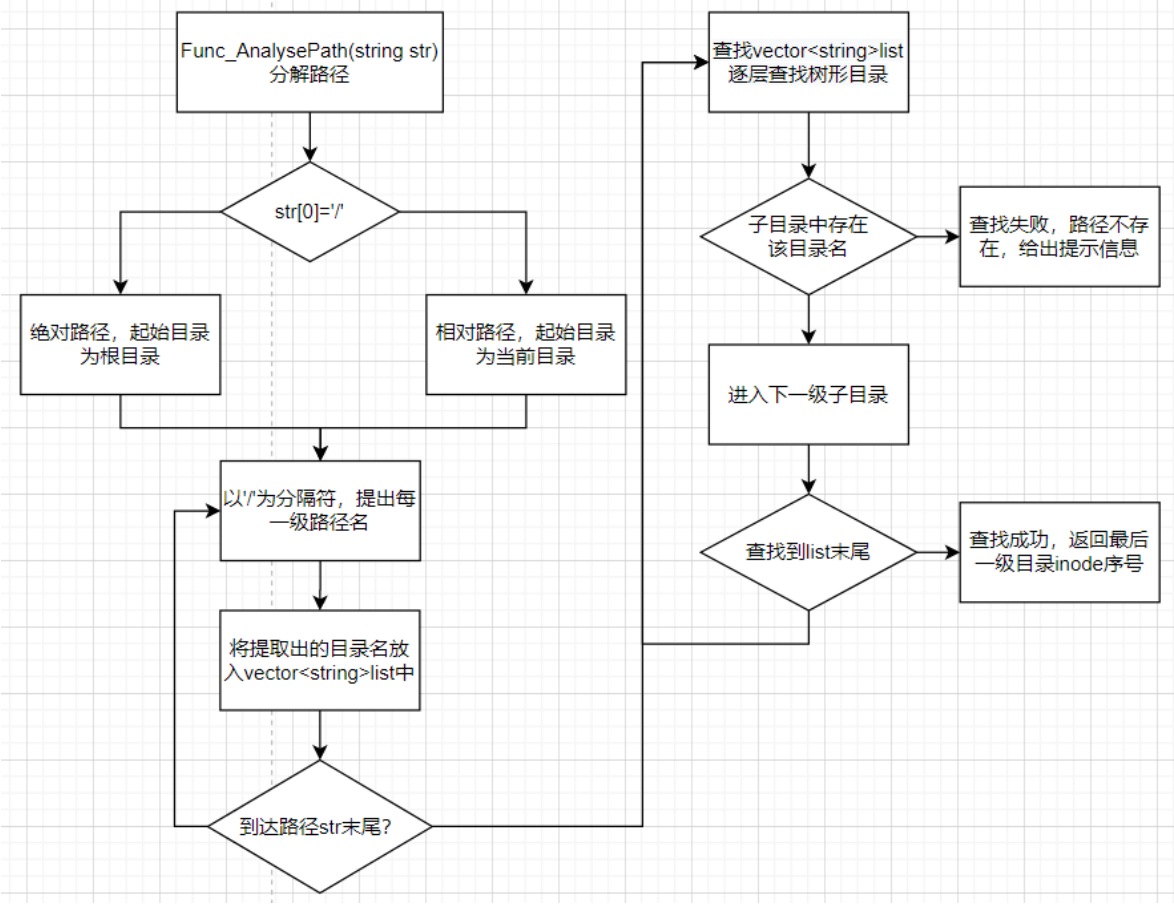
2.2.2 空白块的释放 int Func_ReleaseBlock(int block_num)

在释放空白块时，查看当前 superblock 中 s_nfree 是否为 100，若为 100，则将原来的链接块移出 superblock，在 superblock 中创建一个新的链接块，将[0]指向刚刚移出的链接块，同时将最后一个链接块分配出去，将 s_nfree 设为 1；否则在 s_free[s_nfree++]位置释放，填入释放块号。

释放过程的函数流程图为分配过程的逆向，此处不再给出。

2.3 路径与 inode 块的转换 int Func_AnalysePath(string str)

在输入某个路径时，假设该路径合法。首先判断路径是绝对路径还是相对路径，若为绝对路径，则起始目录为根目录，若为相对路径，则起始目录为当前目录。之后设置 vector<string> 来存储每一级目录名，以/为分隔符对一长串目录进行分解。当目录分解完成后依次按名称查询 vector 中对应的目录 inode，查看其子目录是否有下一个，若有则进入下一级目录，若没有则返回查找失败。



2.4 物理块和逻辑块之间的映射转换 int Func_AddrMapping(int inode, int logic_addr);

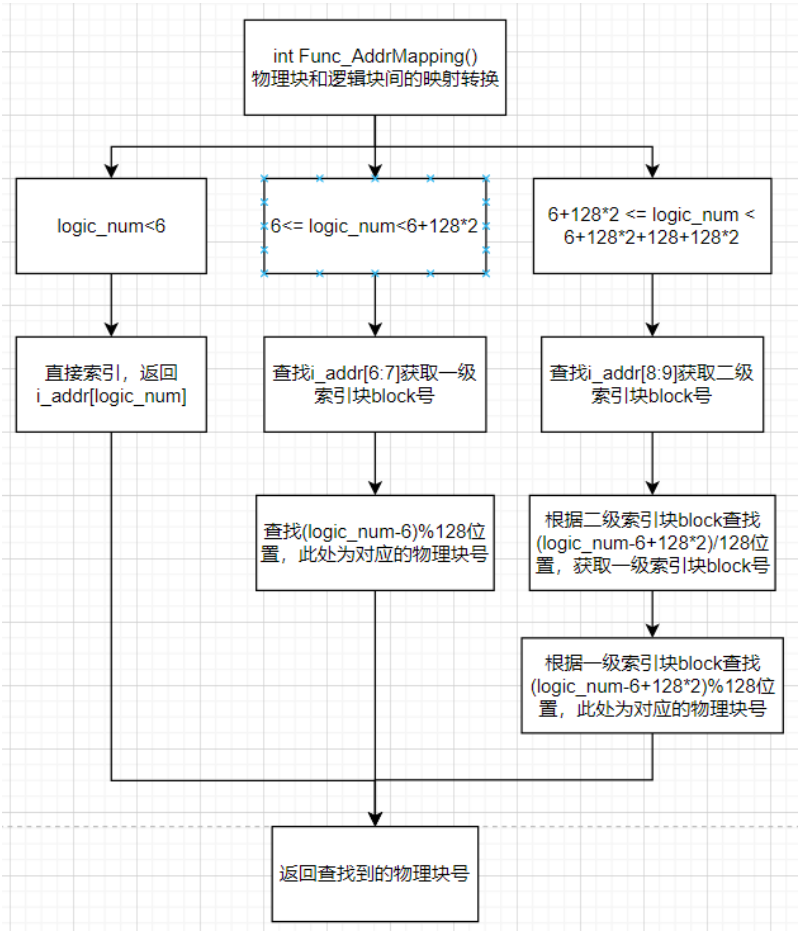
在文件的 Inode 结构中，有 int i_addr[10]数组用于从文件逻辑块号查找物理块号。其中 i_addr[0:5]为直接索引，i_addr[6:7]为一级索引，i_addr[8:9]为二级索引。具体查找方法如下：

2.4.1 逻辑块号 < 6：直接索引，查找 i_addr[0:5]获取对应的物理块号。

2.4.2 6 <= 逻辑块号 < 6+128*2：一级索引，先查找 i_addr[6:7]获取一级索引块 block 号，再根据一级索引块 block 查找(logic_num-6)%128 位置，此处为对应的物理块号。

2.4.3 6+128*2 <= 逻辑块号 < 6+128*2+128+128*2：二级索引，先查找 i_addr[8:9]获取二级索引块 block 号，再根据二级索引块 block 查找(logic_num-6+128*2)/128 位置，获取一级索引块 block 号。再根据一级索引块 block 查找(logic_num-6+128*2)%128 位置，此处为对应的物理块号。

分配和释放的算法与查找类似，使用同样的算法映射，此处不再详细说明。



3. 目录管理模块

3.1 查看当前目录信息 int Func_Ls(vector<string> str)

首先检查参数合法性，之后遍历当前目录 now_dir 结构中的 child 子目录，打印 child 子目录项的名称、权限、所有者等各项信息。

3.2 创建目录 int Func_Mkdir(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的目录及其父目录。若目录已存在则创建失败，否则分配磁盘 block 空间，创建一个新的目录结构，并将新的目录结构加入父目录的 child 子目录项中。

3.3 删除目录 int Func_Rm(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的目录及其父目录。若目录不存在则删除失败，否则从父目录的 child 子目录项中查找该目录并删除，释放被删除的 block 空间。需要注意，由于删除的是目录，因此需要递归删除。先递归删除其中的子目录项和文件项并释放对应 block 和 inode 空间，最后再删除本目录并释放空间。

3.4 修改目录权限 int Func_Chmod(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的目录，若目录不存在则修改失败。修改对应目录的 inode.mode 目录权限。

3.5 进入目录 int Func_Cd(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的目录。若目录不存在则进入失败，否则修改当前目录 now_dir 为目标目录。

4. 文件管理模块

4.1 打开文件 int Func_OpenFile(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的文件。若文件不存在则打开失败，否则检查当前系统中打开文件数目 inode_open_num 是否达到上限。若未达到打开上限则将打开文件的信息记录在 my_file_map 结构中,初始化其中[inode_open_num]项，之后 inode_open_num++。

4.2 关闭文件 int Func_CloseFile(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的文件。若文件不存在则关闭失败，否则检查当前系统中打开文件结构 my_file_map 是否存在该文件，若存在，则从打开文件结构中删除该文件，之后 inode_open_num-=1。

4.3 读文件 int Func_ReadFile(vector<string> str)

首先检查参数合法性，之后查找内存中打开文件结构 my_file_map 是否存在该文件，若存在，则读取当前文件指针位置并记录。计算实际读取的字节数为 read_byte = min(read_byte, (inode.i_size - my_file_map[open_pos].offset)。之后使用 Func_AddrMapping 函数获取逻辑块映射的物理块，以 block 块为单位读取数据，同时将文件指针移动到读取后的位置。

4.4 写文件 int Func_WriteFile(vector<string> str)

首先检查参数合法性，之后查找内存中打开文件结构 my_file_map 是否存在该文件，若存在，则读取当前文件指针位置并记录。计算写入后新的文件大小 new_size = max((unsigned int)(write_inode->i_size), write_file->offset + str[2].size())。根据新的文件大小使用 Func_AllocateBlock()函数分配 block 块记录在 inode 的 i_addr 索引中，以 block 块为单位写入数据，同时将文件指针移动到写入后的位置。

4.5 移动文件指针 int Func_LseekFile(vector<string> str)

首先检查参数合法性，之后查找内存中打开文件结构 my_file_map 是否存在该文件，若存在，则修改当前文件指针位置。

4.6 修改文件权限 int Func_Chmod(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的文件。若文件不存在则修改权限失败，否则修改磁盘中对应文件的 Inode 结构的 inode.mode 文件权限。同时若内存中打开文件结构 my_file_map 存在该文件则使用 Func_RenewMemory()函数将磁盘中数据与内存同步。

4.7 展示系统中打开文件 int Func_OpenFileState(vector<string> str)

遍历内存中打开文件结构 my_file_map，打印所有打开文件的名称、所有者、文件指针位置等。

4.8 从系统外部加载文件 int Func_WriteFileFromMine(vector<string> str)

输入系统外部的路径，读取其中文件内容。将读取到的内容使用 Func_WriteFile 函数写入文件。

4.9 从文件系统向外部导出文件 int Func_WriteFileToMine (vector<string> str)

输入系统外部的路径，将文件系统中的对应文件导出到外部。

4.10 创建文件 int Func_CreateFile(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的文件。若文件存在则创建失败，否则分配一个 inode 结构，将该文件加入其父目录的 child 子目录项中。

4.11 删除文件 int Func_Rm(vector<string> str)

首先检查参数合法性，之后使用 Func_AnalysePath 函数分析路径，查找路径对应的文件。若文件不存在则删除失败，否则释放该文件的 inode 结构和 block 数据块。

5. 多用户管理模块

5.1 创建用户 int Func_AddUser(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户名是否已经存在或用户数是否已达上限。若存在或达到上限则创建失败，否则新建一个 UserInfo 结构，将对应的用户名、密码、uid 填入，user_list.user_num+=1，将新的用户列表存入磁盘。

5.2 删除用户 int Func_DelUser(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户是否已经存在。若不存在则删除失败，否则新建一个 UserInfo 结构，将对应的用户名、密码、uid 填入，将新的用户列表存入磁盘。

5.3 用户的登录 int Func_UserToken(string name, string passwd)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户名和密码是否被存在。若不存在则登录失败，否则登录成功。

5.4 用户的退出 int Func_UserExit(vector<string> str)

首先检查参数合法性，之后保存内存中修改，退出文件系统。

5.5 展示用户信息 int Func_ShowAllUser(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，遍历用户列表中的所有用户并打印用户名、uid、用户组等内容。

5.6 向用户组中加入用户 int Func_AddUserToGroup(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户是否已经存在。若不存在则加入失败，否则将用户的用户组修改为输入的用户组。

5.7 从用户组中删除用户 int Func_DelUserFromGroup(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户是否已经存在。若不存在则删除失败，否则将用户的对应用户组删除。

5.8 修改用户密码 int Func_UpdateUserPassword(vector<string> str)

首先检查参数合法性，从磁盘中读入当前的用户列表，之后检测输入的用户是否已经存在。若不存在则修改失败，否则用户输入旧密码验证身份，若密码正确则修改为新密码。

装 订 线

五．运行结果分析

1. 基础登录部分

1.1 初始化文件系统

1.1.1 重新构建磁盘

新建 root 用户和创建 bin、home 等起始目录：

```
是否重新构建磁盘？（y：重新构建，n：不重新构建）：y
创建新用户成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
登录
用户名:
```

1.1.2 不重新构建磁盘

```
是否重新构建磁盘？（y：重新构建，n：不重新构建）：n
登录
```

1.2 登录

1.2.1 登录成功

```
E:\操作系统课设\filesystem\filesystem\Debug\filesystem.exe
是否重新构建磁盘？（y：重新构建，n：不重新构建）：n
登录
用户名：root
密码：root123
欢迎登录，可以通过输入help查看本文件系统支持的指令
[root ~]#
```

1.2.2 登录失败

```
登录
用户名：root
密码：111
账号或密码有误，请重新输入
登录
用户名:
```

1.3 打印操作提示信息

```
[root ~]#help
支持指令          指令格式                                指令功能
help                                                      提示信息
cd                <文件路径>                              切换当前目录至指定路径
ls                                                        展示当前目录下内容
mkdir             <文件路径>                              在指定路径下创建目录
rm                <文件路径>                              删除指定路径下的目录或文件
chmod             <权限>      <文件路径>                  修改目录或文件的权限
create            <文件路径>                              在指定路径下创建文件
open              <文件路径>                              打开指定文件
close             <文件路径>                              关闭指定文件
cat               <文件路径>                              打印指定文件所有内容
read              <文件路径>  <读长度>                    读文件
write             <文件路径>  <写内容>                    写文件
lseek             <文件路径>  <新的文件指针位置>          移动文件指针
writein           <文件路径>  <外部文件路径>              从外部导入文件进入文件系统
writeout          <文件路径>  <外部文件路径>              从文件系统向外部导出文件
opened                                                    查看打开文件信息
adduser           <用户名>    <用户密码>                  增加用户
deluser           <用户名>                                删除用户
addusertogroup    <用户名>    <用户组名>                  增加用户进用户组
deluserfromgroup  <用户名>    <用户组名>                  从用户组中删除用户
whoami                                                    打印当前用户
showalluser                                               打印所有用户信息
updatepassword    <用户名>                                修改用户密码
su                <用户名>                                切换当前登录用户
exit                                                      退出文件系统
```

1.4 退出系统

```
[root ~]#exit
已退出文件系统

E:\操作系统课设\filesystem\filesystem\Debug\filesystem.exe (进程 41060)已退出，代码为 0。
按任意键关闭此窗口. . .
```

2. 目录管理部分

2.1 查看当前目录信息

```
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
[root ~]#
```

2.2 创建新目录

```
[root ~]#mkdir newdir
创建新文件夹成功
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      newdir          -       Thu Jun  2 00:11:46 2022
[root ~]#
```

2.3 切换目录

2.3.1 进入下级目录

```
[root ~]#cd newdir
[root newdir]#
```

2.3.2 返回上级目录

在本文件系统中，“.”代表当前目录，“..”代表上级目录，“~”代表根目录：

```
[root newdir]#cd ..
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      newdir          -       Thu Jun  2 00:13:15 2022
```

2.3.3 绝对路径切换目录

```
[root ~]#cd newdir
[root newdir]#mkdir newdir2
创建新文件夹成功
[root newdir]#cd ..
[root ~]#cd /newdir/newdir2
[root newdir2]#
```

2.4 修改目录权限

```
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      newdir          -       Thu Jun  2 00:16:09 2022
[root ~]#chmod 644 newdir
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
drw-r--r--      newdir          -       Thu Jun  2 00:16:09 2022
```

2.5 删除目录

在本文件系统中，删除目录会递归删除其子目录和文件

```
[root ~]#rm newdir
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
```

3. 文件管理部分

3.1 创建文件

3.1.1 相对路径创建文件

```
[root ~]#create file1
创建新文件成功
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
-rwxrwxrwx      file1           0       Thu Jun  2 00:18:15 2022
[root ~]#
```

3.1.2 绝对路径创建文件

```
[root ~]#create /home/file2
创建新文件成功
[root ~]#cd home
[root home]#ls
privilege       name            size    updatetime
drwxrwxrwx      texts           -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      reports         -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      photos          -       Thu Jun  2 00:08:30 2022
-rwxrwxrwx      file2           0       Thu Jun  2 00:19:13 2022
[root home]#
```

在文件管理之后运行过程中以 file1 文件为例。

3.2 打开文件

```
[root ~]#open file1
[root ~]#opened
file_name       user_name       offset
file1           root            0
[root ~]#
```

3.3 查看系统中所有打开文件

```
[root ~]#open /home/file2
[root ~]#opened
file_name       user_name       offset
file1           root            0
/home/file2     root            0
```

3.4 修改文件指针

```
[root ~]#opened
file_name       user_name       offset
file1           root            0
[root ~]#lseek file1 5
[root ~]#opened
file_name       user_name       offset
file1           root            5
```

3.5 写文件

```
[root ~]#write file1 test测试
[root ~]#cat file1
test测试
[root ~]#
```

3.6 读文件

3.6.1 不依赖文件指针，读取全部内容

```
[root ~]#cat file1
test测试
```

3.6.2 依赖文件指针，读取指定字节

```
[root ~]#lseek file1 0
[root ~]#read file1 4
test
[root ~]#
```

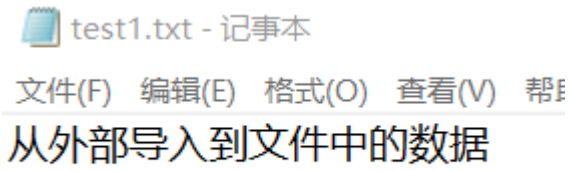
3.7 修改文件权限

```
[root ~]#chmod 000 file1
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:19:24 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
----------      file1           8       Thu Jun  2 00:50:32 2022
[root ~]#open file1
[root ~]#read file1 3
没有对应权限，打开文件失败
[root ~]#write file1 asd
没有对应权限，打开文件失败
[root ~]#
```

3.7 从文件系统外部导入文件

要导入的文件如下：

需要注意的是，由于编码格式问题，需要将要导入的文本文件编码格式设为 ANSI。



将外部数据导入文件系统中文件：

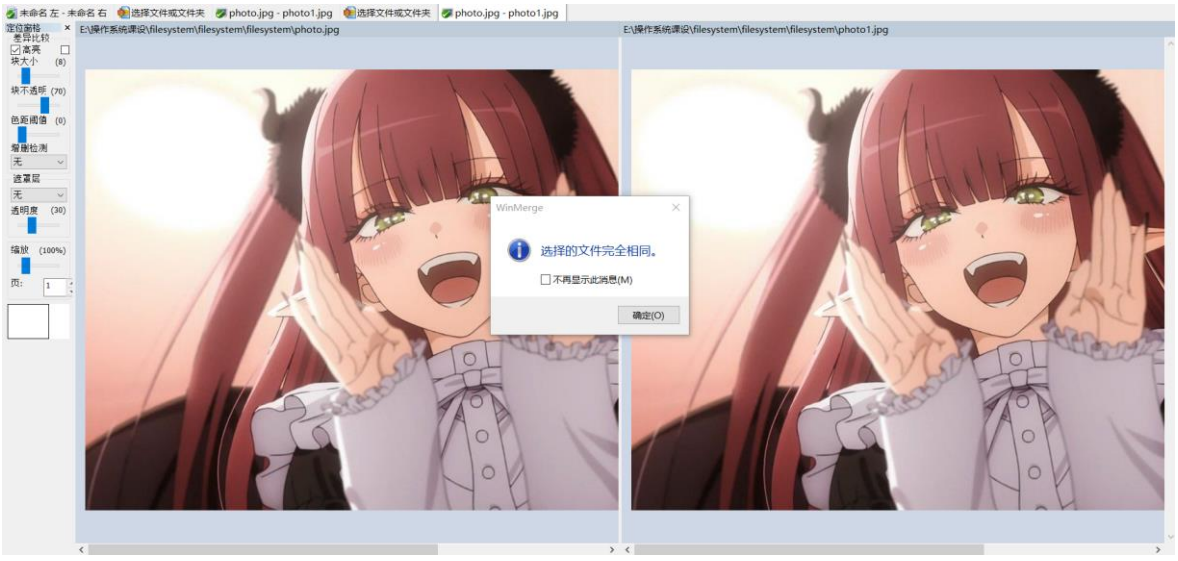
```
[root ~]#writein file1 test1.txt
[root ~]#cat file1
从外部导入到文件中的数据
[root ~]#
```

3.8 从文件系统向外部导出文件

此处以图片作为检测文件：

```
[root ~]#open file2
[root ~]#writein file2 photo.jpg
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 02:04:11 2022
drwxrwxrwx      etc             -       Thu Jun  2 02:04:11 2022
drwxrwxrwx      home            -       Thu Jun  2 02:04:11 2022
drwxrwxrwx      dev             -       Thu Jun  2 02:04:11 2022
-rwxrwxrwx      file1           27      Thu Jun  2 02:05:00 2022
-rwxrwxrwx      file2           86053   Thu Jun  2 02:09:42 2022
[root ~]#writeout file2 photo1.jpg
[root ~]#
```

将先前输入的图像文件与导出的图像文件进行比对，结果完全相同：



3.9 删除文件

3.9.1 未关闭情况下删除文件

```
[root ~]#opened
file_name        user_name        offset
file1            root             24
[root ~]#rm file1
该文件已打开，请先关闭
```

3.9.2 已关闭情况下删除文件

```
[root ~]#close file1
[root ~]#rm file1
[root ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:19:24 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
[root ~]#
```

4. 用户管理部分

4.1 创建用户

```
[root ~]#adduser user1 123456
创建新用户成功
[root ~]#showalluser
user_id          user_name        user_group
0                root             null
1                user1            null
[root ~]#
```

4.2 删除用户

```
[root ~]#deluser user1
删除用户成功
[root ~]#showalluser
user_id          user_name        user_group
0                root             null
[root ~]#
```

4.3 切换登录用户

```
[root ~]#su user1 123456
[user1 ~]#whoami
user1
[user1 ~]#
```

4.4 显示当前登录用户

```
[root ~]#whoami
root
[root ~]#
```

4.5 展示所有用户信息

```
[root ~]#showalluser
user_id          user_name        user_group
0                root             null
2                user1            null
3                user2            null
[root ~]#
```

4.6 将用户加入用户组

4.6.1 加入用户组

```
[root ~]#addusertogroup root group1
添加成功
[root ~]#showalluser
user_id          user_name        user_group
0                root             group1
2                user1            null
3                user2            null
[root ~]#
```

4.6.2 用户组相关权限验证

修改文件权限为同组可以读写：

```
[user1 ~]#chmod 770 file1
[user1 ~]#ls
privilege       name            size    updatetime
drwxrwxrwx      bin             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      etc             -       Thu Jun  2 00:08:30 2022
drwxrwxrwx      home            -       Thu Jun  2 00:19:24 2022
drwxrwxrwx      dev             -       Thu Jun  2 00:08:30 2022
-rwxrwx---      file1           0       Thu Jun  2 01:28:35 2022
[user1 ~]#
```

此时非同一用户组不能写入，设为同一用户组后可以写入：

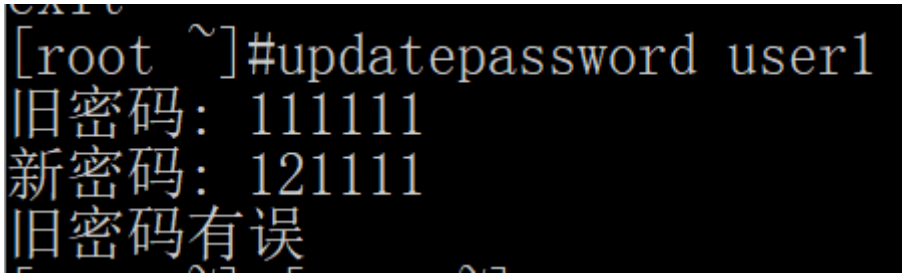
```
[user1 ~]#write file1 eeee
没有对应权限，打开文件失败
[user1 ~]#addusertogroup user1 group1
添加成功
[user1 ~]#showalluser
user_id         user_name       user_group
0               root            group1
2               user1           group1
3               user2           null
[user1 ~]#write file1 eeee
[user1 ~]#cat file1
```

4.7 从用户组中删除用户

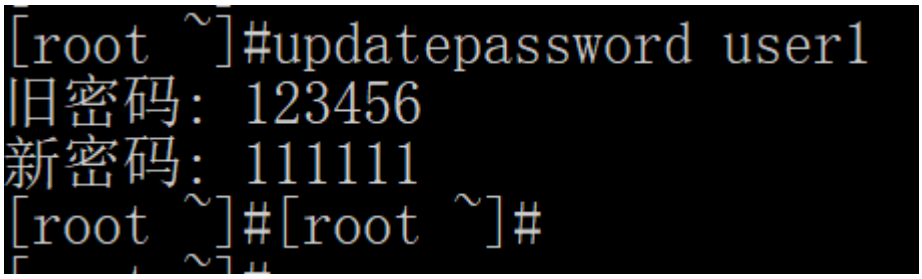
```
[root ~]#deluserfromgroup root group1
删除成功
[root ~]#showalluser
user_id         user_name       user_group
0               root            null
2               user1           null
3               user2           null
[root ~]#
```

4.8 修改用户密码

4.8.1 旧密码无效情况：



4.8.2 旧密码有效情况：



5.课程设计要求测试

5.1 创建文件夹，从外部导入文件进入系统

文件夹已经初始化创建完成，执行以下三组指令向系统中导入课设报告，关于课程设计报告的 ReadMe.txt 和一张图片：

导入课设报告：

```
create /home/texts/report.pdf
open /home/texts/report.pdf
writein /home/texts/report.pdf report.pdf
close /home/texts/report.pdf
```

导入 ReadMe.txt：

```
create /home/reports/ReadMe.txt
open /home/reports/ReadMe.txt
writein /home/reports/ReadMe.txt ReadMe.txt
close /home/reports/ReadMe.txt
```

导入一张图片：

```
create /home/photos/photo.jpg
open /home/photos/photo.jpg
writein /home/photos/photo.jpg photo.jpg
close /home/photos/photo.jpg
```

执行以下三组指令导出对应文件：

导出课设报告：

open /home/texts/report.pdf
writeout /home/texts/report.pdf report_out.pdf
close /home/texts/report.pdf

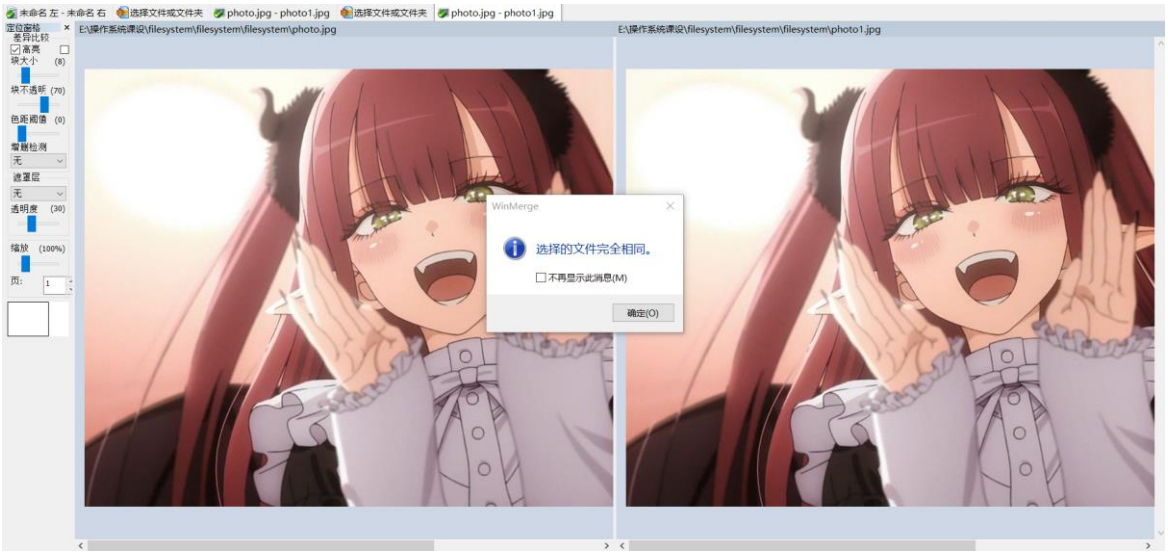
导出 ReadMe.txt：
open /home/reports/ReadMe.txt
writeout /home/reports/ReadMe.txt ReadMe_out.txt
close /home/reports/ReadMe.txt

导出一张图片：
open /home/photos/photo.jpg
writeout /home/photos/photo.jpg photo_out.jpg
close /home/photos/photo.jpg

验证，此处以图片为例，将图片导入/home/photos/photo.jpg 路径下，并以 photo_out.jpg 为名导出：

```
[root reports]#open /home/reports/ReadMe.txt
[root reports]#writeout /home/reports/ReadMe.txt ReadMe_out.txt
[root reports]#close /home/reports/ReadMe.txt
[root reports]#create /home/photos/photo.jpg
创建新文件成功
[root reports]#open /home/photos/photo.jpg
[root reports]#writein /home/photos/photo.jpg photo.jpg
[root reports]#close /home/photos/photo.jpg
[root reports]#open /home/photos/photo.jpg
[root reports]#writeout /home/photos/photo.jpg photo_out.jpg
[root reports]#close /home/photos/photo.jpg
[root reports]#cd /home/photos
[root photos]#ls
privilege       name            size    updatetime
-rwxrwxrwx      photo.jpg       86053   Thu Jun  2 02:44:53 2022
[root photos]#
```

通过对比软件，说明导出文件与导入文件一致，其他两组同理：



5.2 命令行方式测试文件读写

新建文件/test/Jerry，打开该文件，任意写入 800 个字节：

```
[root ~]#create /test/Jerry
当前目录下文件/目录数已达上限，创建目录失败
[root ~]#open /test/Jerry
路径不存在，打开文件失败
[root ~]#mkdir test
创建新文件夹成功
[root ~]#create /test/Jerry
创建新文件成功
[root ~]#open /test/Jerry
[root ~]#write /test/Jerry 1111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111112222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222
[root ~]#cd test
[root test]#ls
privilege        name             size    updatetime
-rwxrwxrwx       jerry            800     Thu Jun  2 03:01:38 2022
```

将文件读写指针定位到第 500 字节，读出 500 个字节到字符串 abc，并将 abc 写回文件：

```
[root test]#lseek /test/Jerry 500
[root test]#read /test/Jerry 500
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
[root test]#write /test/Jerry 2222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222
[root test]#ls
privilege        name             size    updatetime
-rwxrwxrwx       jerry            1100    Thu Jun  2 03:01:38 2022
```

由上图可见 readbyte=300，即读完剩下的所有内容，在将读到的 300 字节写入文件，此时的文件大小为 1100 字节。

六. 用户使用说明

1. 初次登录：

在初次登录时选择 y 重新构建磁盘文件系统，初始用户名为 root，密码为 root123。

```
是否重新构建磁盘？（y：重新构建，n：不重新构建）：y
创建新用户成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
创建新文件夹成功
登录
用户名：root
密码：root123
欢迎登录，可以通过输入help查看本文件系统支持的指令
[root ~]#
```

2. 指令说明

输入 help 可以查看文件系统支持的指令集及其使用方法。

```
[root ~]#help
支持指令          指令格式                                  指令功能
help                                                        提示信息
cd                <文件路径>                                切换当前目录至指定路径
ls                                                          展示当前目录下内容
mkdir             <文件路径>                                在指定路径下创建目录
rm                <文件路径>                                删除指定路径下的目录或文件
chmod             <权限>      <文件路径>                    修改目录或文件的权限
create            <文件路径>                                在指定路径下创建文件
open              <文件路径>                                打开指定文件
close             <文件路径>                                关闭指定文件
cat               <文件路径>                                打印指定文件所有内容
read              <文件路径> <读长度>                       读文件
write             <文件路径> <写内容>                       写文件
lseek             <文件路径> <新的文件指针位置>             移动文件指针
writein           <文件路径> <外部文件路径>                 从外部导入文件进入文件系统
writeout          <文件路径> <外部文件路径>                 从文件系统向外部导出文件
opened                                                      查看打开文件信息
adduser           <用户名>   <用户密码>                     增加用户
deluser           <用户名>                                  删除用户
addusertogroup    <用户名>   <用户组名>                     增加用户进用户组
deluserfromgroup  <用户名>   <用户组名>                     从用户组中删除用户
whoami                                                      打印当前用户
showalluser                                                 打印所有用户信息
updatepassword    <用户名>                                  修改用户密码
su                <用户名>                                  切换当前登录用户
exit                                                        退出文件系统
```

装 订 线

七. 实验总结

在本次操作系统课程设计中，我完成了一个基于 UNIX V6++操作系统的文件系统，完成了任务的基本要求，同时实现了允许多个用户同时访问二级文件系统的额外要求，并撰写了实验报告。在实验过程中，我增进了自己对于操作系统的了解，对逻辑地址到物理地址的映射、数据块的分配与释放算法等内容有了进一步的了解。

除此以外，在编写相关代码的过程中，我通过使用单例模式，对不同功能模块使用面向对象的思想进行类的封装，遵循代码规范和设计模式。同时使用了 STL 容器、函数指针绑定等一系列 C/C++语言的相关技术，提高了自己的编程能力。