**北 京 林 业 大 学**

**2019学年— 2020 学年第 2 学期 操作系统A课程设计 实习报告书**

专 业：\_\_\_信息学院网络工程专业

班 级： 网络工程18班

姓 名： 陈艳

学 号： 181002520

实习地点： 安徽省淮南市寿县保义镇

辅导教师： 李巨虎老师

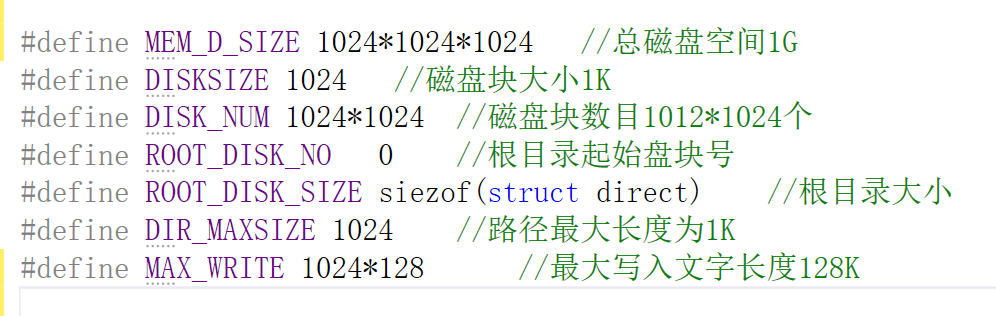
实习内容： 文件系统模拟

实习环境： Windows操作系统相应的开发平台和工具

实习步骤、方法、技术方案、结果、分析：

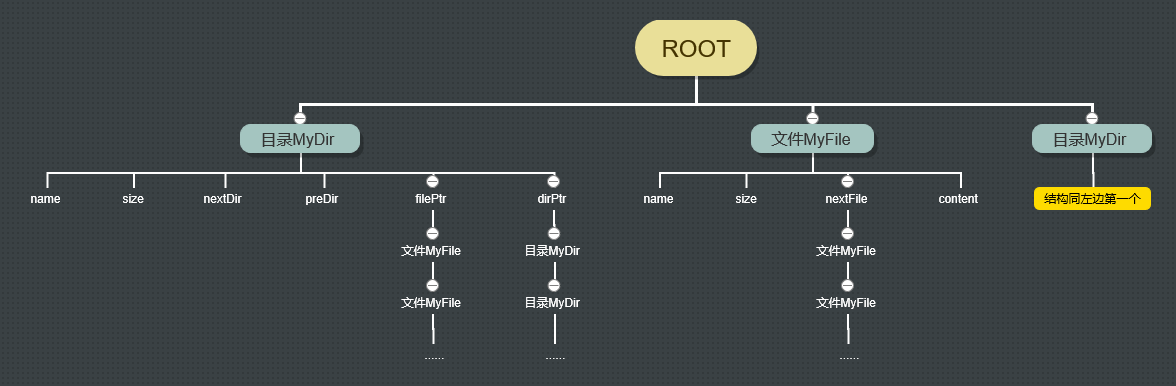
**第一部分：**描述实习中设计的文件系统格式

1.第一部分：用一个文件来模拟一个磁盘驱动器，文件系统大小，以及整体架构的相关常量宏定义如下：



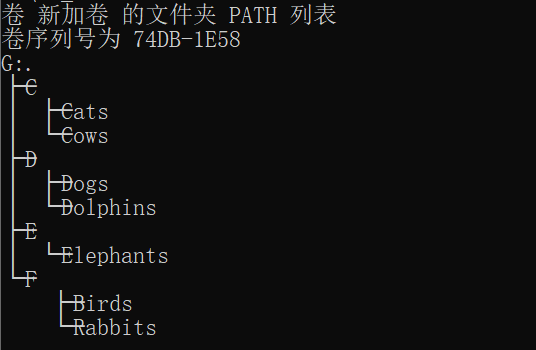
此磁盘驱动器是基于一个**多级文件目录的目录树架构**：

以根目录为主，向下呈现为分支状的目录结构的一种文件架构。



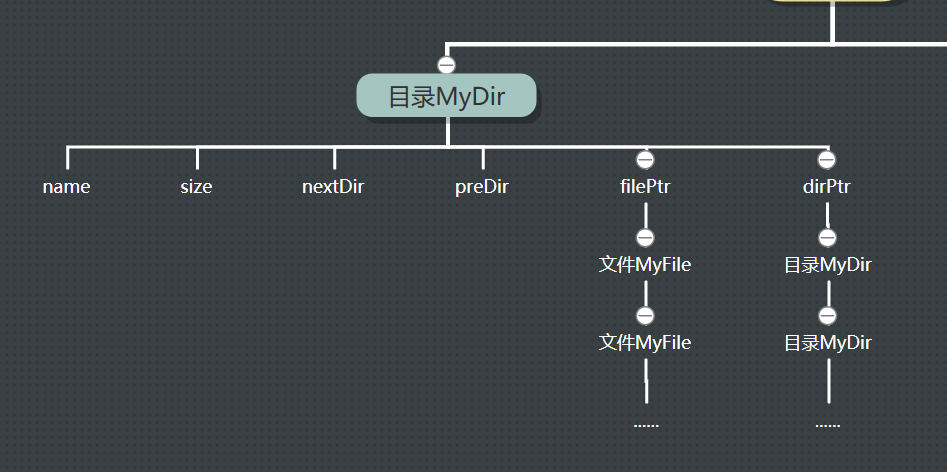
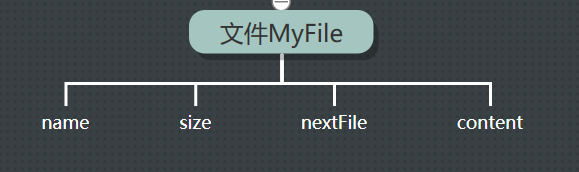
整个目录树架构最重要的就是根目录（root），所有的文件都与目录树有关。

2.在该磁盘驱动器中存放文件和目录，使用show(类似于windows下的tree命令)查看文件和目录结构如下：

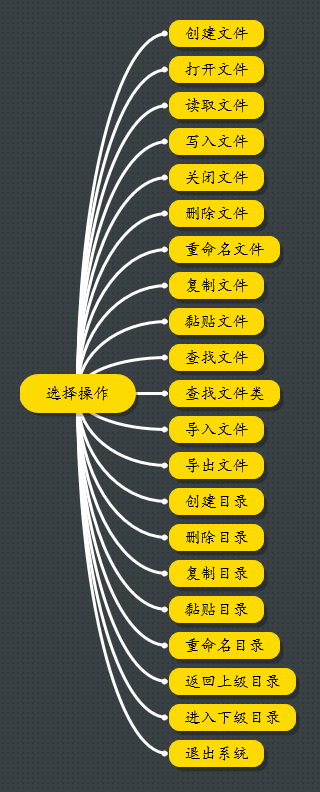


1. 在模拟的磁盘驱动器中，结合文件系统的基础理论知识，管理记录在磁盘驱动器中的内容（下图仅说明对文件和目录的操作，不涉及全局操作）：

主要借助于文件结构体和目录结构体管理：



以下是对于文件和目录的action操作：

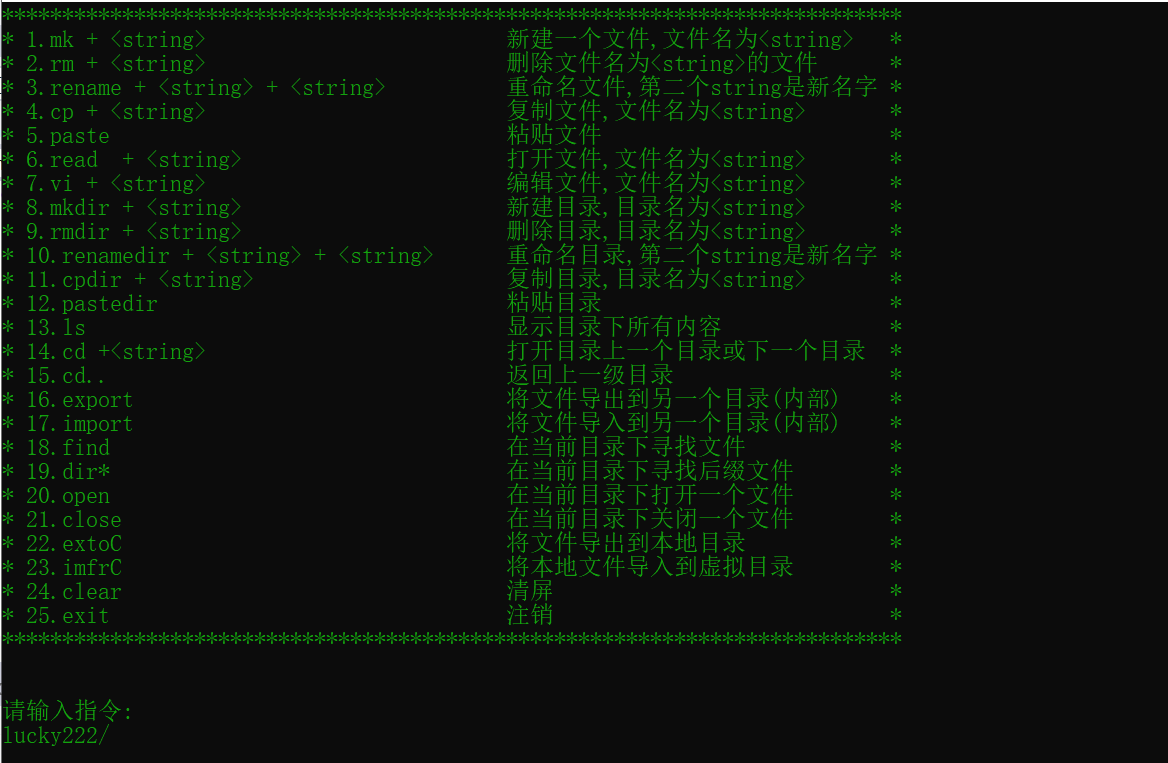


第二部分：文件系统的访问软件所包含的功能，即实现的命令及有关参数

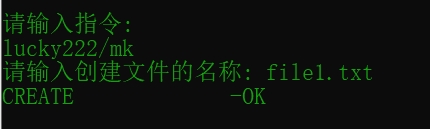
4.第二部分甲模块书写命令行形式的驱动器内容访问**shell交互接口**，即仿照cmd.exe的形式完成磁盘驱动器中内容的查看：

查看结果案例显示如下：

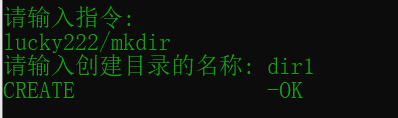
帮助指令：



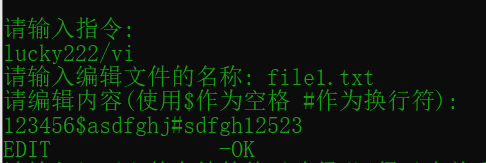
创建文件：



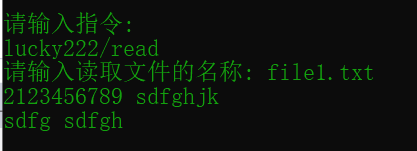
创建目录：



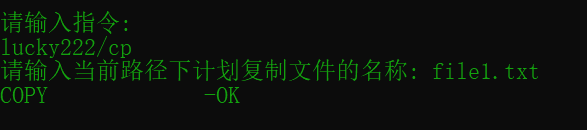
编辑文件：



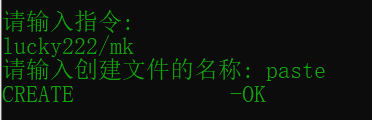
读取文件：



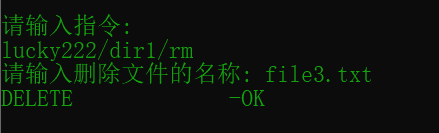
复制文件：



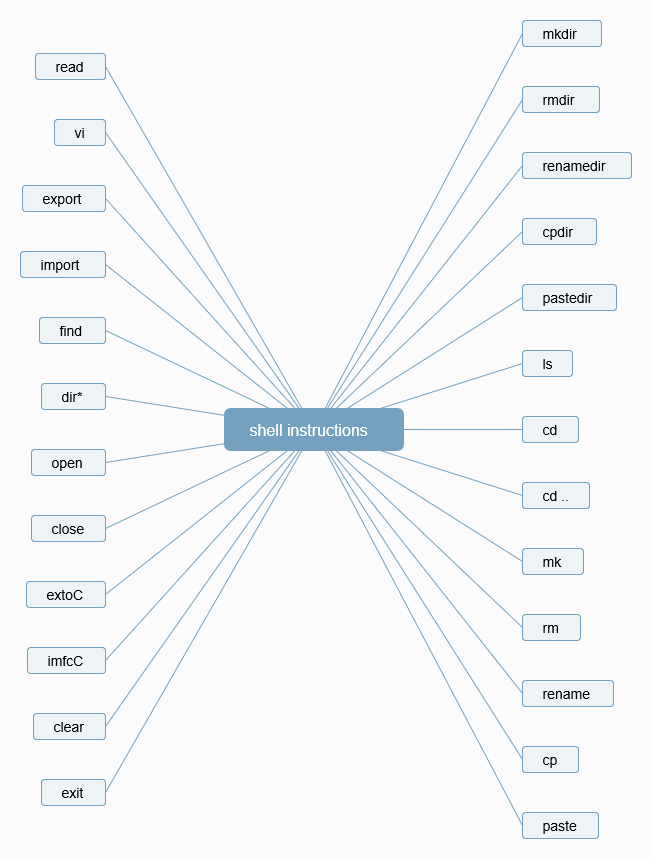
黏贴文件：



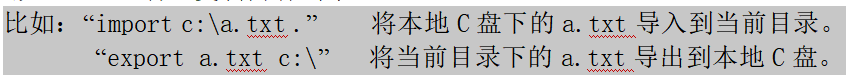
删除文件：



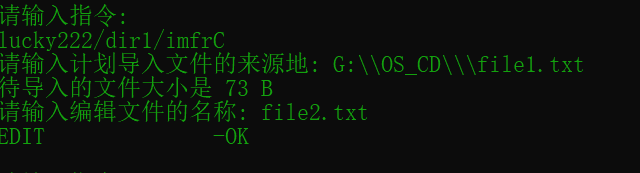
5.具体实现指令有：



6.第二部分乙模块支持从本地磁盘复制内容到虚拟的磁盘驱动器中，也支持从虚拟的磁盘驱动器复制内容到本地磁盘:



从本地导入文件到虚拟系统：

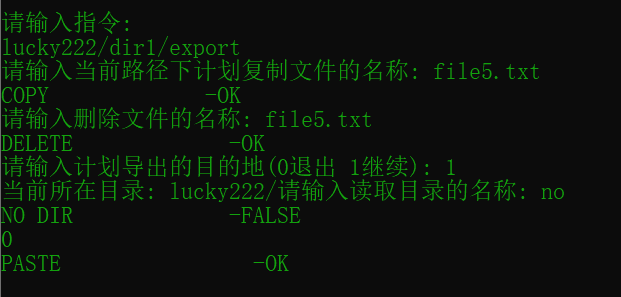


将虚拟系统文件导出到本地：

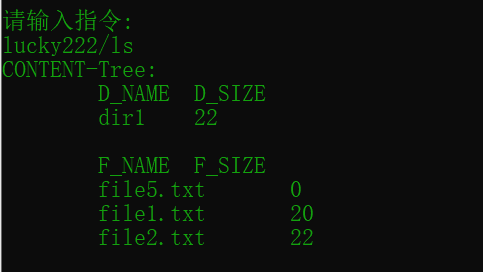


附加：

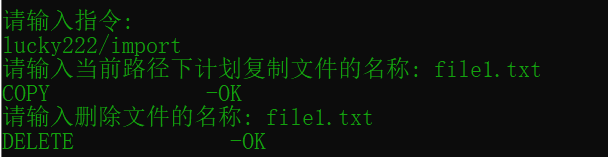
文件系统内部互相导出：



导出结果截图显示：

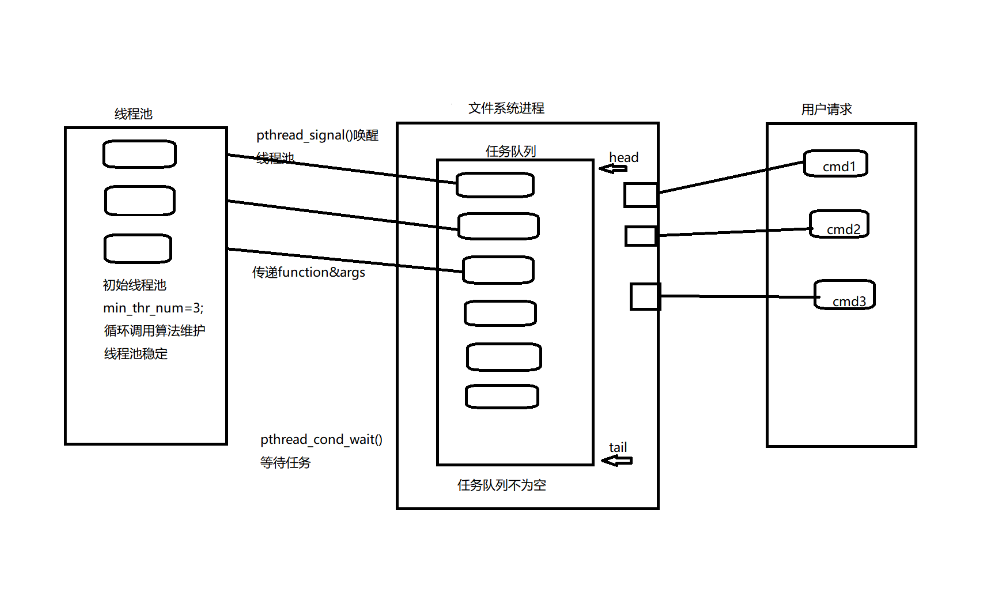


文件系统内部导入：



7.提高部分：用多线程的方式进行设计，具体来说，用一个线程来和用户进行交互，接受请求并将请求转换为对应的消息，通知后台维护虚拟磁盘驱动器的线程。

以下是个人对于多线程式并发的思路理解：

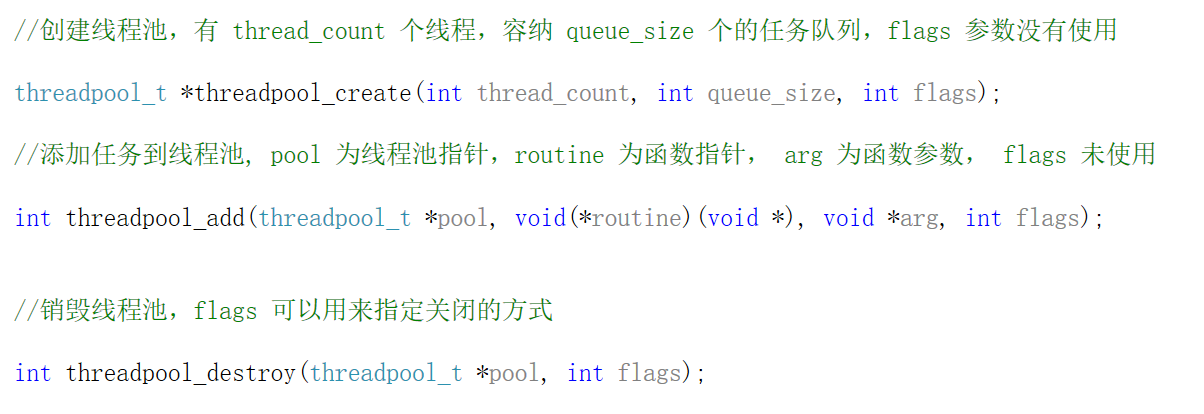


思路分析：每一次客户端打开一个cmd命令窗口，实际上都是在一个文件系统进程内部工作，每一次发出一个请求指令，例如读文件file1，此时文件系统作为服务器，这个客户端指令则会将函数和参数一并存储在任务队列中（队列不为满的情况下）；

在文件系统与线程池之间，文件系统作为客户端，此时线程池担任服务器的角色，一旦任务队列有任务出现，唤醒线程池，线程池则从队列中取出任务，完成后撤销线程，释放空间；

线程池结构和参数解释见第四部分主要结构体说明模块；

以下是线程池对外的三个接口：



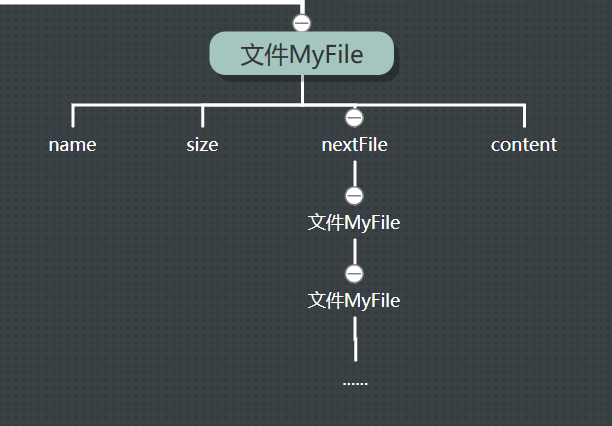
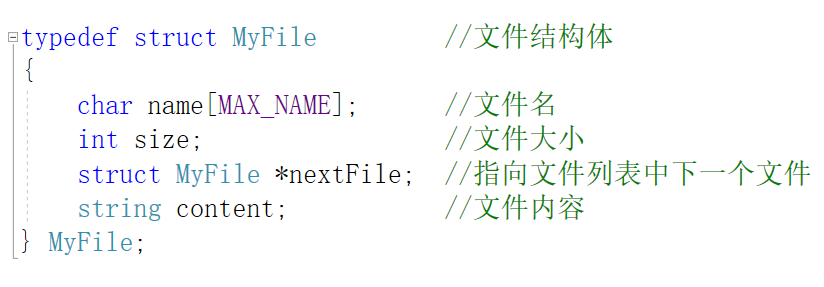
第三部分：文件系统的访问软件的整体流程图，以及各功能的流程图

8.详见使用虚拟系统使用说明文档

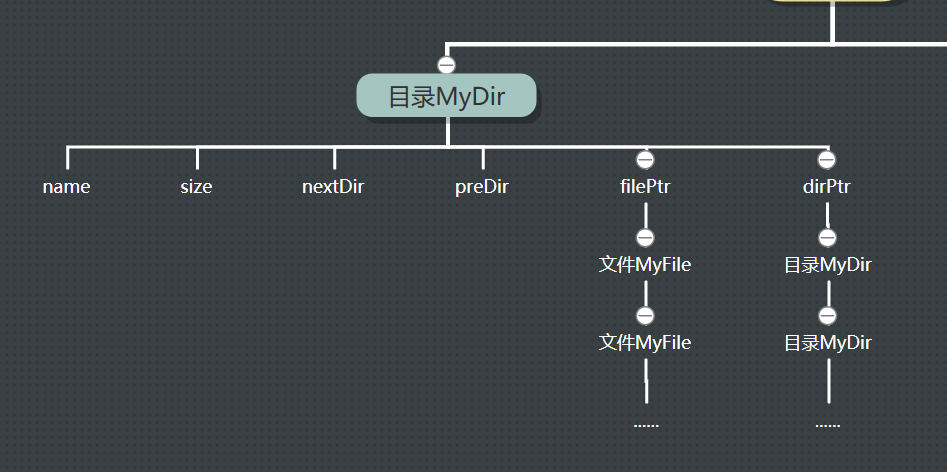
第四部分：主要数据结构的说明和主要函数的说明

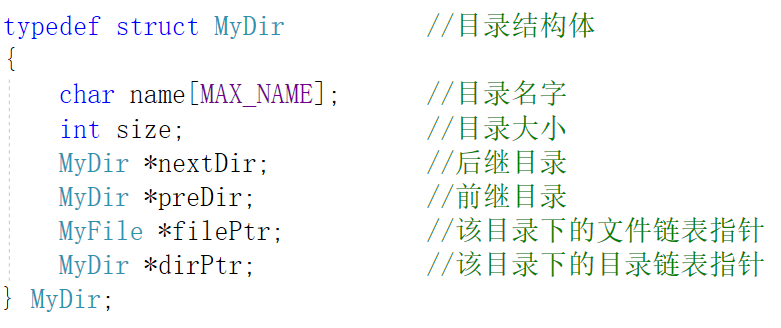
9.主要数据结构：

文件结构体：

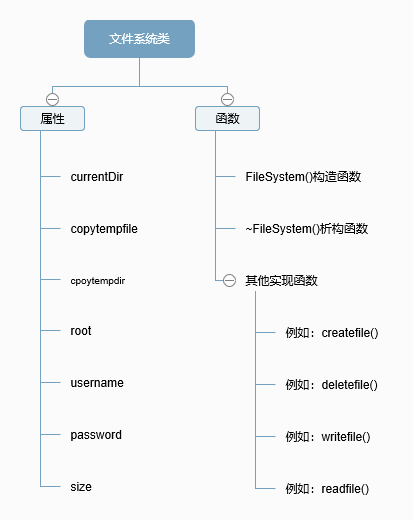
  


目录结构体：





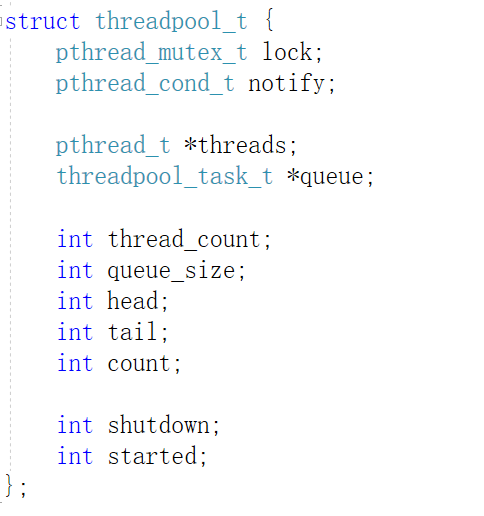
文件系统类：



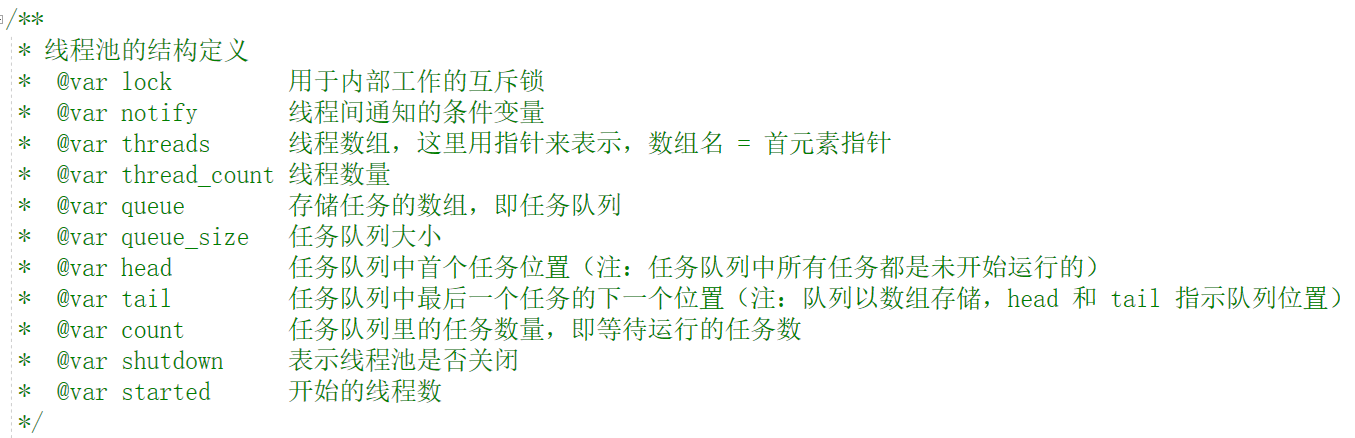




线程池结构体：

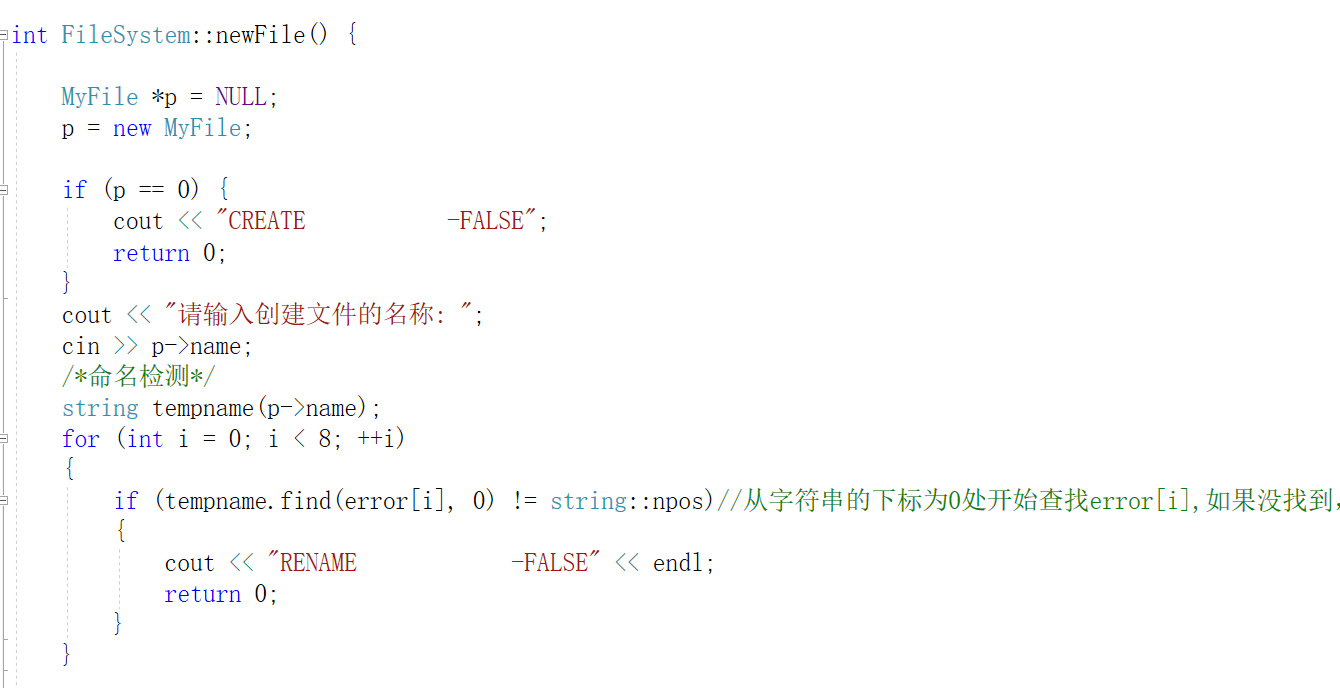


重要参数解释：



1. 主要函数说明：

Createfile()函数：创建文件，首先输入文件名，检测命名是否符合规范；



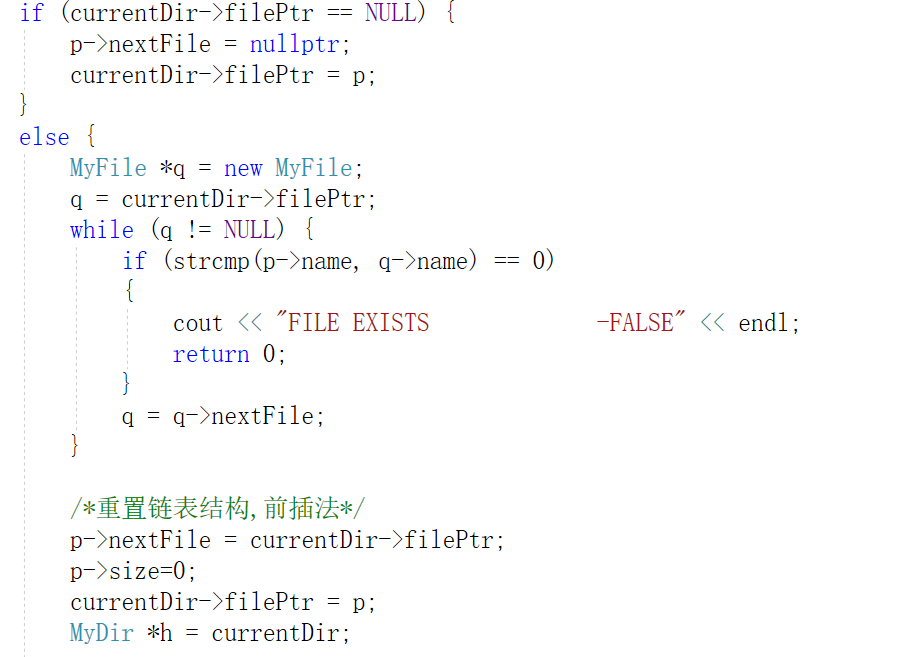
创建时候情况如下

\* 1. 目录下没有文件

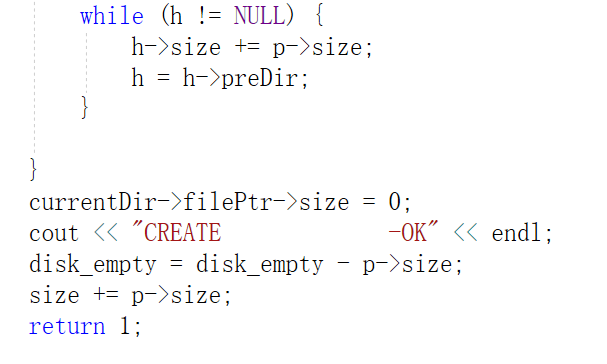
\* 2. 目录下有文件，新文件命名冲突

\* 3. 目录下有文件，新文件无命名冲突

检测有无同名函数



同时，更改上级目录：



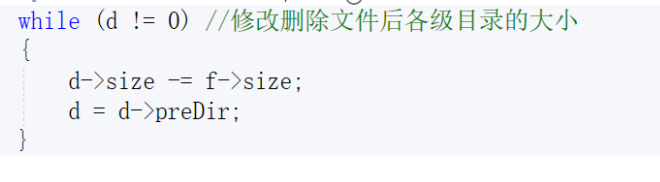
Deletefile()函数：

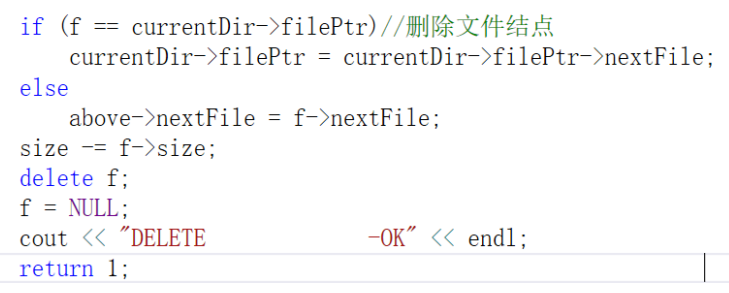
首先判断此目录下是否含有该文件，其次修改删除文件后各级目录的大小；

\* 删除时考虑

\* 1. 需要删除的文件恰好是目录文件链表中的头节点

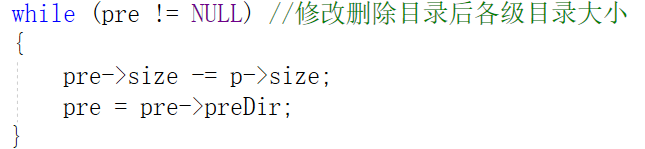
\* 2. 需要删除的文件在链表中间





DeleteDir()函数：

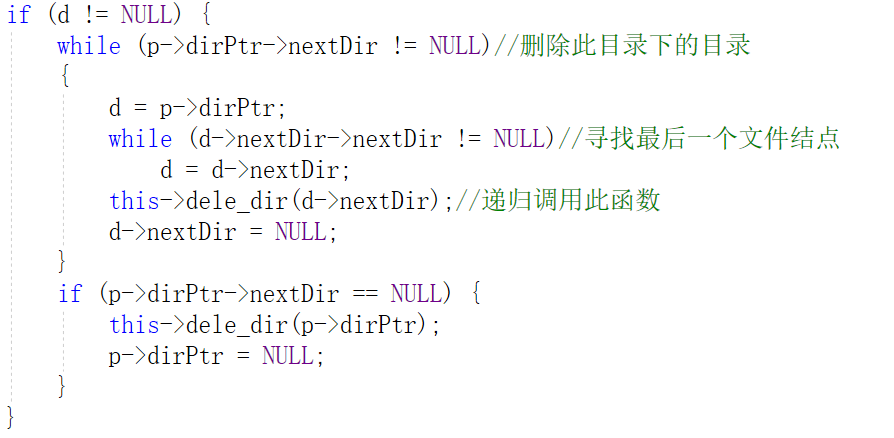
首先修改此目录上级目录的每一次结点，其次递归删除目录下的每一个子目录，再递归删除目录下的每一个链接文件



删除文件：



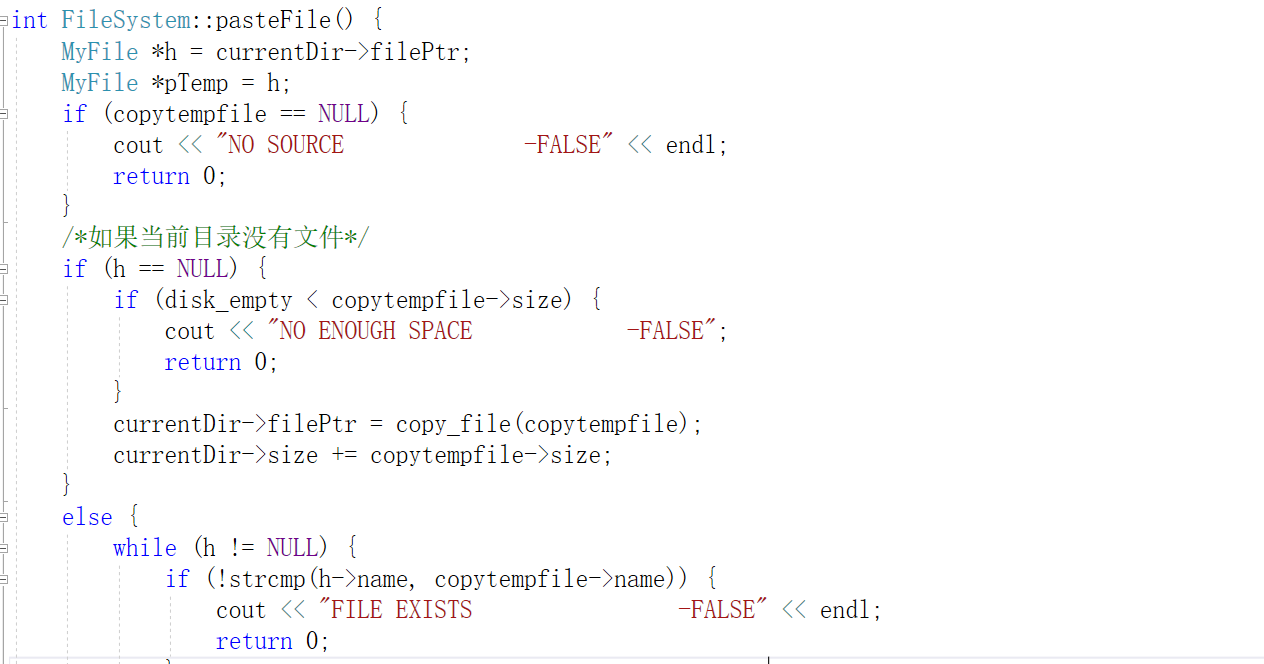
删除子目录：



Copy\_file() & pasteFile()函数:

首先将计划要复制的文件保存到copytempfile，用于临时存储；然后使用paste指令，将copytempfile的内容一一复制到目标文件中：



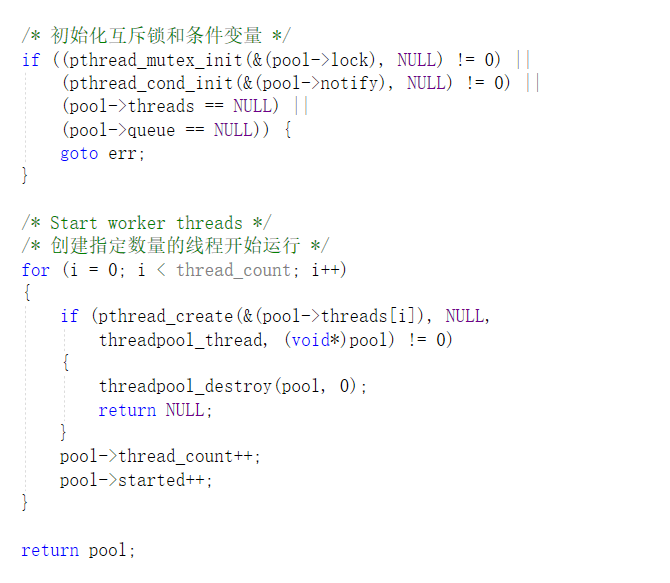


关于多线程的函数：

Threadpool\_create()函数：线程池创建

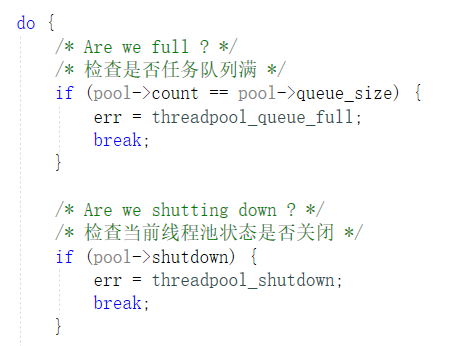
首先申请内存，若申请失败直接结束；反之，继续执行下去，继而初始化线程池大小THREAD，任务队列大小QUEUE，以及队列的head tail指向；然后初始化THERAD个最小线程数，在线程池中等待任务，一旦有任务到来，唤醒线程池开始工作；一旦任务超过线程池容量，则wait，一直自循环检测，直到线程池有空闲块出现；最后，初始化互斥锁和条件变量。

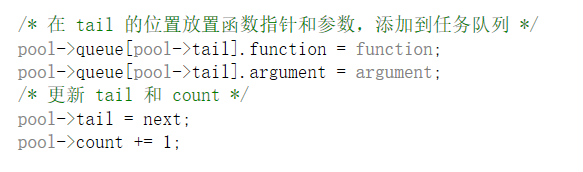




Threadpool\_add()函数：  
向线程池中添加任务：function & parameters

首先检测互斥锁，其次查询任务队列是否空，是否满；若空，则可以添加任务，并唤醒线程池；若满，则等待队列出现空闲区域；然后将function&args一并传入任务队列,进一步查询线程池状态，传入参数和执行函数加入任务；最后，释放互斥锁。



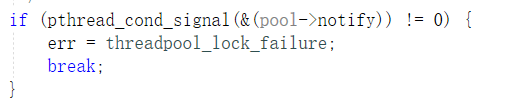


pthread\_cond\_broadcast功能和pthread\_cond\_signal函数一样，均是用信号唤醒线程池的操作：

发出 signal,表示有 task 被添加进来了

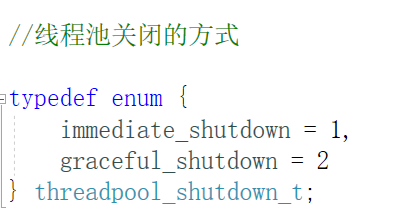
如果由因为任务队列空阻塞的线程，此时会有一个被唤醒

如果没有则什么都不做



Thread\_destroy()函数：

线程销毁函数：首先取得互斥资源，判断是否已在其他地方关闭；若没有，则读取指定要求的销毁方式shutdown=1或者2

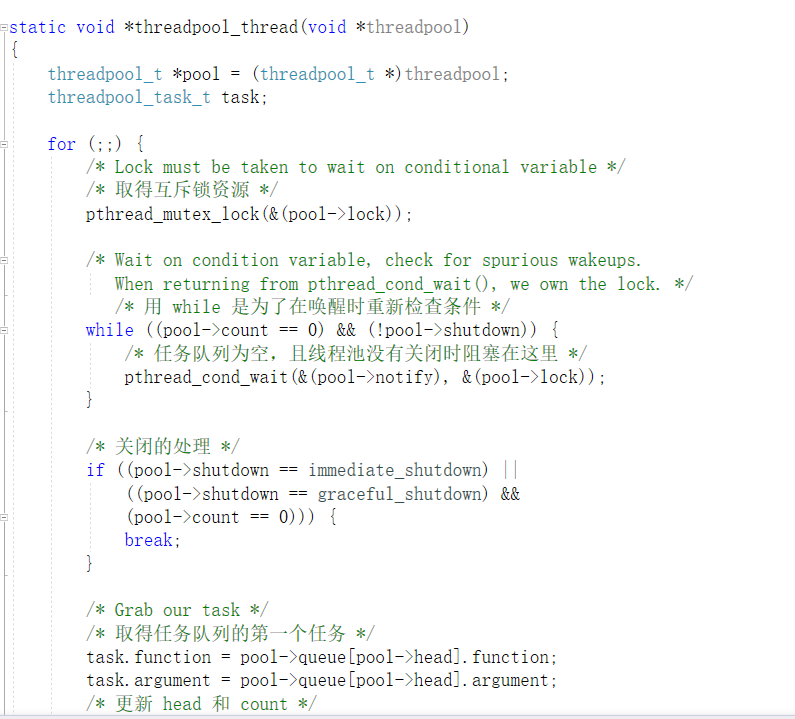


唤醒因条件变量而等待的线程，并释放互斥锁；最后等待线程结束。



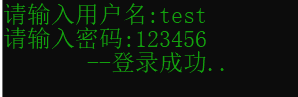
Threadpool\_thread()函数：

内部的静态函数，是每个线程池的线程都在跑的函数，用于彼此之间互通信号变量



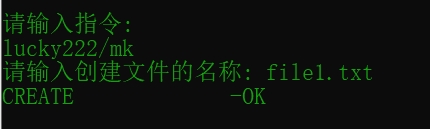
第五部分：测试部分

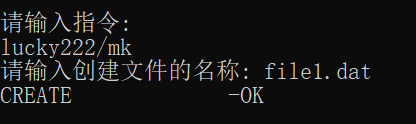
11.系统登录测试：



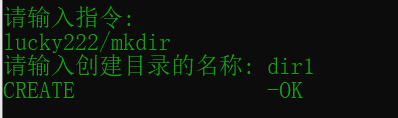
12.指令测试(主要以dat二进制文件和txt文本文件为测试样例说明)：

创建文件：

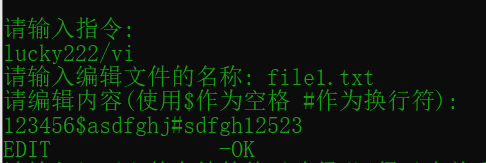


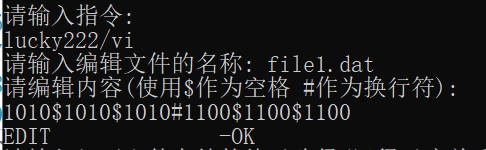


创建目录：

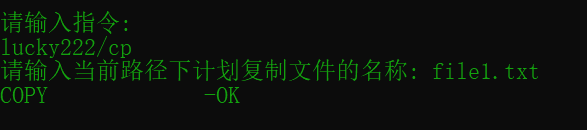


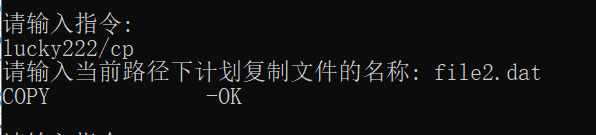
编辑文件：





复制文件：

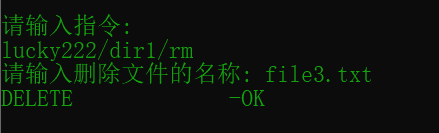


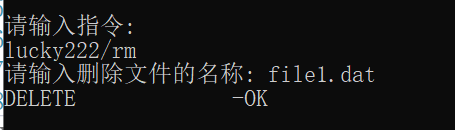


黏贴文件：

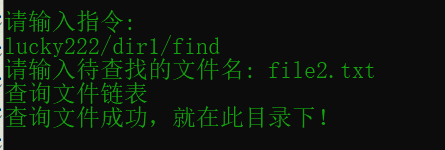


删除文件：

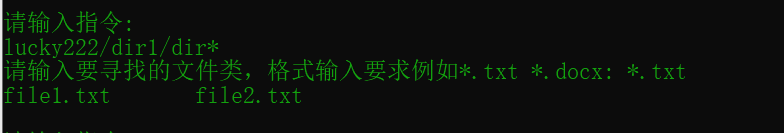




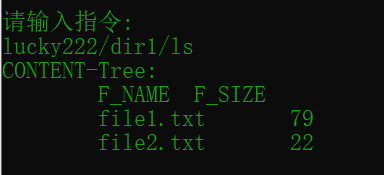
查找文件：



查找相同后缀文件：



查看目录树：

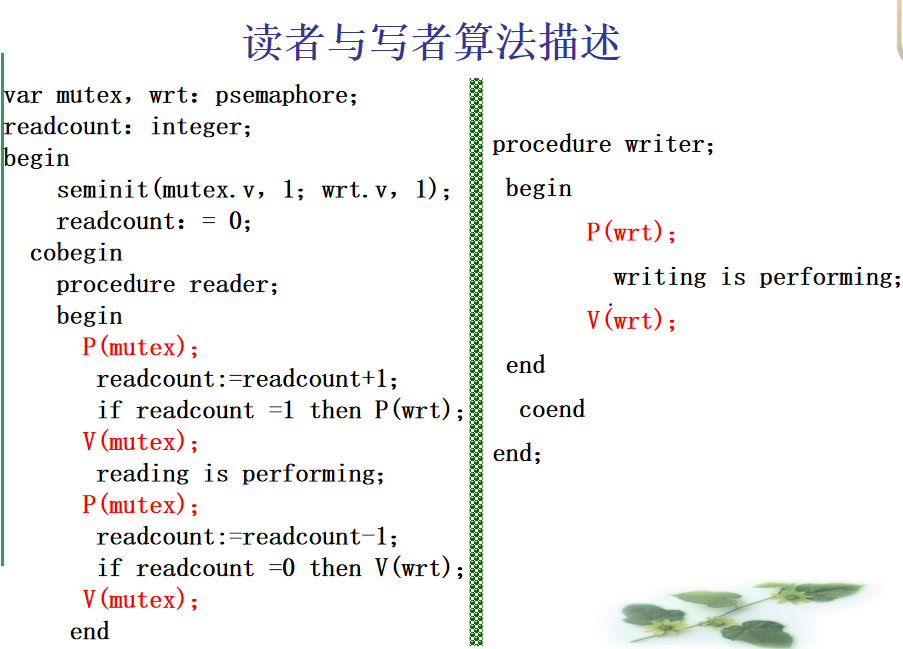


查看版本信息：

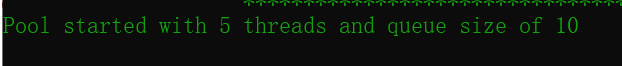


13.多线程测试：

对于文件的读写是一个互斥的操作，在第三章的reader-writer关系中，我们知道**读者与读者之间是共享的，读者与写者之间是互斥的，写者与写者之间是互斥的**，他们之间的PV操作如下：



实现截图说明：



实现结果截图如下：

两个读者同时读取一个文件：

读取文件：

