

# 操作系统实验五：文件系统

**实验报告**

学 院： 计算机科学与技术学院

班 级： 计科1601

姓 名： 赵宇

学 号： 16281026

**北京交通大学**

2019年6月13日

**一、实验简介**

本实验要求在模拟的I/O系统之上开发一个简单的文件系统。用户通过create, open, read等命令与文件系统交互。文件系统把磁盘视为顺序编号的逻辑块序列，逻辑块的编号为0至*L −* 1。I/O系统利用内存中的数组模拟磁盘。

**2 I/O**系统

实际物理磁盘的结构是多维的：有柱面、磁头、扇区等概念。I/O系统的任务是隐藏磁盘的结构细节，把磁盘以逻辑块的面目呈现给文件系统。逻辑块顺序编号，编号取值范围为0至*L−*1，其中*L*表示磁盘的存储块总数。实验中，我们可以利用数组ldisk[C][H][B]构建磁盘模型，其中CHB分别表示柱面号，磁头号和扇区号。每个扇区大小为512字节。I/O系统从文件系统接收命令，根据命令指定的逻辑块号把磁盘块的内容读入命令指定的内存区域，或者把命令指定的内存区域内容写入磁盘块。文件系统和I/O系统之间的接口由如下两个函数定义：

*•* read\_block(int i, char \*p);

该函数把逻辑块*i*的内容读入到指针*p*指向的内存位置，拷贝的字符个数为存储块的长度*B*。

*•* write block(int i, char \*p);

该函数把指针*p*指向的内容写入逻辑块*i*，拷贝的字符个数为存储块的长度*B*。此外，为了方便测试，我们还需要实现另外两个函数：一个用来把数组*ldisk* 存储到文件；另一个用来把文件内容恢复到数组。

**3** 文件系统

文件系统位于I/O系统之上。

**3.1** 用户与文件系统之间的接口

文件系统需提供如下函数；create, destroy, open, read, write。

*•* create(filename): 根据指定的文件名创建新文件。

*•* destroy(filename): 删除指定文件。

*•* open(filename): 打开文件。该函数返回的索引号可用于后续的read, write, lseek,或close操作。

*•* close(index): 关闭制定文件。

*•* read(index, mem\_area, count): 从指定文件顺序读入*count*个字节*memarea*指定的内存位置。读操作从文件的读写指针指示的位置开始。

*•* write(index, mem\_area, count): 把*memarea*指定的内存位置开始的*count*个字节顺序写入指定文件。写操作从文件的读写指针指示的位置开始。

*•* lseek(index, pos): 把文件的读写指针移动到*pos*指定的位置。*pos*是一个整数，表示从文件开始位置的偏移量。文件打开时，读写指针自动设置为0。每次读写操作之后，它指向最后被访问的字节的下一个位置。*lseek*能够在不进行读写操作的情况下改变读写指针能位置。

*•* directory: 列表显示所有文件及其长度。

**3.2** 文件系统的组织

磁盘的前*k*个块是保留区，其中包含如下信息：位图和文件描述符。位图用来描述磁盘块的分配情况。位图中的每一位对应一个逻辑块。创建或者删除文件，以及文件的长度发生变化时，文件系统都需要进行位图操作。前*k*个块的剩余部分包含一组文件描述符。每个文件描述符包含如下信息：

*•* 文件长度，单位字节

*•* 文件分配到的磁盘块号数组。该数组的长度是一个系统参数。在实验中我们可以把它设置为一个比较小的数，例如3。

**3.3** 目录

我们的文件系统中仅设置一个目录，该目录包含文件系统中的所有文件。除了不需要显示地创建和删除之外，目录在很多方面和普通文件相像。目录对应0号文件描述符。初始状态下，目录中没有文件，所有，目录对应的描述符中记录的长度应为0，而且也没有分配磁盘块。每创建一个文件，目录文件的长度便增加一分。目录文件的内容由一系列的目录项组成，其中每个目录项由如下内容组成：

*•* 文件名

*•* 文件描述符序号

**3.4** 文件的创建与删除

创建文件时需要进行如下操作；

*•* 找一个空闲文件描述符(扫描ldisk [0]～ldisk [k - 1])

*•* 在文件目录里为新创建的文件分配一个目录项（可能需要为目录文件分配新的磁盘块）

*•* 在分配到的目录项里记录文件名及描述符编号．

*•* 返回状态信息（如有无错误发生等）

删除文件时需要进行如下操作（假设文件没有被打开）：

*•* 在目录里搜索该文件的描述符编号

*•* 删除该文件对应的目录项并更新位图

*•* 释放文件描述符

*•* 返回状态信息

**3.5** 文件的打开与关闭

文件系统维护一张打开文件表．打开文件表的长度固定，其表目包含如下信息：

*•* 读写缓冲区

*•* 读写指针

*•* 文件描述符号

文件被打开时，便在打开文件表中为其分配一个表目；文件被关闭时，其对应的表目被释放。读写缓冲区的大小等于一个磁盘存储块。打开文件时需要进行的操作如下：

*•* 搜索目录找到文件对应的描述符编号

*•* 在打开文件表中分配一个表目

*•* 在分配到的表目中把读写指针置为０，并记录描述符编号

*•* 读入文件的第一块到读写缓冲区中

*•* 返回分配到的表目在打开文件表中的索引号

关闭文件时需要进行的操作如下：

*•* 把缓冲区的内容写入磁盘

*•* 释放该文件在打开文件表中对应的表目

*•* 返回状态信息

**3.6** 读写

文件打开之后才能进行读写操作．读操作需要完成的任务如下：

1. 计算读写指针对应的位置在读写缓冲区中的偏移

2. 把缓冲区中的内容拷贝到指定的内存位置，直到发生下列事件之一：

*•* 到达文件尾或者已经拷贝了指定的字节数。这时，更新读写指针并返回相应信息

*•* 到达缓冲区末尾。这时，把缓冲区内容写入磁盘，然后把文件下一块的内容读入磁盘。最后返回第2步。

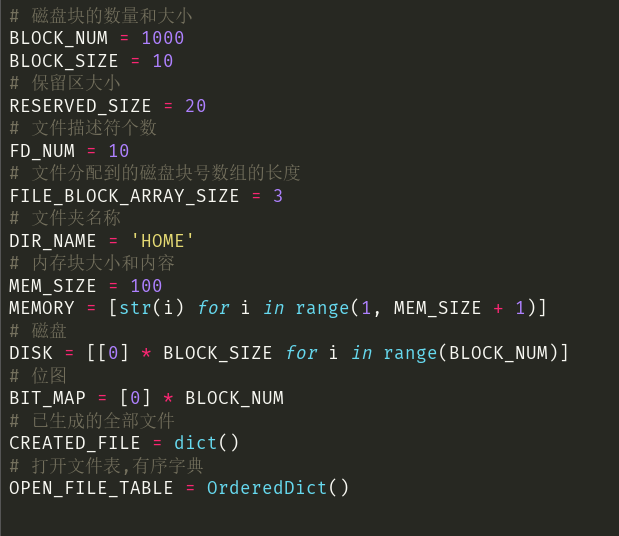
其他操作请同学们自己考虑。

**4** 测试

为了能够对我们的模拟系统进行测试，请编写一个操纵文件系统的外壳程序或者一个菜单驱动系统。

**二、文件系统基本组织**

实验要求基本数据结构和参数设置如下：



本文件系统的构建大致分为三个模块，用四个类及其方法实现，分别是文件类、目录类、文件描述符类、打开对象类。

各个类与函数的功能解释如下：

## 文件处理

|  |
| --- |
| *# 文件类* **class** File(object):  *# 文件描述符，磁盘块号列表*  **def** \_\_init\_\_(self, fd, file\_name, new=1):   *# 返回文件的第index块的内容* **def** get\_block(self, index):   *# 关闭文件时刷新某一文件的内容* **def** refresh\_block(self, index, new\_block):   *# 为文件添加新的磁盘块* **def** add\_new\_block(self):    *# 返回文件长度* **def** get\_file\_length(self):    *# 初始化文件，清空内容* **def** initial(self):  *# 目录类* **class** Directory(object):  *# 文件信息：文件描述符和文件描述符二元组列表* file\_fd\_name = list()   **def** \_\_init\_\_(self, fd, dir\_name):   *# 添加文件到文件夹中* **def** add\_file(self, fd, file\_name):   *# 根据文件名删除文件,返回文件描述符* **def** delete\_file(self, file\_name):    *# 依据文件名得到文件描述符* **def** get\_fd(self, file\_name):   *# 文件描述符类* **class** FileDescriptor(object):  **def** \_\_init\_\_(self, fd\_num):    *# 返回未使用过的的文件描述符* **def** get\_free\_fd(self):   *# 使用文件或者目录对象指针代替  # 文件描述符列表中对应位置的0* **def** fd\_allocate(self, fd, file):    *# 释放文件描述符* **def** fd\_release(self, fd):   *# 打开文件对象* **class** OpenFile(object):    **def** \_\_init\_\_(self, fd):    *# 将文件中的下一块的内容读到缓冲区中* **def** buffer\_next\_block(self):    *# 返回当前缓冲区存放的是文件内容的第几块* **def** get\_block\_ptr(self):   *# 返回缓冲区内容* **def** get\_buffer(self):    *# 返回文件描述符* **def** get\_fd(self):    *# 将缓冲区的内容写入文件相应块* **def** dump\_buffer(self):    *# 在缓冲区中载入最后一块内存* **def** buffer\_tail\_block(self):    *# 为读操作重新设置缓冲区内容* **def** read\_buffer(self):    *# 进入读状态* **def** enter\_read(self):  *# 进入写状态* **def** enter\_write(self): |

## I/O系统

|  |
| --- |
| *# 读取磁盘块的内容num个字节到内存* **def** read\_block(block\_data, mem\_begin, num):  **def** write\_block(mem\_begin, num):  *# 将磁盘内容保存到文件中* **def** dump\_disk():   *# 将文件内容恢复到磁盘* **def** load\_disk(): |

## 用户与文件系统接口

|  |
| --- |
| *# 根据文件名创建文件* **def** create(file\_name):   *# 根据文件名删除文件* **def** destroy(file\_name):   *# 打开文件* **def** open\_file(file\_name):   *# 关闭文件* **def** close\_file(file\_name):  *# 从指定文件读取num个字节到mem\_begin开始的内存区中* **def** read(file\_name, mem\_begin, num):   *# 将内存数据添加到文件中,不覆盖原来的内容* **def** append(file\_name, begin, num):   *# 将内存内容写入到文件中，覆盖原文件* **def** write(file\_name, begin, num):   *# 修改已打开文件的读写指针* **def** read\_write\_seek(file\_name, pos):   *# 查看文件内容* **def** view\_file(file\_name):   *# 打印当前文件情况* **def** file\_status(): |

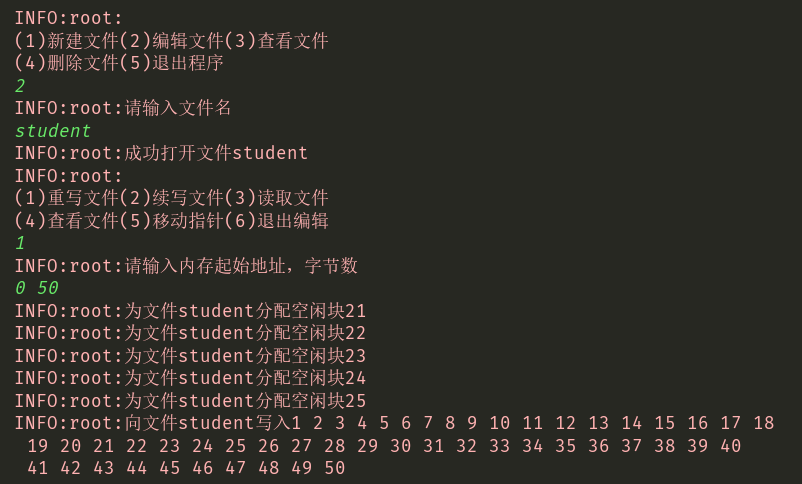
## 4. 测试程序

|  |
| --- |
| *# 菜单驱动程序* **def** menu\_drive():  *#主程序*   **if** \_\_name\_\_ == **'\_\_main\_\_'**:  *#函数调用关系*  load\_disk()  menu\_drive()  dump\_disk() |

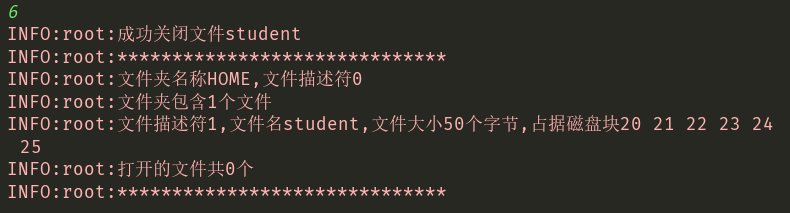
1. **实验结果及测试**
2. 新建文件



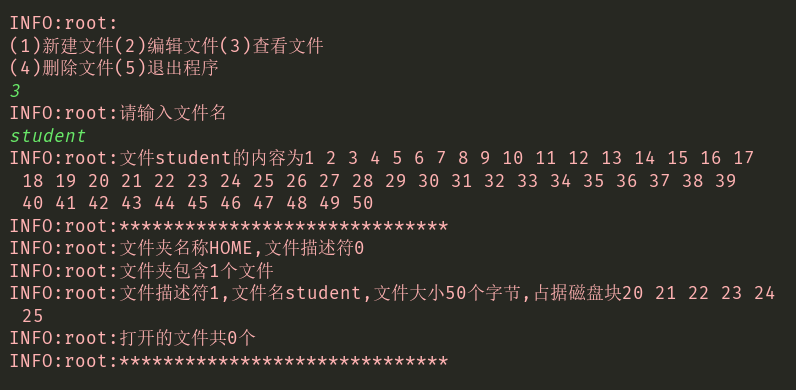
1. 编辑文件



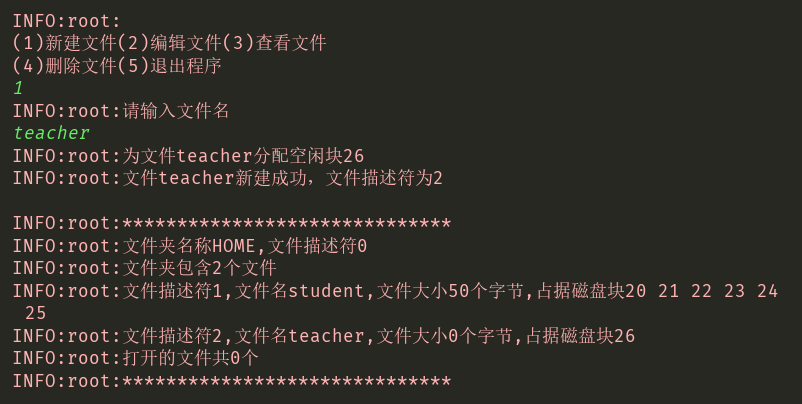
输入命令关闭文件：

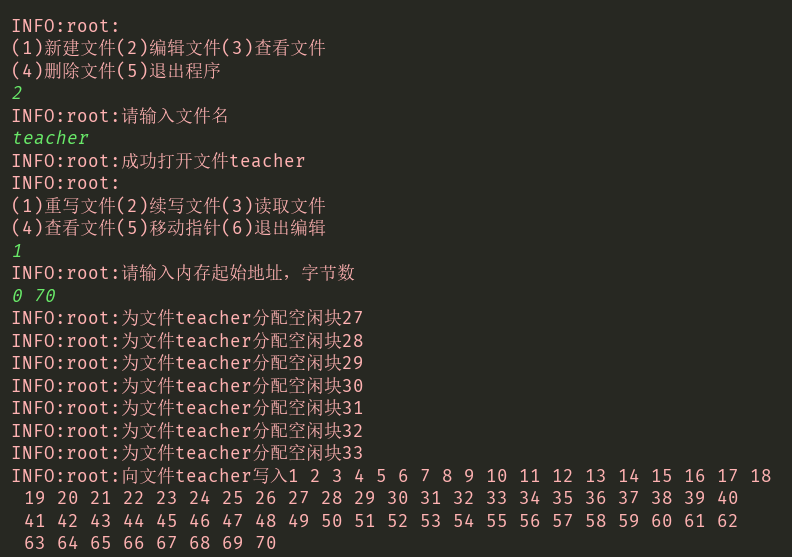


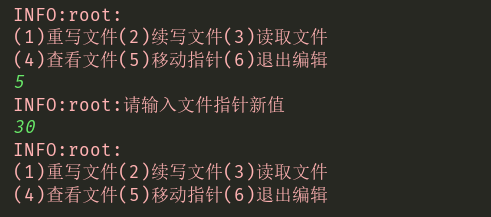
1. 查看文件：



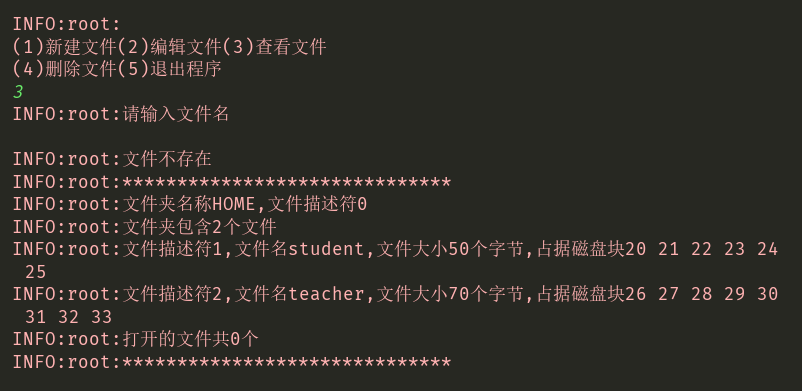
再按照刚才流程新建teacher文件，为文件分配70个字节空间：



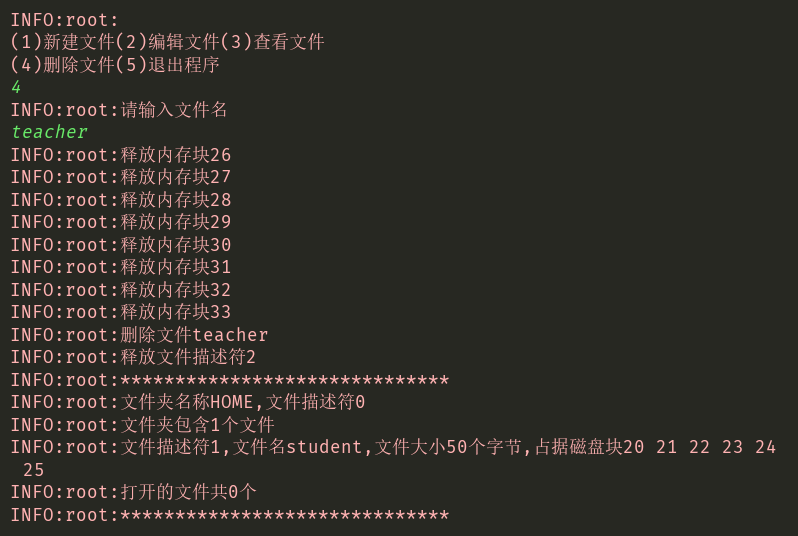




查看文件，发现当前文件夹包含两个文件，就是刚才建立的student和teacher



1. 删除文件



此时，位图生成情况如下：（程序目录下的disk.txt文件）

