文件系统管理项目

2152402 段婷婷

目录

1. 项目目的

- 理解文件存储空间的管理；
- 掌握文件的物理结构、目录结构和文件操作；
- 实现简单文件系统管理；
- 加深文件系统实现过程的理解；

2. 项目简述与功能

- 在内存中开辟一个空间作为文件存储器，在其上实现一个简单的文件系统，提供格式化、创建子目录、删除子目录、显示目录、更改当前目录、创建文件、打开文件、关闭文件、写文件、读文件、删除文件等操作；
- 退出这个文件系统时，需要该文件系统的内容保存到磁盘上，以便下次可以将其回复到内存中来。
- 文件逻辑结构采用无结构文件。
- 文件存储空间管理采取链接结构；
- 空闲空间管理可采用位图方法；
- 文件目录采用多级目录结构，目录项目中包含：文件路径名、创建时间等信息。

3. 项目设计

3.1 项目开发环境

- 系统：Windows 11 家庭中文版
- IDE：PyCharm 2022.3.3 (Community Edition)
- 语言：Python
- Python 解释器：通过 conda 部署 Python 环境，Python 版本为 3.8；通过 pip 安装 PyQt5

3.2 项目运行方式

- 直接运行：
    - 已经通过 pyinstaller 生成了 Windows 上的可执行文件。
    - 进入 My_FileSystem\dist\main 目录点击 main.exe，即可运行程序
- 编译运行
    - Python 版本：python 3.8
    - 安装 PtQt5（pip install PyQt5）

- 进入源码所在目录，运行源码（python main.py）

1.3 项目整体设计

- BLOCK.py：此文件定义磁盘的物理块，采用位图方法管理空闲空间。
- FAT.py：此文件定义 FAT（File Allocation Table）表，采取链接结构。
- FCB.py：此文件定义 FCB（File Control Block）。
- Catelog.py：此文件定义多级目录结点。
- MainForm.py：此文件定义了主窗口。
- EditForm.py：此文件定义了编辑文件窗口。
- AttributeForm.py：定义属性窗口
- File_Widget.py：实现列表控件。
- main.py：主函数。

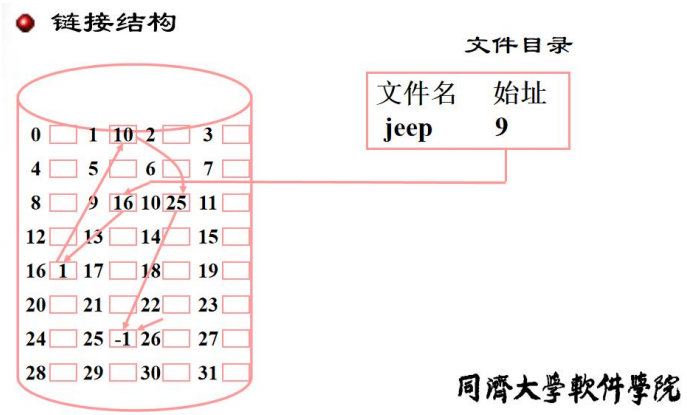
4. 文件结构

4.1 文件逻辑结构：无结构文件

- 采用无结构文件，即流式文件。
- 构成文件的基本单位是字符，文件是有逻辑意义、无结构的一串字符的集合。
- 优点:灵活性大。

4.2 文件物理结构：链接结构

- 示意图如下：



- 优点：
  ①提高了磁盘空间利用率,不存在外部碎片问题；
  ②有利于文件插入和删除；
  ③有利于文件动态扩充。
- 缺点：
  ①存取速度慢，不适于随机存取；
  ②可靠性问题，如指针出错；
  ③更多寻道次数和寻道时间；链接指针占用一定的空间。

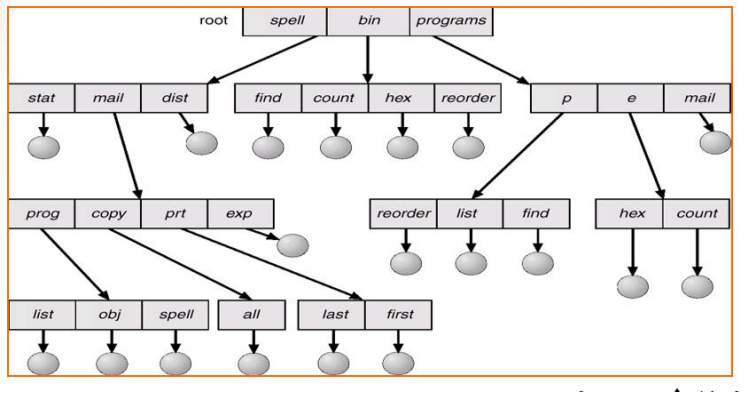
5. 目录结构

5.1 目录项：文件控制块（FCB）

- name：文件名
- createTime：创建时间
- createTime：最后修改时间
- self.start=-1：起始位置

5.2 目录结构：多级目录

- 示意图如下：



- 优点：

①提供更好的文件组织性，可以将文件按照不同的主题、类型或功能放置在不同的目录中；

②具有良好的可扩展性，可以通过创建新的目录来组织和管理文件；

③提供文件的隔离性，不同目录可以用于存储不同的文件类型或访问权限。

- 缺点：

①多级目录结构可能受到深度限制的限制，而且较深的目录结构可能会导致访问文件变得复杂，并且路径长度可能变得很长；

②查找效率问题，若文件被放置在深层目录中，会降低文件的查找效率；

③移动和重命名的复杂性，移动或重命名目录可能会导致文件路径的更改，这可能会影响到引用该文件的其他文件或应用程序。

5.3 文件访问方法：目录检索和文件寻址

本项目实现了目录检索和文件寻址两种文件访问方法：

- 目录检索：

① 在本项目中，用户通过选择窗口左侧的属性目录表来检索目录项。

② 根据路径名从根或当前目录开始检索：

- 文件寻址：

① 通过 FCB 中文件物理起始位置来定位和操作文件的实际数据；

② 直接访问文件的内容。

6. 空闲空间管理

本项目采用位示图的思想管理空闲空间，但是具体实现做了一些修改。具体来说，用一串位反映磁盘空间中的分配使用情况，每个物理块对应一位，空闲物理块为-2，否则为-1 或自然数。若为自然数，则表示的是链接的下一块的块号；若为-1，则表示后面没有链接的块，此块为结尾块。由此将位图管理空闲空间的方法和文件的链接结构结合了起来。

操作步骤：

- 申请物理块时，在位示图中查找为-2 的位，返回对应物理块号；
- 归还时，将对应位转置-2；

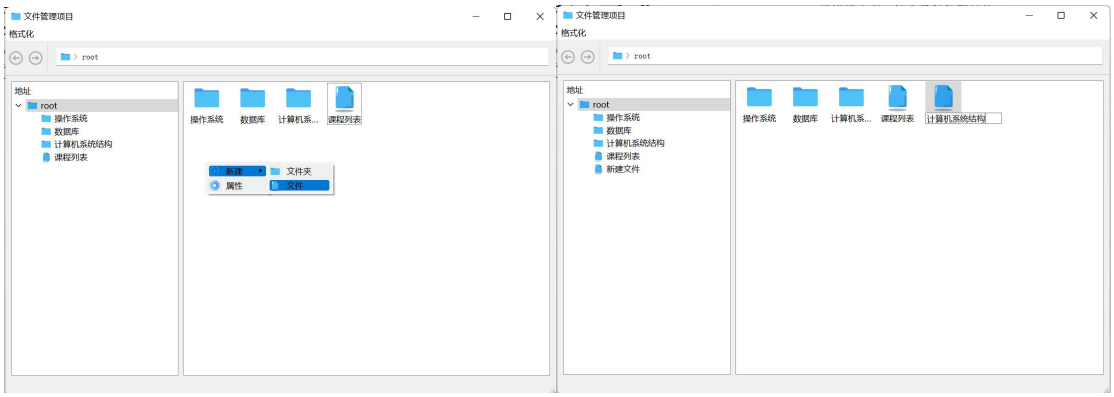
优点：

- 描述能力强，适合各种物理结构。

7. 文件操作

7.1 新建文件/文件夹

在空白处右键，选择新建-文件/文件夹。



实现步骤：

① 启动对新建文件夹项进行编辑，以允许用户输入自定义名称。

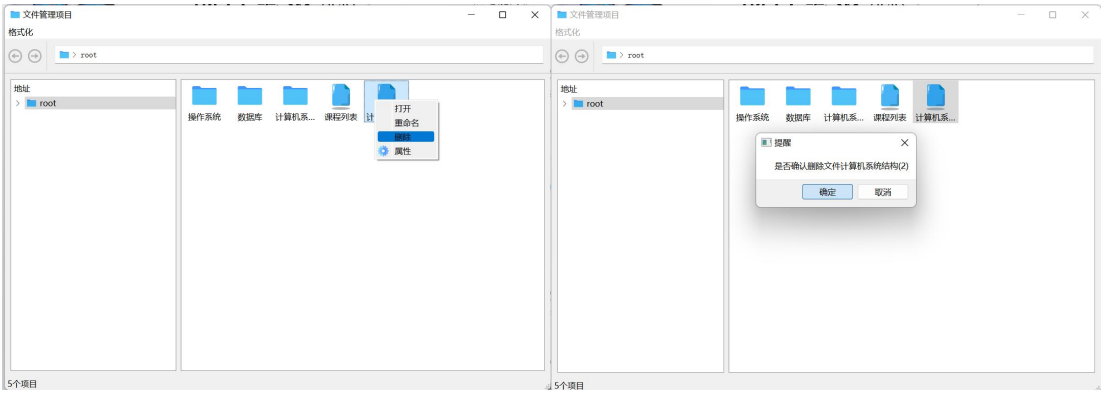
② 创建一个新的 CatalogNode 对象，表示新建的文件/文件夹，并将其添加到当前节点的子节点列表中。在创建 CatalogNode 对象的过程中，会创建新的 FCB。

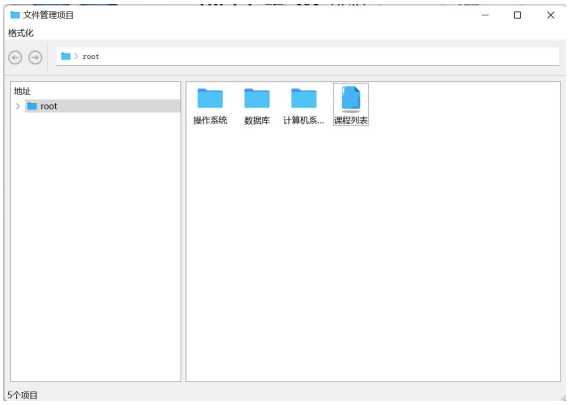
③ 将新的 CatalogNode 对象添加到目录表中。

④ 更新树形视图。

7.2 删除文件/文件夹

选择待删除的文件/文件夹，右键，选择删除。



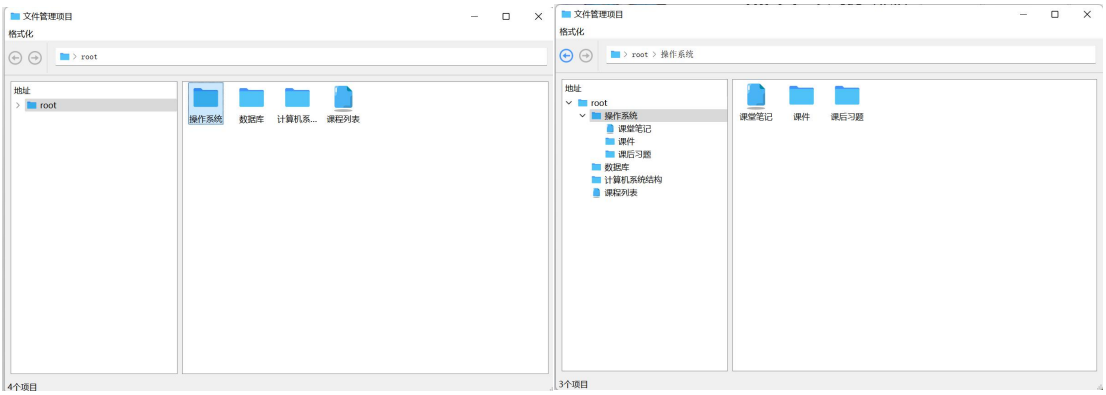


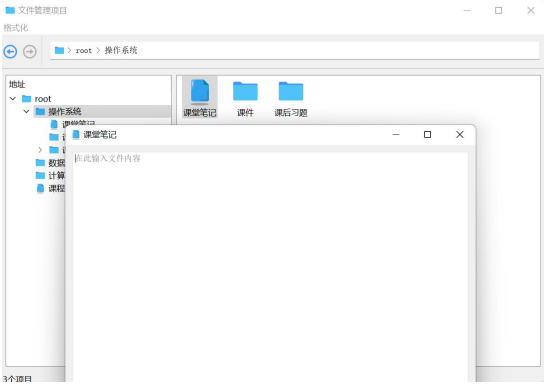
实现步骤：

① 检查是否有选中的文件项，如果没有则直接返回。
② 获取最后一个选中的文件项和其索引。
③ 创建一个消息框用于确认是否确定删除文件或文件夹。
④ 根据选中的文件项类型设置提示框的文本。
⑤ 显示提示框并等待用户的响应。
⑥ 如果用户点击了取消按钮，则返回。
⑦ 从列表视图中删除选中的文件项。
⑧ 删除文件项对象。
⑨ 递归地删除选中文件项的子文件或子文件夹。
⑩ 从当前节点的子节点列表中删除选中的文件项。
⑪ 更新目录结构表和树形视图。

7.3 查看文件/文件夹

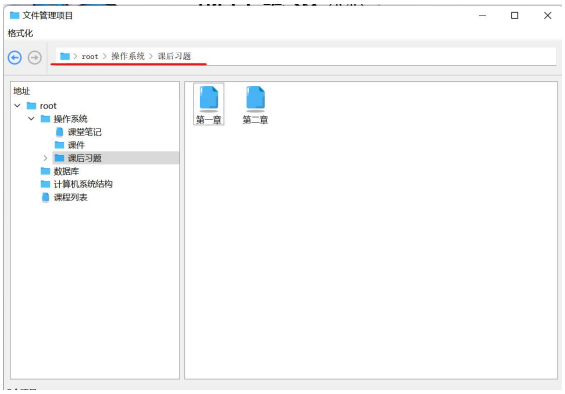
选择待查看的文件/文件夹，双击即可查看内容。





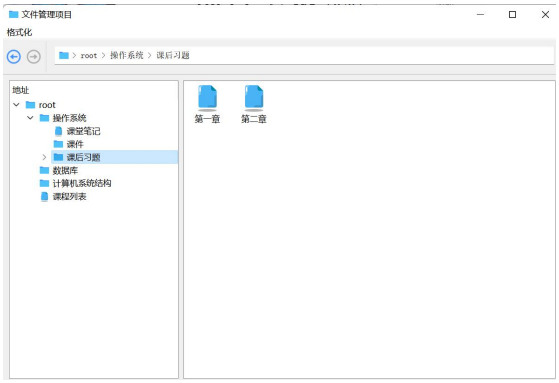
7.4 显示目录

目录显示在窗口顶部。



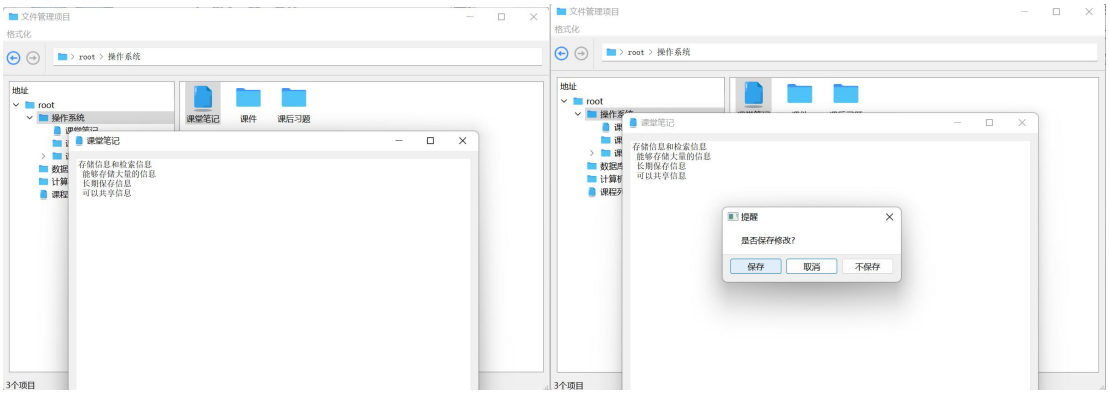
7.5 更改目录

单击左侧目录树的目录项，可以进入对应的文件夹。



7.6 写文件

双击文件打开后即可写文件，退出时选择保存即可。



此处阐述将数据写入磁盘的实现步骤：

① 调用文件分配表（FAT）对象的 write 方法来将数据写入磁盘。方法中的循环会不断执行以下操作，直到所有数据都被写入磁盘：

② 调用 findBlank 方法找到一个空闲的物理块（磁盘块）的索引 loca。

③ 如果 loca 的值为-1，表示磁盘空间不足，触发一个异常并打印错误信息，然后返回。

④ 如果 cur 不等于-1，表示当前不是第一个物理块，将 cur 对应的 FAT 表项更新为 loca，即当前块的下一个块是新找到的块 loca；如果 cur 等于-1，表示当前是第一个物理块，将 start 赋值为 loca。

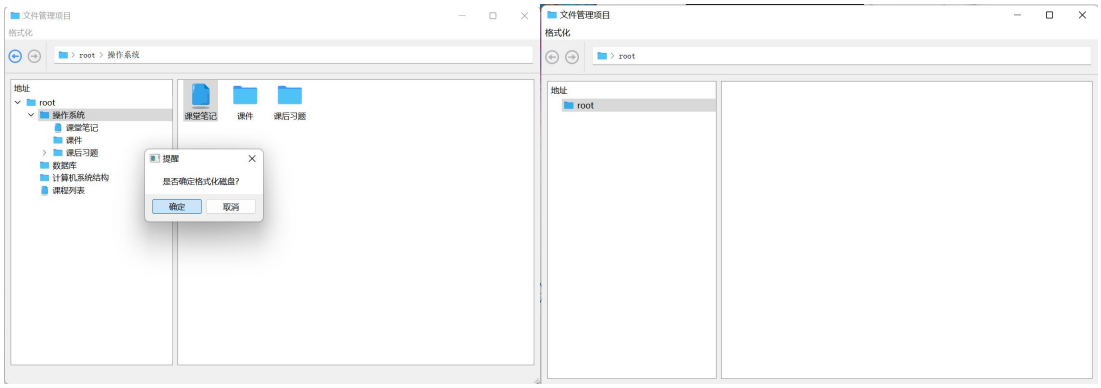
⑤ 将数据写入 disk[cur]对应的磁盘块，并将写入后剩余的数据赋值给 data。

⑥ 将 cur 对应的 FAT 表项设置为-1，表示当前块已被使用。

⑦ 回到循环开始，继续处理下一块数据。

7.6 格式化

点击顶部的”格式化”选项，选择确定即可格式化。



实现步骤：

① 结束编辑操作。

② 创建一个消息框用于确认是否确定格式化磁盘。

③ 如果用户点击了取消按钮，则返回。

④ 创建一个新的 FAT 对象，并将其写入 fat 文件。

⑤ 创建一个新的包含空块的磁盘列表，并将其写入 disk 文件。

⑥ 创建一个新的根目录节点，并将其写入 catelog 文件。

⑦ 隐藏当前窗口。

⑧ 创建一个 mainForm 窗口对象，并显示它。

8. 项目总结与心得

在这个项目中，我深入了解了文件系统的工作原理和设计思路。通过实践，我对无结构文件、文件的链接结构、多级目录结构和位示图管理空闲空间等方法有了更深入的理解。

文件系统的设计与实现需要考虑多个因素，包括文件的组织方式、目录结构、文件的链接方式等。在项目中，我学会了根据需求选择合适的文件系统结构，并实现其相应的功能。

通过使用 PyQt5 框架，我又一次实践了创建交互式的图形用户界面。

在项目实践中，我遇到了一些挑战和问题，但通过查找资料、调试和不断尝试，我能够克服困难并找到解决方案，锻炼了我的问题解决能力和自学能力。

通过这个项目，我不仅加深了对文件系统和 PyQt5 框架的理解，还提升了自己的编程技能和实践能力，受益匪浅。