**北 京 林 业 大 学**

**2023 — 2024学年第 2 学期 操作系统A 实习报告书**

专 业： 数字媒体技术 班 级： 数媒21-2

姓 名： 郑宛露 学 号： 211002525

实习地点： 学研T09 辅导教师： 孟伟

实习内容： 设计操作系统中的用户命令接口

实习环境： Visual Studio 2022

**目录**

[1 文件系统格式 4](#_Toc169597833)

[1.1 文件系统格式 4](#_Toc169597834)

[1.2 元数据存储 4](#_Toc169597835)

[1.3 访问控制 5](#_Toc169597836)

[1.4 文件存储 6](#_Toc169597837)

[2 功能实现 6](#_Toc169597838)

[2.1 文件操作命令 6](#_Toc169597839)

[2.1.1 创建文件功能 6](#_Toc169597840)

[2.1.2 删除文件功能 7](#_Toc169597841)

[2.1.3 打开文件功能 8](#_Toc169597842)

[2.1.4 关闭文件功能 8](#_Toc169597843)

[2.1.5 读文件功能 8](#_Toc169597844)

[2.1.6 写文件功能 9](#_Toc169597845)

[2.1.7 移动文件功能 9](#_Toc169597846)

[2.1.8 拷贝文件功能 10](#_Toc169597847)

[2.1.9 文件加锁功能 11](#_Toc169597848)

[2.1.10 显示文件前num行功能 11](#_Toc169597849)

[2.1.11 显示文件后num行功能 12](#_Toc169597850)

[2.1.12 文件指针移动功能 12](#_Toc169597851)

[2.2 目录管理操作命令 13](#_Toc169597852)

[2.2.1 进入目录功能 13](#_Toc169597853)

[2.2.2 显示当前目录功能 14](#_Toc169597854)

[2.2.3 创建目录功能 14](#_Toc169597855)

[2.2.4 删除目录功能 15](#_Toc169597856)

[2.3 虚拟磁盘与本地磁盘交互 15](#_Toc169597857)

[2.3.1 import功能 15](#_Toc169597858)

[2.3.2 export功能 16](#_Toc169597859)

[3 流程图 17](#_Toc169597860)

[3.1 创建文件 17](#_Toc169597861)

[3.2 删除文件 18](#_Toc169597862)

[3.3 打开文件 18](#_Toc169597863)

[3.4 关闭文件 18](#_Toc169597864)

[3.5 读文件 19](#_Toc169597865)

[3.6 写文件 19](#_Toc169597866)

[3.7 移动文件 19](#_Toc169597867)

[3.8 拷贝文件 20](#_Toc169597868)

[3.9 文件加锁 20](#_Toc169597869)

[3.10 显示文件前num行功能 20](#_Toc169597870)

[3.11 显示文件后num行功能 21](#_Toc169597871)

[3.12 指针移动功能 21](#_Toc169597872)

[3.13 进入目录功能 21](#_Toc169597873)

[3.14 显示当前目录功能 22](#_Toc169597874)

[3.15 创建目录功能 22](#_Toc169597875)

[3.16 删除目录功能 22](#_Toc169597876)

[3.17 import功能 23](#_Toc169597877)

[3.18 export功能 23](#_Toc169597878)

[4 主要数据结构、函数说明 24](#_Toc169597879)

[4.1 综合功能图 24](#_Toc169597880)

[4.2 主要数据组织结构 24](#_Toc169597881)

[4.2.1 FCB (File Control Block) 24](#_Toc169597882)

[4.2.2 FCBTree 25](#_Toc169597883)

[4.2.3 User 25](#_Toc169597884)

[4.3主要函数说明 26](#_Toc169597885)

[4.3.1 数据结构组织部分 26](#_Toc169597886)

[4.3.2 MyOs部分 27](#_Toc169597887)

[5 课程设计结果截图 28](#_Toc169597888)

[5.1 用户接口部分 28](#_Toc169597889)

[5.2 文件操作命令部分 29](#_Toc169597890)

[5.3 目录管理命令部分 30](#_Toc169597891)

[5.4 文件磁盘交互部分 30](#_Toc169597892)

[5.5 多线程功能 31](#_Toc169597893)

[6 课程设计分析及感受 32](#_Toc169597894)

[6.1 课设分析 32](#_Toc169597895)

[6.2 遇到的问题及解决办法 33](#_Toc169597896)

[6.3 收获与反思 33](#_Toc169597897)

[6.4 尚存在的不足 34](#_Toc169597898)

# 1 文件系统格式

## 1.1 文件系统格式

文件系统是操作系统的核心组成部分，负责数据的存储、检索和管理。

在本课程设计中，我才用了二级目录结构。二级目录结构是指在一级目录（根目录）下有多个子目录，每个子目录中又可以包含文件和进一步的子目录。这种结构通常用于组织大量文件，使得文件管理更加高效和便捷。

具体地，通过用户名，获取用户名下的所有文件及目录，进一步检索用户主目录下的子目录，结构及组织方式如图1.1.1所示。

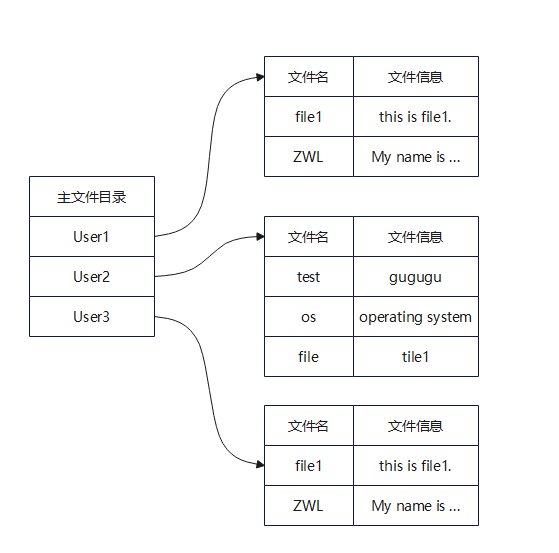


图1.1.1 文件系统格式示意

通过上述方式，将文件分层存放，可以有效地组织文件，避免平铺在一个目录下导致的混乱；可以根据需要在任意目录下创建新的子目录，灵活性和扩展性强；通过分层次存储文件，可以更快速地定位和访问所需文件，提高文件检索效率。

## 1.2 元数据存储

系统的元数据的存储和管理主要通过文件控制块（FCB）来实现，FCB块包含了文件或目录的关键元数据，如文件名、文件大小、创建和修改时间、权限设置以及锁定状态，这些元数据用于描述文件的基本属性和状态。为系统的各种操作提供接口。而FCB块的组织通过FCBTree结构，即根目录作为FCBTree的根节点，所有其他文件和目录作为子节点连接在树上。

元数据存储同时支持多用户操作，每个用户在系统中都有独立的文件空间。系统通过用户对象管理用户的文件列表，并在用户对象中存储每个文件的FCB;当用户登录系统时，系统就会加载用户的FCBTree结构，允许用户访问和管理他们的文件和目录。

FCB结构如图1.2.1所示。

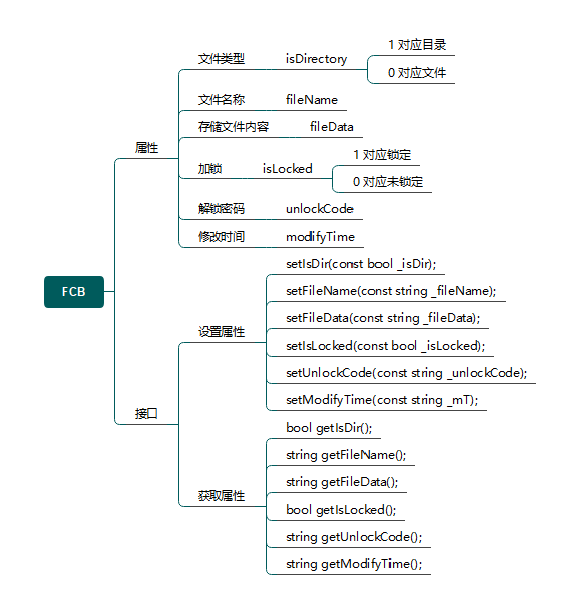


图1.2.1 FCB结构示意图

FCBTree结构如图1.2.2所示。

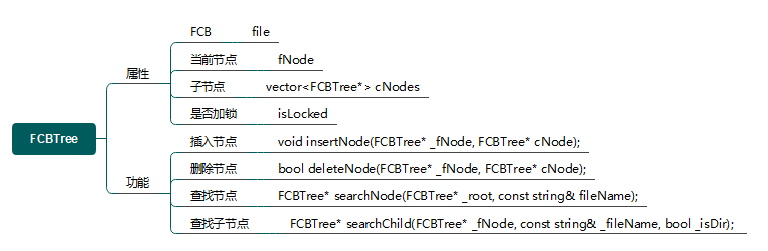


图1.2.2 FCBTree结构示意图

## 1.3 访问控制

访问控制主要涉及用户身份验证、文件加锁机制和多用户环境下的文件隔离三个方面。

首先，用户身份验证是通过定义User类来管理用户信息，并对用户身份进行验证。系统会使用一个对象类来管理每个用户的信息，包括用户名和密码。用户在系统登录时必须提供有效的用户名和密码。登录成功后，系统会加载用户的个人文件系统视图，该视图是基于用户权限和之前的会话状态动态生成的。

用户User类如图1.3.1所示。

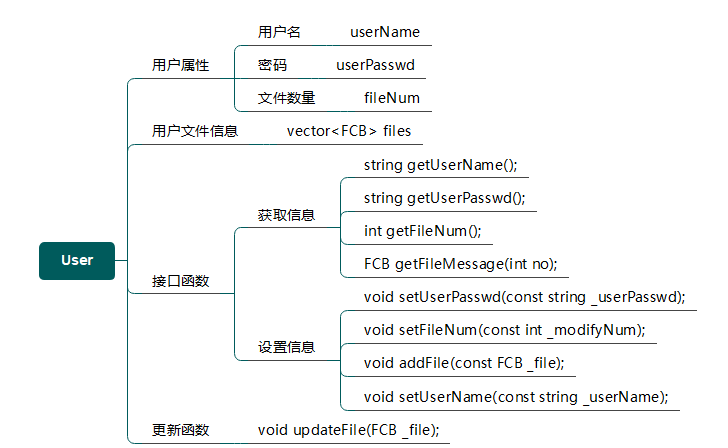


图1.3.1 User结构示意图

其次，加锁机制允许用户对文件进行加锁操作，以防止未经授权的访问和修改。当文件被锁定时，其他用户的访问请求将被暂时阻止，直到文件被解锁。

最后，在多用户环境下，我采用文件锁和用户会话来隔离不同用户的操作，以防止意外的数据覆盖。文件锁可以确保同一时间只有一个用户可以对文件进行写操作，而其他用户只能进行读操作。用户会话则用于跟踪和管理每个用户的操作状态，确保彼此之间的操作互不干扰。

## 1.4 文件存储

文件存储主要负责管理文件的实际内容（对于文件）或指向子目录和文件的链接（对于目录）。在系统中，使用文件控制块（FCB）来存储重要的文件信息。

具体地，我将文件的数据直接存储在 FCB 的 fileData 字段中，按照一定顺序，用特殊字符‘~’隔开，将文件信息存储在如图1.4.1所示的二进制文件data.bin中，并通过FCB中的指针或索引来引用这些数据。

这种设计的优势是可以简化文件的读写操作。当需要访问文件内容时，系统可以通过FCB中的指针或索引直接定位到data.bin中相应的位置，从而获取到文件的数据。

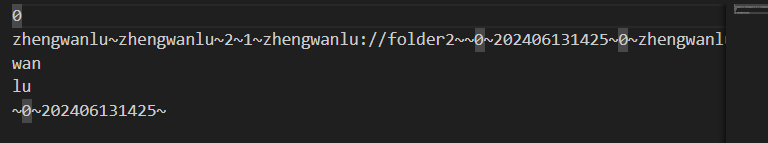


图1.4.1 二进制文件内容示意

# 2 功能实现

## 2.1 文件操作命令

### 2.1.1 创建文件功能

（1）功能描述：

设计文件创建函数createFile实现此功能。

通过用户在命令行输入 create 命令触发，实现创建一个新文件并将其添加到文件系统的当前节点下。

系统首先提示用户输入文件名并检查当前目录下是否已存在同名文件，若不存在，则允许用户输入文件内容，系统会创建一个新的文件控制块（FCB），并将其添加到当前目录的 FCB 树中，同时设置文件的元数据如创建时间和修改时间。

（2）代码截图：

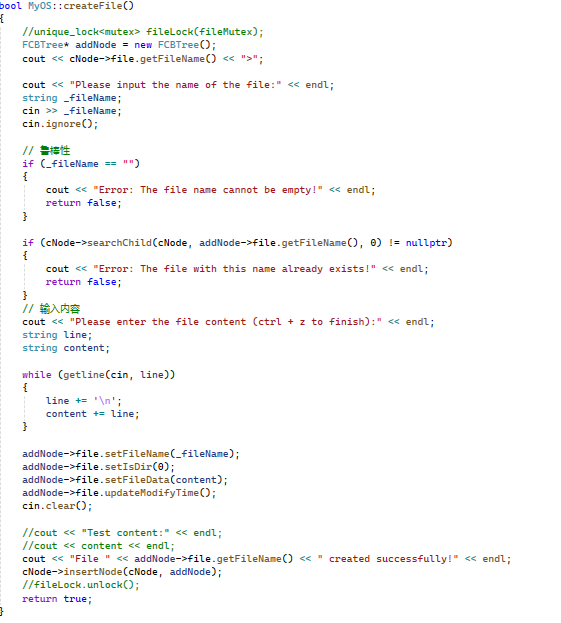


图2.1.1 创建文件功能代码

### 2.1.2 删除文件功能

（1）功能描述：

设计文件删除函数deleteFile实现此功能。

主要功能是删除指定路径下的文件。根据给定的路径查找要删除的文件节点，检查文件是否存在以及是否有其他用户正在修改该文件（多线程），如果文件存在且没有其他用户正在修改，执行删除操作并更新相关信息。

（2）代码截图：



图2.1.2 删除文件功能代码

### 2.1.3 打开文件功能

（1）功能描述：

设计打开文件函数openFile实现此功能。

主要功能是打开指定路径下的文件，并允许用户进行编辑操作。首先根据给定的路径查找要打开的文件节点，检查文件是否存在、是否已经被其他用户打开或正在编辑，如果文件未被锁定，允许用户进行编辑操作，并在用户退出编辑后保存数据，在编辑过程中，根据用户输入执行相应的文件操作。

（2）代码截图：

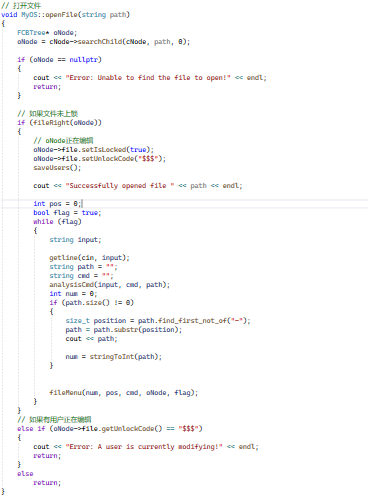


图2.1.3 打开文件功能代码

### 2.1.4 关闭文件功能

（1）功能描述：

设计关闭文件函数closeFile实现此功能。

主要功能是关闭给定文件节点所代表的文件，以及相应地更新文件的锁定状态和解锁码。将文件的锁定状态设置为未锁定，清空文件的解锁码，最后输出文件已关闭的提示信息。

（2）代码截图：

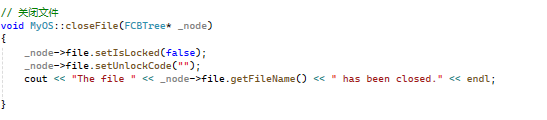


图2.1.4 关闭文件功能代码

### 2.1.5 读文件功能

（1）功能描述：

设计读文件函数readFile实现此功能。

主要功能是从文件中读取数据并输出到控制台。获取要读取的字符数量 num 并初始化读取位置 pos；从文件的当前读取位置 pos 开始读取 num 个字符，更新读取位置 pos，确保不会超过文件数据的总长度，将读取到的数据逐行输出到控制台。

（2）代码截图：

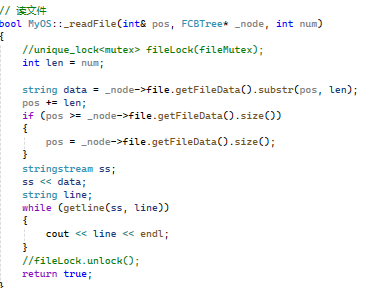


图2.1.5 读文件功能代码

### 2.1.6 写文件功能

（1）功能描述：

设计写文件函数writeFile实现此功能。

从当前文件位置 pos 开始写入用户输入的数据，并更新文件内容和修改时间。这个函数主要实现以下几点：读取文件到指定位置的内容：获取文件从头到当前位置 pos 的内容；接收用户输入：从控制台接收用户输入数据，直到用户使用 ctrl + z 结束输入；更新文件内容：将用户输入的数据追加到之前读取的文件内容后面，并更新文件的数据；更新文件修改时间：调用方法更新文件的最后修改时间；输出成功信息：提示用户写入成功。

（2）代码截图：

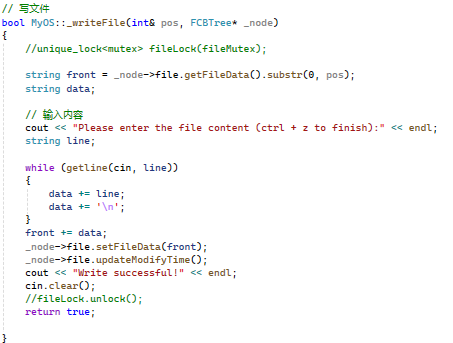


图2.1.6 写文件功能代码

### 2.1.7 移动文件功能

（1）功能描述：

设计moveFile 函数移动文件到指定路径。

首先提取文件名和目标路径，从输入的命令字符串中提取文件名和目标路径，调用 analysisCmd 函数对提取出的部分进行解析，获取文件名和目标路径。

搜索要移动的文件，调用 searchChild 函数在当前节点下搜索要移动的文件，如果找到则继续执行，否则输出错误信息并结束操作。

根据提取的目标路径进行解析，并逐层查找对应的节点，直到找到目标位置或者发现无效路径。检查目标路径下是否存在同名文件，如果存在则输出错误信息并结束操作。将文件节点插入到目标路径下，然后在原路径中删除该文件节点。更新修改计数器，保存用户信息，然后结束操作。

（2）代码截图：

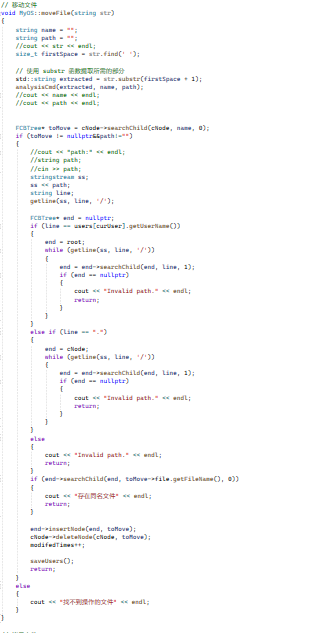
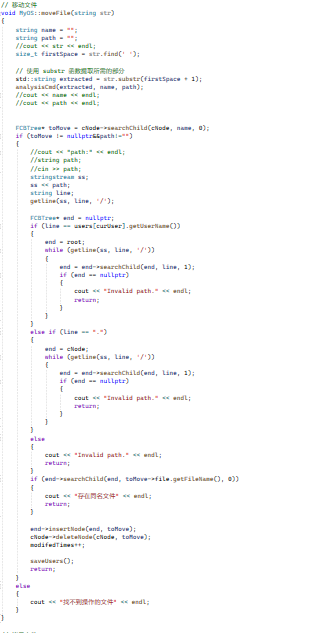


图2.1.7 移动文件功能代码

### 2.1.8 拷贝文件功能

（1）功能描述：

copyFile 函数的主要目的是拷贝一个文件，并赋予其新名称。

提示用户输入需要被拷贝的文件名，在当前目录（节点）下搜索该文件。如果找到了要拷贝的文件，则创建一个新的文件控制块（FCB）节点，并复制文件内容。提示用户输入新文件的名称。检查新文件名是否有效（即在当前目录下是否有同名文件）：如果没有同名文件，则将新节点插入当前目录；如果有同名文件，则输出错误信息并退出。

（2）代码截图：

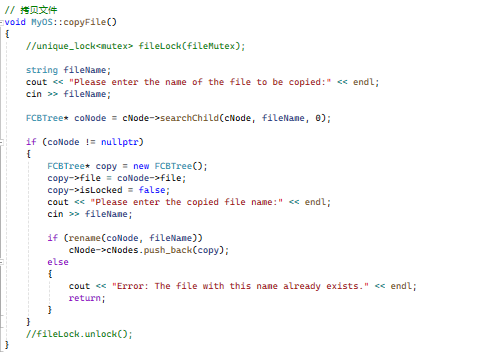


图2.1.8 拷贝文件功能代码

### 2.1.9 文件加锁功能

（1）功能描述：

设计lockFile 函数的主要目的是对文件进行加锁或解锁操作。

提示用户选择要加锁或解锁的文件类型，是目录（文件夹）还是文件。提示用户输入要加锁或解锁的文件名，使用 searchChild 函数在当前目录（节点）下搜索指定文件名的文件节点，如果找到了该文件节点，则指针 lNode 将指向该节点，否则为空。

如果找到了文件节点 lNode：如果文件未加锁，则提示用户输入文件密码，并将文件的加锁状态设置为已加锁，保存密码。如果文件已加锁，则提示用户输入文件密码，并验证密码是否正确。如果密码正确，则将文件的加锁状态设置为未加锁，并清除密码。如果没有找到文件节点，输出错误信息。

调用 saveUsers 函数保存用户信息（可能与文件加锁相关）。

（2）代码截图：

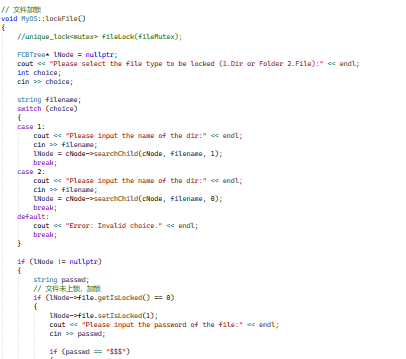
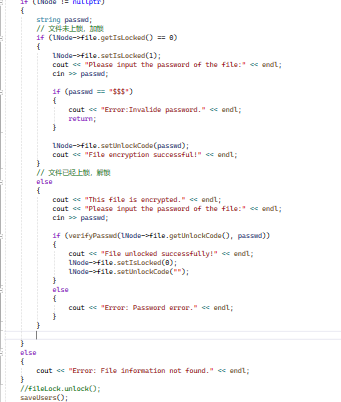
 

图2.1.9 文件加锁功能代码

### 2.1.10 显示文件前num行功能

（1）功能描述：

设计headFile 函数，从给定的文件节点 \_node 中读取文件数据，并显示文件的前 num 行内容。

通过文件节点 \_node 调用 getFileData 函数获取文件的全部数据，并存储在字符串变量 data 中。使用 stringstream 对象 ss 将文件数据 data 进行处理，以便逐行读取，使用 while 循环结合 getline 函数，逐行读取文件数据，并显示每一行的内容，在每次读取一行后，将 num 减一，直到 num 变为 0 或者已经读取完所有行。

（2）代码截图：

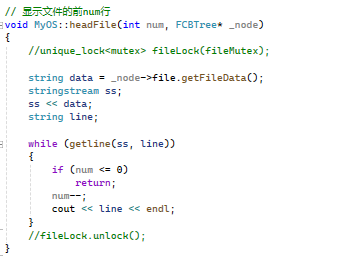


图2.1.10 显示文件前num行功能代码

### 2.1.11 显示文件后num行功能

（1）功能描述：

设计tailFile 函数从给定的文件节点 \_node 中读取文件数据，并显示文件的最后 num 行内容。

通过文件节点 \_node 调用 getFileData 函数获取文件的全部数据，并存储在字符串变量 data 中，使用 stringstream 对象 ss 将文件数据 data 进行处理，以便逐行读取，使用另外一个 stringstream 对象 \_ss 复制一份文件数据，用于最后的逐行读取。使用第一个 while 循环结合 getline 函数，统计文件的总行数并保存在变量 len 中；使用第二个 while 循环结合 getline 函数，逐行读取文件数据，并显示最后 num 行的内容，在每次读取一行后，将 len 减一，直到 len 变为 0 或者已经读取完所有行，此时即可退出循环。

（2）代码截图：

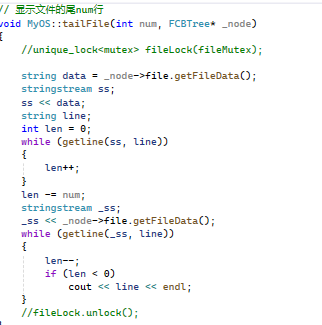


图2.1.11 显示文件后num行功能代码

### 2.1.12 文件指针移动功能

（1）功能描述：

设计lseek 函数通过获取用户输入的偏移量来调整文件读写指针的位置，使其在合法范围内移动。

提示用户输入偏移量并读取输入，计算新的指针位置，并确保指针位置不超出文件数据的范围。通过标准输入 cin 获取用户输入的偏移量 offset，将当前的文件读写指针 pos 加上用户输入的 offset，得到新的指针位置，如果新的指针位置小于 0，则将指针位置设为 0（防止指针位置越界），如果新的指针位置大于文件数据的大小，则将指针位置设为文件数据的大小（防止指针位置超过文件末尾）。

（2）代码截图：

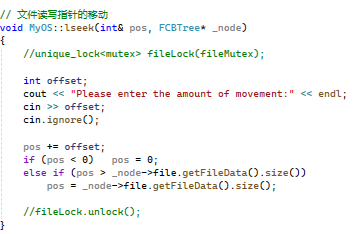


图2.1.12 文件指针移动功能代码

## 2.2 目录管理操作命令

### 2.2.1 进入目录功能

（1）功能描述：

设计了 cd 函数，用于在自定义的操作系统中切换当前目录。

cd 函数的主要功能是根据用户输入的路径来切换当前目录。返回上一级目录（如果输入路径为 ".."）；或切换到指定子目录（如果输入路径有效）。

（2）代码截图：

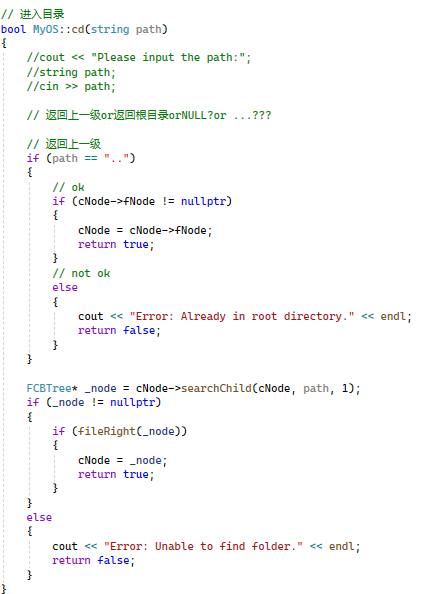


图2.2.1 进入目录功能代码

### 2.2.2 显示当前目录功能

（1）功能描述：

设计dir ()函数，用于在自定义的操作系统中显示当前目录下的文件和子目录信息。

dir 函数的主要功能是遍历当前目录下的文件和子目录，并输出它们的修改时间、类型（文件或目录）以及名称。使用循环遍历当前目录 cNode 下的所有子节点，对于每个子节点，输出其修改时间、类型和名称。输出格式包括修改时间、类型（文件或目录）和名称，以及换行符进行分隔。

（2）代码截图：

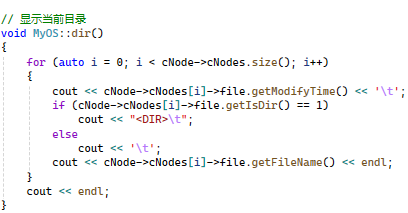


图2.2.2 显示当前目录功能代码

### 2.2.3 创建目录功能

（1）功能描述：

设计mkdir() 函数，用于在自定义的操作系统中创建目录。

mkdir 函数的主要功能是在当前目录下创建一个新的子目录，并要求用户输入新目录的名称。创建一个新的 FCBTree 对象 dNode，该对象用于表示新的子目录，设置新目录的属性，包括目录标识符、名称等。提示用户输入新目录的名称，检查当前目录下是否已存在同名的子目录，如果存在则输出错误信息并结束操作；否则继续创建新目录。将新创建的子目录节点插入到当前目录的节点列表中，作为当前目录的子节点。

（2）代码截图：

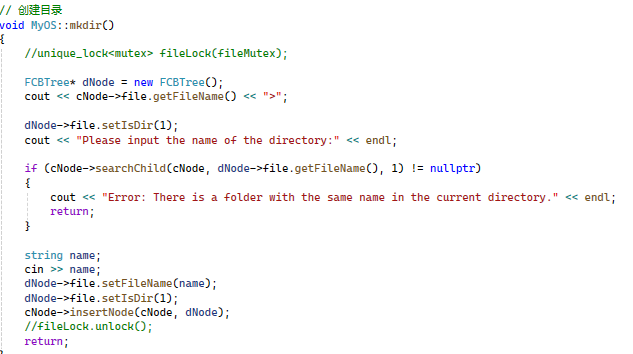


图2.2.3 创建目录功能代码

### 2.2.4 删除目录功能

（1）功能描述：

设计rmdir ()函数，用于在自定义操作系统中删除指定的目录。

rmdir 函数的主要功能是删除指定的目录，并释放其占用的资源。根据用户输入的路径，查找指定的目录节点，如果找到了对应的目录节点，则执行后续的删除操作；否则输出错误信息并结束操作。调用 deleteNode 方法删除找到的目录节点，输出文件删除成功的提示信息。对当前用户的数据进行清空操作，更新用户的文件数量信息。

（2）代码截图：



图2.2.4 删除目录功能代码

## 2.3 虚拟磁盘与本地磁盘交互

### 2.3.1 import功能

（1）功能描述：

设计importFile ()函数，用于在自定义操作系统中导入文件。

importFile 函数的主要功能是根据用户提供的文件路径，将文件内容导入到操作系统中，并创建相应的文件节点。根据用户输入的文件路径，提取文件名和路径信息，使用输入的文件路径创建并打开文件流，以便读取文件内容，检查文件是否成功打开，若打开失败则输出错误信息并结束操作；根据文件路径和名字创建一个新的文件节点；将文件内容读取到内存中，并设置到文件节点的属性中；调用 insertNode 方法将新创建的文件节点插入到当前目录节点中，如果同名文件已存在，则输出错误信息并结束操作。

（2）代码截图：

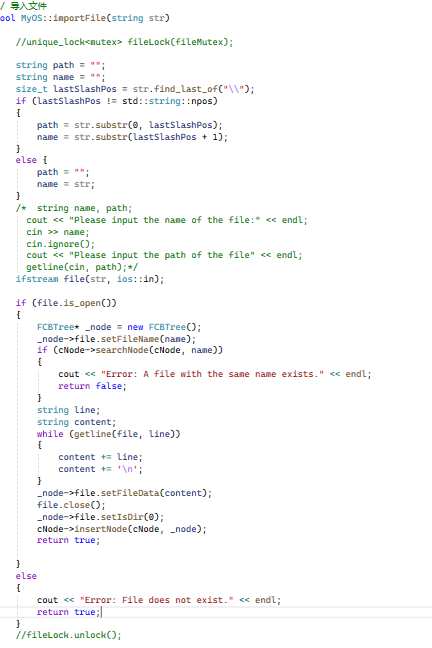


图2.3.1 import功能代码截图

### 2.3.2 export功能

（1）功能描述：

设计exportFile 函数，用于在自定义操作系统中导出文件内容到指定路径的功能。

exportFile 函数的主要功能是根据用户提供的文件名和导出路径，将指定文件的内容导出到指定路径中。根据用户输入的命令，提取文件名和导出路径信息，使用文件名在当前目录中查找对应的文件节点。如果文件节点不存在，则输出错误信息并结束操作，检查文件节点是否被其他用户锁定，若被锁定则输出错误信息并结束操作。打开指定路径的文件流，若打开失败则输出错误信息并结束操作，将文件节点中的内容写入到文件流中，并关闭文件流。

（2）代码截图：

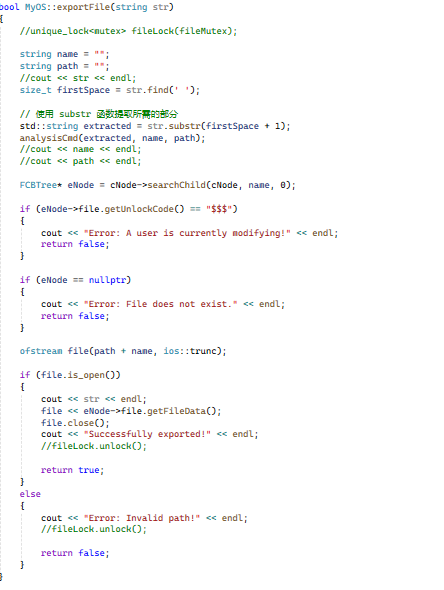


图2.3.2 export功能代码截图

# 3 流程图

## 3.1 创建文件

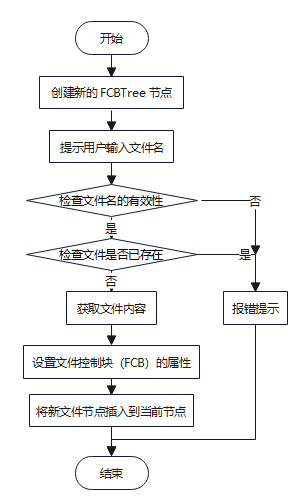
****

图3.1 创建文件流程图

## 3.2 删除文件

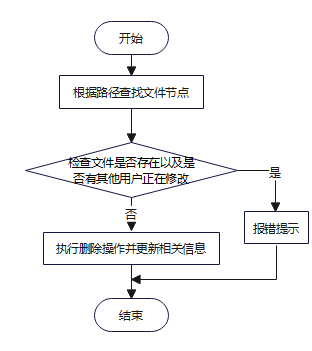
****

图3.2 删除文件流程图

## 3.3 打开文件

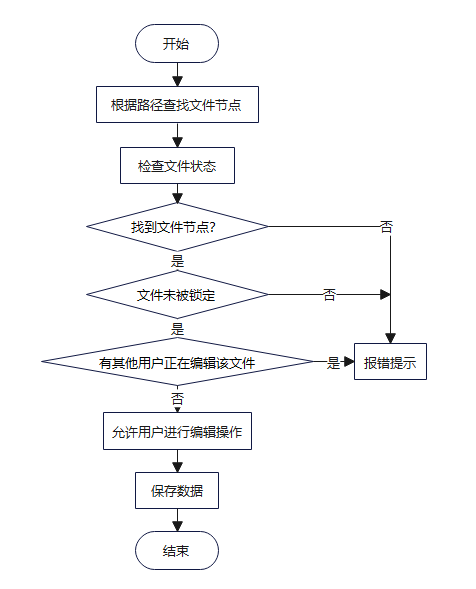
****

图3.3 打开文件流程图

## 3.4 关闭文件

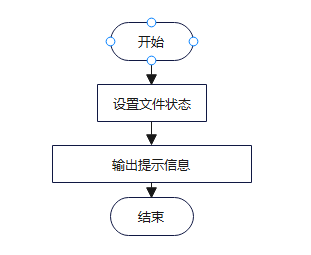


图3.4 关闭文件流程图

## 3.5 读文件

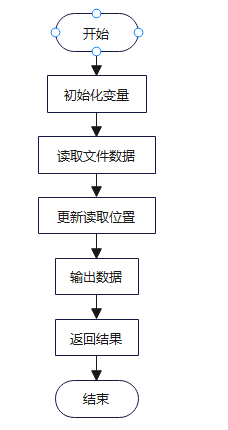


图3.5 读文件流程图

## 3.6 写文件

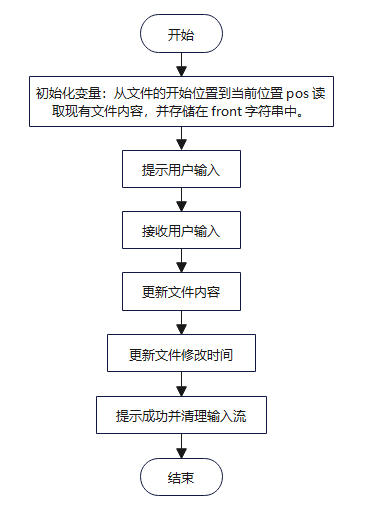


图3.6 写文件流程图

## 3.7 移动文件

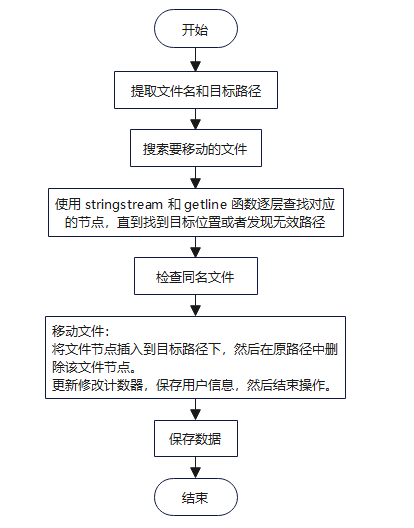


图3.7 移动文件流程图

## 3.8 拷贝文件

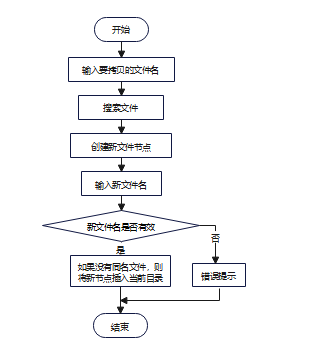


图3.8 拷贝文件流程图

## 3.9 文件加锁

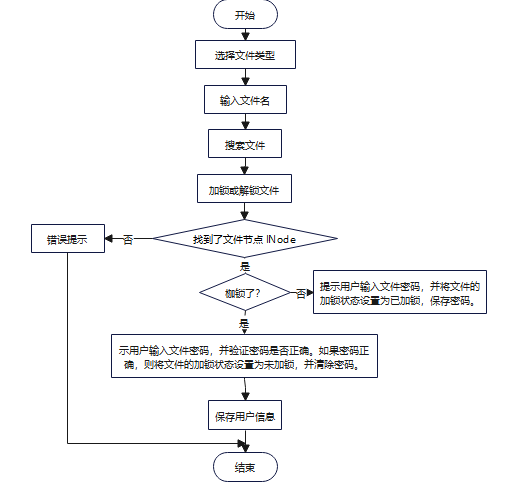


图3.9 文件加锁流程图

## 3.10 显示文件前num行功能

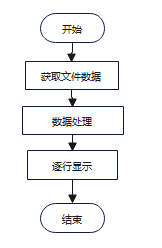


图3.10 显示文件前num行功能流程图

## 3.11 显示文件后num行功能

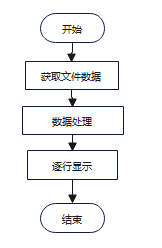


图3.11 显示文件后num行功能流程图

## 3.12 指针移动功能

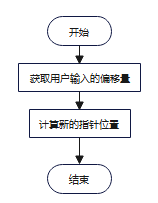
****

图3.12 指针移动功能流程图

## 3.13 进入目录功能

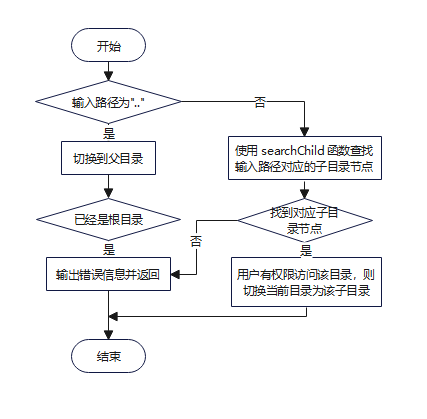


图3.13 进入目录功能流程图

## 3.14 显示当前目录功能

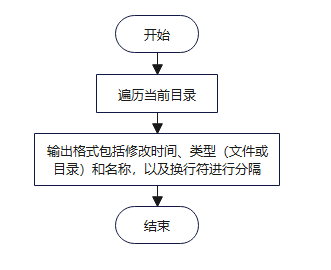


图3.14 显示当前目录功能流程图

## 3.15 创建目录功能

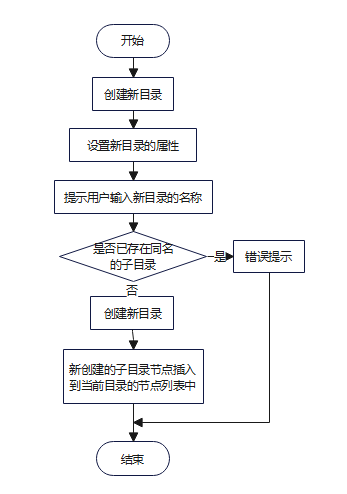


图3.15 创建目录功能流程图

## 3.16 删除目录功能

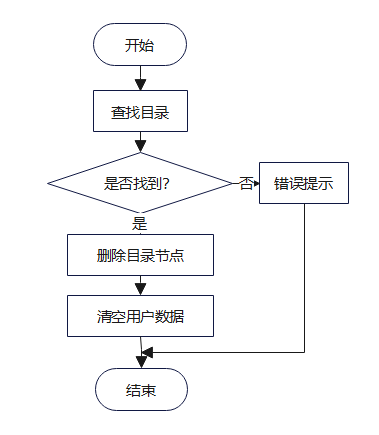


图3.16 删除目录功能流程图

## 3.17 import功能

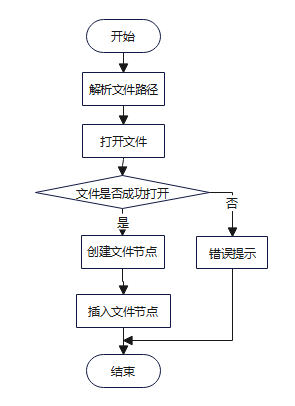


图3.17 import功能流程图

## 3.18 export功能

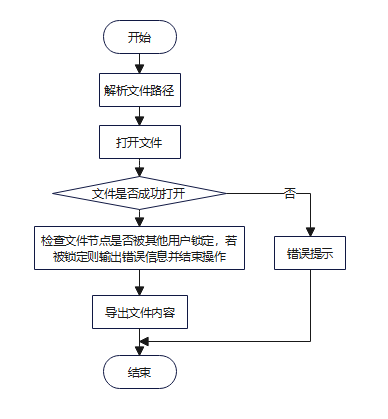


图3.18 export功能流程图

# 4 主要数据结构、函数说明

## 4.1 综合功能图



图4.1 综合功能示意

## 4.2 主要数据组织结构

### 4.2.1 FCB (File Control Block)

（1）设计思路：

FCB 作为文件控制块，用于描述文件和目录的各种属性，每个 FCB 实例存储了关键的文件信息，包括文件名、文件数据、目录标志、锁定状态、修改时间和解锁码。

其中，文件名 (fileName) 标识文件或目录的名称，用于查找和访问；文件数据 (fileData) 对于文件存储实际内容，而对于目录通常为空；目录标志 (isDirectory) 是一个布尔值，指示当前 FCB 是文件还是目录；锁定状态 (isLocked) 表示文件是否被锁定，以便控制并发访问；修改时间 (modifyTime) 记录文件或目录的最后修改时间；解锁码 (unlockCode) 在文件被加锁时存储解锁所需的密码。

为了管理这些属性，FCB 提供了一系列方法，例如 setFileName 和 getFileName 用于设置和获取文件名，setFileData 和 getFileData 用于设置和获取文件内容，setIsDir 和 getIsDir 用于设置和获取目录标志，setIsLocked 和 getIsLocked 用于设置和获取锁定状态，setModifyTime 和 getModifyTime 用于设置和获取修改时间，setUnlockCode 和 getUnlockCode 用于设置和获取解锁码。

（2）主要代码及注释：

FCB结构代码如图4.2.1所示。



图4.2.1 FCB代码

### 4.2.2 FCBTree

（1）设计思路：

FCBTree 用于表示文件系统的目录结构，是一个树状结构，每个节点都包含一个 FCB 实例以表示文件或目录。

为了在文件系统中执行插入、删除、查找等操作，FCBTree 提供了 insertNode、deleteNode、searchChild 和 searchNode 方法，其中 insertNode 用于在指定父节点下插入子节点，deleteNode 用于从父节点中删除子节点，searchChild 用于在父节点下搜索名为 name 的子节点，并指明是否为目录，searchNode 用于在整个树结构中搜索名为 name 的节点。

（2）主要代码及注释：

FCBTree结构代码如图4.2.2所示。

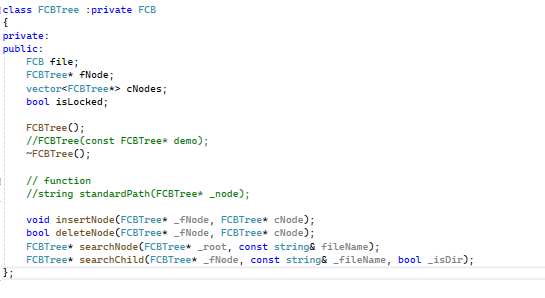


图4.2.2 FCBTree代码

### 4.2.3 User

（1）设计思路：

User 代表系统中的一个用户，它管理用户的身份认证信息和与该用户相关的文件列表，从而确保用户只能访问自己的数据。

User 包含了用户名 (userName)、密码 (userPasswd) 以及文件列表 (files) 等属性，其中文件列表存储了与用户相关联的 FCB 对象。

为了管理用户及其文件，User 提供了 getUserName 和 getUserPasswd 方法用于返回用户名和密码，setFileNum 和 getFileNum 方法用于设置和获取用户文件数量，addFile 方法用于向用户文件列表添加一个文件，clearData 方法用于清除用户的文件列表。

（2）主要代码及注释：

User结构代码如图4.2.3所示。



图4.2.3 User代码

## 4.3主要函数说明

### 4.3.1 数据结构组织部分

（1）Getter和 Setter方法

对于 FCB 对象的属性管理，getter 和 setter 方法是基础。

getter 方法getFileName()、 getFileData()用于读取 FCB 对象中的属性值。这些方法分别返回文件名、文件数据、目录标志、锁定状态、解锁码和最后修改时间，确保外部可以方便地获取对象的内部状态。

相应的 setter 方法setFileName() 、setFileData()，允许修改这些属性值，每个方法接受一个参数，用于设置 FCB 对象的相应属性。

通过 getter 和 setter 方法的组合，系统能够有效地管理和访问 FCB 对象的各种属性，从而提供良好的数据封装和接口设计。

（2）在目录树中插入节点

设计insertNode(FCBTree\* \_fNode, FCBTree\* cNode) 方法用于在指定的父节点下插入子节点 cNode。

核心代码 \_fNode->cNodes.push\_back(cNode)实现了将子节点插入到父节点的子节点列表中，并将 cNode 的父节点设置为 \_fNode。插入操作完成后，还需要更新 cNode 的文件修改时间，以反映最新的目录结构变动。这种设计确保了目录树的结构化管理，便于后续的目录操作和维护。

对应函数部分如图4.3.1所示。



图4.3.1 部分功能函数

（3）从目录树中删除节点

对于删除操作，deleteNode(FCBTree\* \_fNode, FCBTree\* cNode) 方法在父节点 \_fNode 的子节点列表中查找并删除指定的子节点 cNode。

找到后将其从列表中移除。由于删除操作仅影响被删除的节点，不涉及剩余节点的时间戳，因此不更新父节点的修改时间。这种处理方式确保了目录树的一致性，并能有效管理目录下的文件和子目录。

（4）递归搜索节点

searchNode(FCBTree\* \_root, const string& \_fileName) 方法用于在指定的根节点 \_root 开始的子树中递归搜索文件名为 \_fileName 的节点。该方法通过递归遍历所有子节点，逐一比较每个节点的文件名。如果找到匹配的节点则返回该节点，若未找到则返回空。这种递归搜索机制能够有效地在复杂的目录树中定位指定的文件或目录，方便用户对文件系统进行管理和操作。

（5）搜索子节点

searchChild(FCBTree\* \_fNode, const string& \_fileName, bool \_isDir) 方法用于在父节点 \_fNode 的直接子节点中搜索文件名为 \_fileName 且类型匹配 \_isDir 的节点。实现方法是遍历 \_fNode 的所有子节点，比较每个节点的文件名和目录标志。如果找到符合条件的节点，则返回该节点；否则返回空。这种精确的搜索方式有助于快速查找父目录下的特定文件或子目录，提高目录操作的效率。

### 4.3.2 MyOs部分

（1）核心管理函数

主要包括MyOS::kernel() 和 MyOS::updateData()。

MyOS::kernel() 作为文件系统的主调度器，用于处理所有命令行输入的命令。该函数运行在一个无限循环中，等待命令变量 CMD 被设置。利用互斥锁 m 和条件变量 cv 来确保命令处理的同步性。根据用户输入的命令字符串，通过 stringToEnum 函数转换为枚举类型，进行分支处理，调用相应的处理函数以完成创建、删除、打开等各种文件操作。

MyOS::updateData() 用于刷新并重载用户和文件系统数据。首先清空当前的用户列表和文件系统结构，然后重新从存储介质加载数据，以确保数据的一致性和系统的稳定性。



图4.3.1 updateData函数截图

（2）用户和会话管理函数

initMenu()，userLogin()，createUser() 和 saveUsers() 组成。

MyOS::initMenu() 初始化并显示用户登录或注册的菜单。根据用户的选择调用 userLogin(), createUser(),最终函数通过返回当前用户的索引来设置当前用户状态。

MyOS::userLogin() 处理用户登录逻辑，提示用户输入用户名和密码，然后遍历已注册的用户列表，检查输入的用户名和密码是否与现有用户匹配

MyOS::createUser() 负责注册新用户。提示用户输入新的用户名和密码，然后检查新用户名是否已存在于用户列表中，确保用户名的唯一性。注册成功后，用户可以选择直接登录或返回主菜单。

MyOS::saveUsers() 则保存用户信息到文件。遍历用户列表，序列化每个用户的信息（用户名和密码）并写入指定的数据文件。

# 5 课程设计结果截图

## 5.1 用户接口部分

用户接口部分为登入界面，注册界面，登录界面，help帮助信息，结果截图如图5.1所示，顺序为从左到右，从上到下

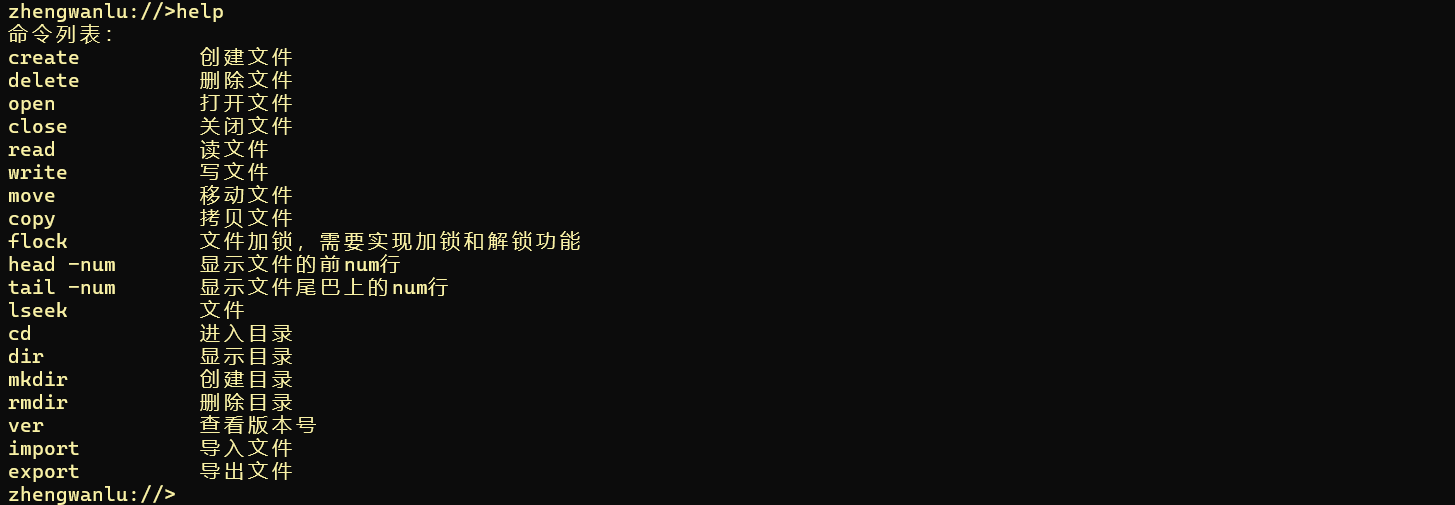


图5.1 用户接口部分结果图

## 5.2 文件操作命令部分

登入界面后创建文件和目录，用dir命令显示结果，并打开文件结果截图如图5.2所示。

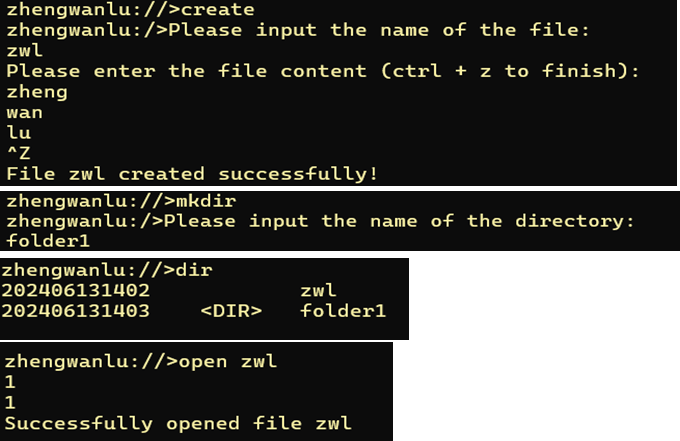


图5.2 文件操作命令结果

打开文件后同样对应有help帮助，进行读写文件操作后查看开头和结尾，移动指针继续操作文件后并关闭，再次打开未open就write/的提示。结果如图5.3所示。

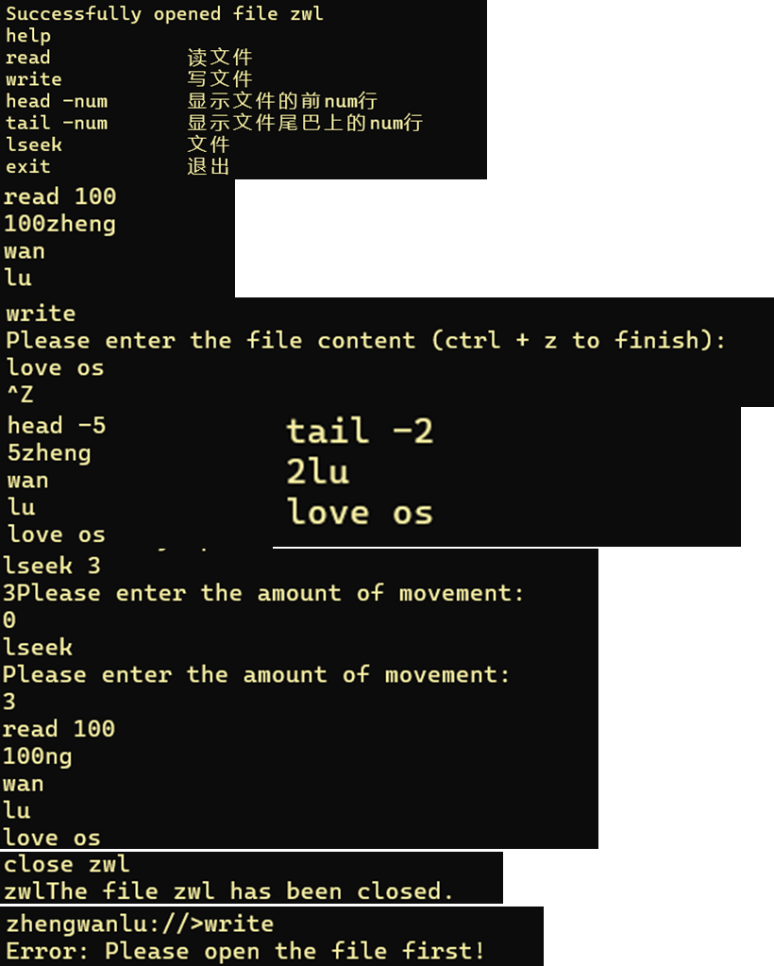


图5. 3 文件操作命令结果

使用clear命令清空屏幕操作，并查看版本号，之后delete命令删除操作过的文件结果如图5.4所示。

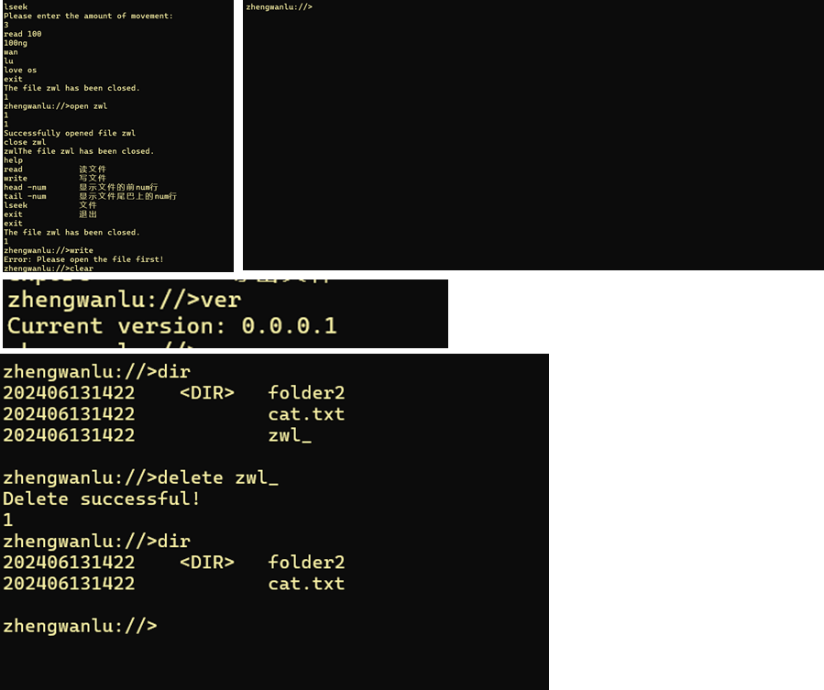


图5.4 文件操作命令结果

## 5.3 目录管理命令部分

如图5.5，展示了move与cd指令结合，单个cd指令，mkdir与dir指令结合，rmdir指令

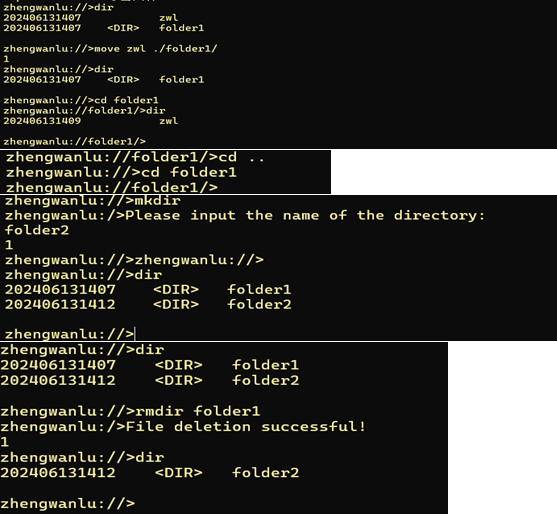


图5.5 目录管理命令结果截图

## 5.4 文件磁盘交互部分

首先导入文件路径及信息，为cat.txt。导入并打开读取文件，文件信息完整，之后创建文件并导出，显示导出成功，二进制文件，文件信息完整，同时使用copy和flock指令复制上锁。结果如图5.6所示。

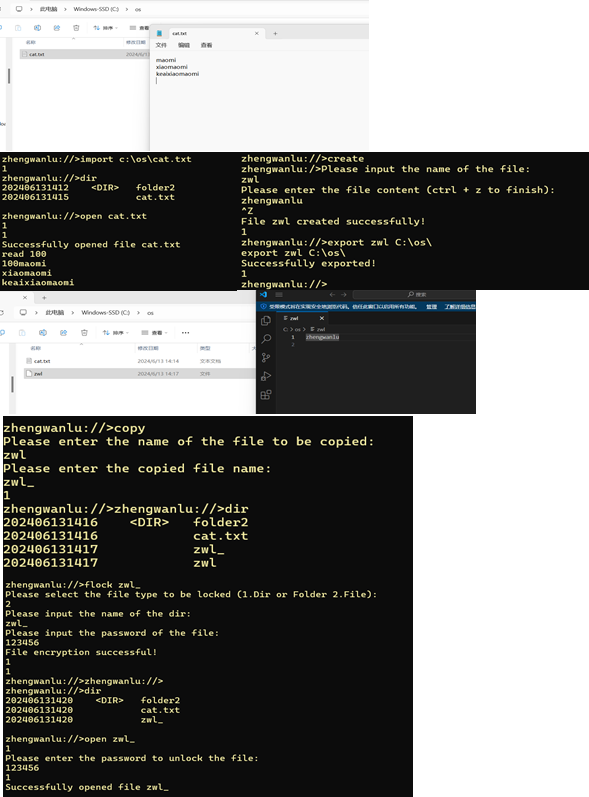


图5.6 文件磁盘交互结果

## 5.5 多线程功能

多线程功能主要体现在同步，竞争两方面，在两个用户线程测试下，同一个文件创建删除后，目录与文件内容都都一样，当一个用户线程操作文件时，另一个用户线程无法操作，互斥竞争，最后两个线程都可以完成退出。结果如图5.7所示。

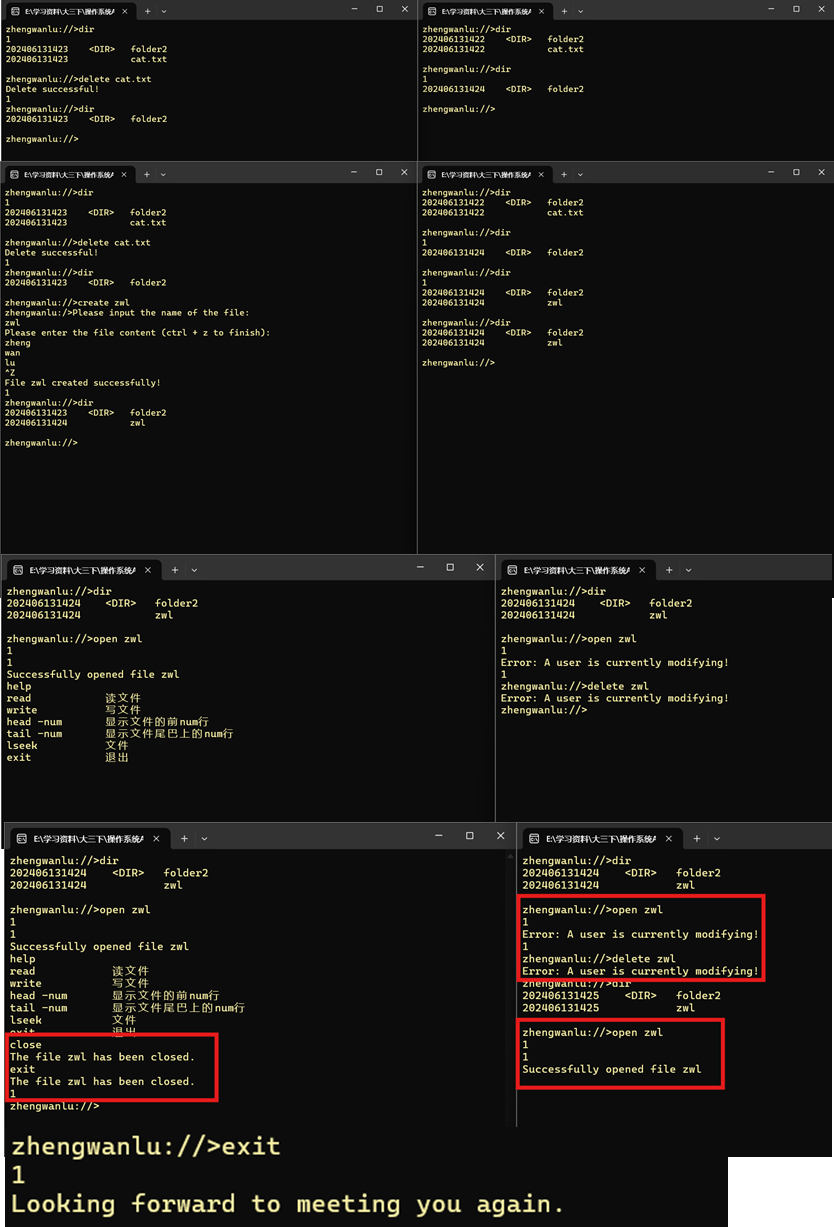


图5.7 多线程功能结果

# 6 课程设计分析及感受

## 6.1 课设分析

在本次课程设计中，我实现了一个基于 C++ 的简易文件系统，重点在于创建一个模拟环境，让用户可以在该环境中执行文件和目录的创建、删除、移动、复制等操作，同时支持用户管理和数据的导入导出功能。整个系统涵盖了用户会话管理、文件操作、目录管理以及与外部系统的数据交互，旨在提供一个基础但功能完整的文件管理平台。

## 6.2 遇到的问题及解决办法

（1）问题1：并发控制设计

解决办法：为了解决并发访问的问题，我主要采用了互斥锁（mutex）和条件变量。在关键操作如文件的创建、删除、读取和写入过程中，通过互斥锁保证同一时刻只有一个线程可以操作同一个文件或目录资源。

具体实现在MyOS类的kernel函数中，我使用了unique\_lock来加锁，并通过条件变量cv管理线程间的同步，确保在文件或目录被其他线程使用时，当前线程会等待直到资源可用。

（2）问题2：文件系统的完整性

解决办法：为了维护文件系统的完整性和逻辑正确性，我设计了两个关键的数据结构：FCB（文件控制块）和FCBTree（文件控制块树）。

FCB负责存储单个文件或目录的详综信息，如文件名、文件数据、锁定状态等。FCBTree则是一个树形结构，用于表示整个文件系统的目录层次，每个节点都是一个FCB对象。通过在FCBTree中添加、删除或搜索节点，系统可以高效地管理文件和目录。在insertNode和deleteNode方法中，操作前会检查父节点的存在，确保每次插入或删除操作不会破坏树的结构。

（3）问题3：文件数据一致性

解决办法：为了保证文件数据在读写操作中的一致性，我实施了文件锁定机制。使用isLocked标志和unlockCode来控制对文件的访问。

当一个线程（或用户）正在写入文件时，文件会被锁定，其他线程尝试访问该文件时将会被暂时阻塞或要求输入解锁码。此外，修改时间modifyTime在每次文件被修改时更新，帮助系统检测到潜在的冲突或不一致情况。这种机制确保了即使在多用户访问的情况下，文件的数据也能保持一致性和最新状态。

## 6.3 收获与反思

通过这次课程设计，我对文件系统的工作原理有了更深入的理解，尤其是在文件系统的结构设计和功能实现方面。在开发过程中，我面对了多种挑战，例如如何高效地管理文件元数据，以及如何实现文件和目录操作的原子性。

从个人知识收获的方面来讲，我获益匪浅。主要体现在深化了操作系统理论知识，通过实际操作和编码，使我能够将抽象的理论知识具体化，从而更好的理解了操作系统文件部分中的文件控制块(FCB)和目录树结构(FCBTree)的工作原理和应用。

同时我也有C++编程技能的提升：在实现文件系统的过程中，我大量使用了C++的多种高级特性，STL容器、智能指针、多线程和同步机制等。这不仅加深了我对C++语言的掌握，也增强了我在面对复杂问题时的编程能力。

## 6.4 尚存在的不足

（1）不足1：代码的健壮性

改进思路：可以加强错误处理机制，例如通过引入异常处理框架来捕捉和处理运行时发生的各种异常。此外，可以增加更多的用户输入验证和错误反馈机制，确保在输入数据不符合预期时能够给予用户明确的指导和错误提示，从而提升系统的稳定性和可靠性。

（2）不足2：用户体验性

改进思路：考虑开发一个图形用户界面（GUI），使用如Qt或Electron等框架，可以大幅提升用户的交互体验。图形界面可以直观地展示文件和目录结构，提供拖拽、右键菜单等现代化操作，使得用户操作更为直观和方便。此外，加入配置向导和帮助文档也能进一步降低用户的学习成本。

（3）不足3：功能扩展

改进思路：为系统添加更多高级功能，如：文件加密，权限管理，文件版本控制。通过这些功能的扩展，可以使文件系统更加完善。