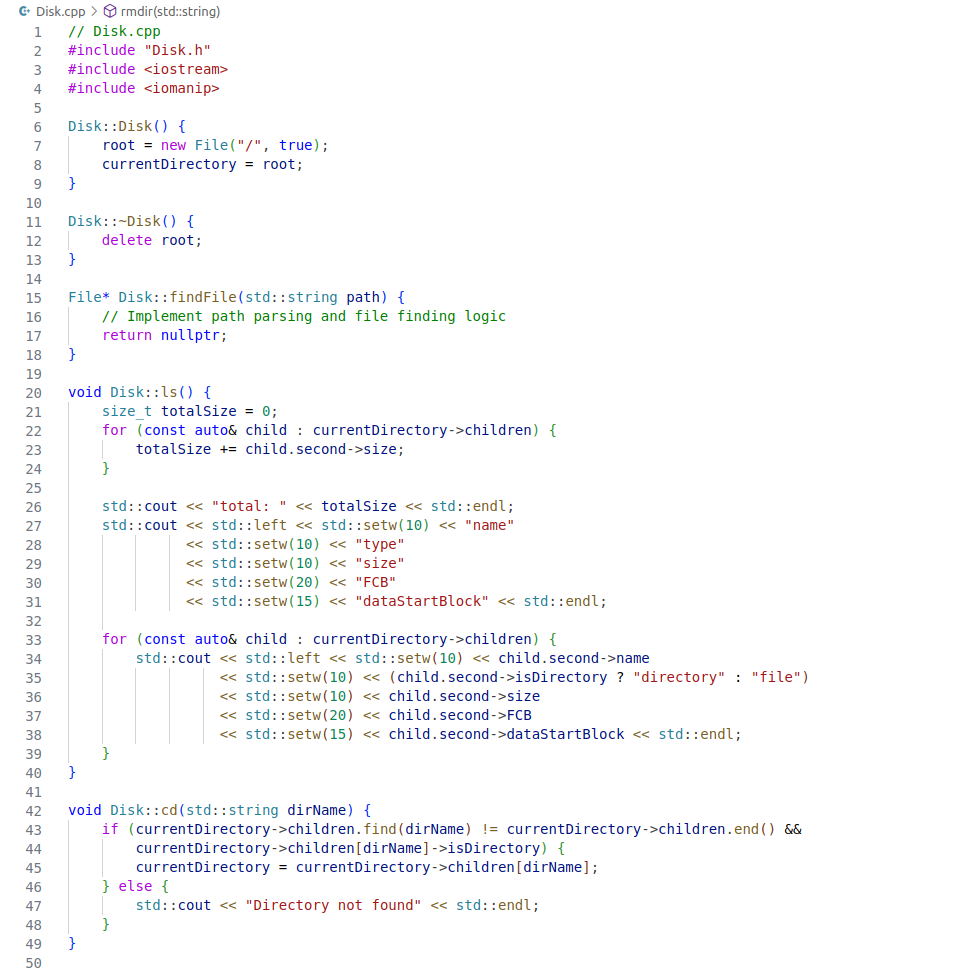
6.2代码

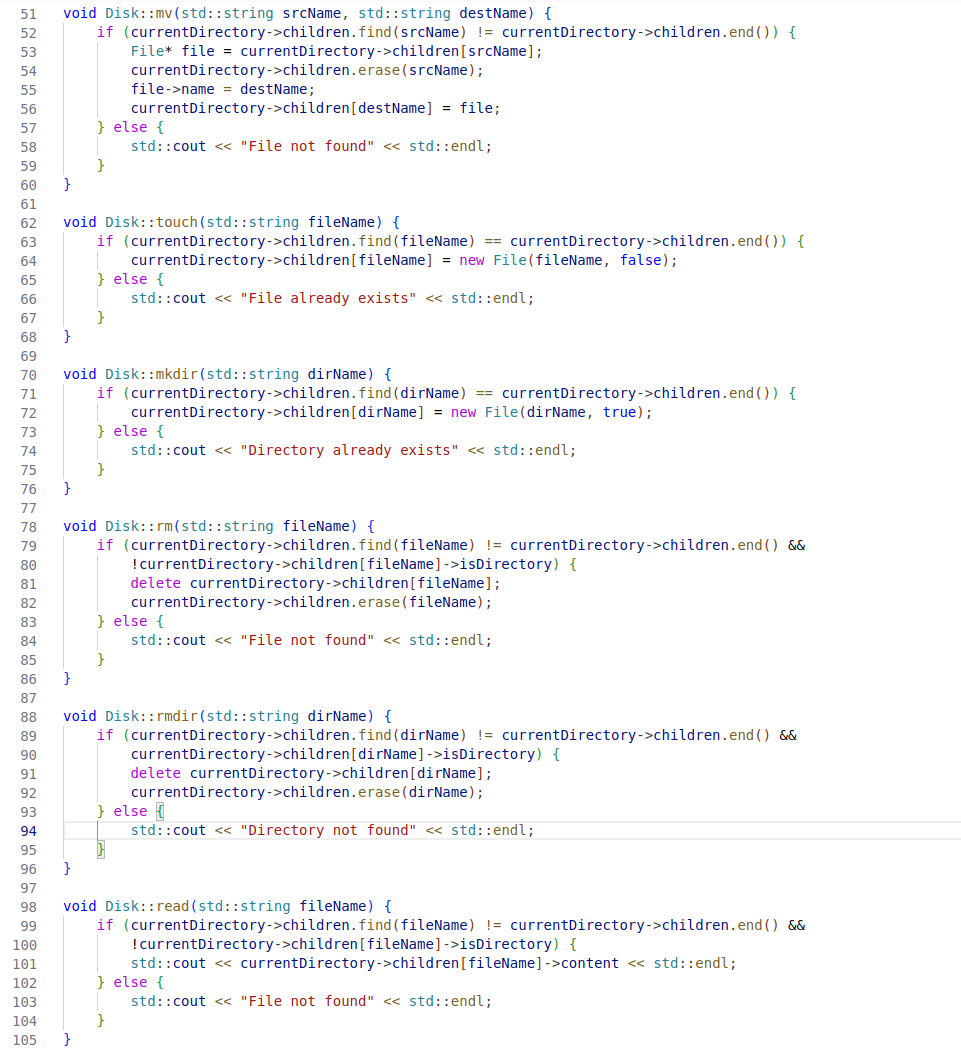
1. Disk.h



`Disk.h` 定义了一个名为 `Disk` 的类，旨在模拟计算机文件系统的核心功能。该类通过私有成员 `File\* root` 和 `File\* currentDirectory` 来追踪文件系统的根目录和用户当前所在目录。`findFile` 是一个私有方法，用于在给定路径下搜索文件或目录。公共接口包括一系列常用文件操作：`ls` 列出目录内容，`cd` 更改工作目录，`mv` 移动或重命名文件，`touch` 创建空文件，`mkdir` 创建目录，`rm` 删除文件，`rmdir` 删除空目录，`read` 和 `write` 分别用于读取和写入文件内容，以及 `exit` 函数，用于清理资源并结束程序运行。

1. Disk.cpp







1. 构造函数：初始化 Disk 对象时，创建根目录文件 "/"，并将当前目录设为根目录
2. 析构函数：析构函数负责删除根目录，从而清理整个文件系统。
3. findFile 方法:此方法应实现路径解析和文件查找逻辑，但实际代码中未完成，返回 nullptr。
4. ls 方法：列出当前目录下的所有子项，包括名称、类型、大小、FCB（文件控制块）和数据起始块信息，并显示总大小。
5. cd 方法：更改当前目录至给定的目录名，如果该目录存在且是目录类型。
6. mv 方法：移动或重命名文件，从当前目录中找到源文件后，更新其名称和位置。
7. touch 方法：在当前目录中创建一个新的空文件，如果文件名不存在。
8. mkdir 方法：在当前目录中创建一个新的目录，如果目录名不存在。
9. rm 方法：删除当前目录下的非目录类型的文件，如果文件存在。
10. rmdir 方法：删除当前目录下的目录，如果目录存在。
11. read 方法：读取当前目录下文件的内容并打印，如果文件存在且不是目录。
12. write 方法：写入内容到当前目录下的文件，更新文件大小和数据起始块，如果文件存在且不是目录。
13. exit 方法：输出退出消息，用于指示程序将结束文件系统操作。
14. File.h



1. std::string name：文件或目录的名称。
2. bool isDirectory：标志位，如果为 true，表示该 File 对象代表一个目录；否则，它是一个普通文件。
3. size\_t size：对于文件来说，表示其大小（以字节为单位）。对于目录，可以表示该目录下所有文件和子目录的总数或占用空间。
4. std::string content：仅当 isDirectory 为 false 时有效，存储文件的实际内容。
5. std::map<std::string, File\*> children：目录特有的属性，用于存储子文件和子目录。键是子文件或目录的名称，值是指向子 File 对象的指针。
6. void\* FCB：文件控制块（File Control Block）指针，这里将其设计为 void\* 类型，通常用于操作系统内部管理文件，但在本上下文中，它实际上指向当前的 File 对象自身，可能是为了方便访问或管理。
7. size\_t dataStartBlock：数据起始块，表示文件内容在内存中的起始位置，对于基于内存的文件系统特别有用，可以是文件内容字符串的起始地址。
8. File.cpp



构造函数：

1. 初始化 File 对象的基本属性：name, isDirectory, size, FCB, dataStartBlock。
2. 如果创建的是一个文件（而非目录），则设定 dataStartBlock 为文件内容 content 的内存地址。这通过 reinterpret\_cast 将 &content 的指针转换为 size\_t 类型的值来实现，这在基于内存的文件系统中，可视为文件内容的起始地址。
3. FCB 成员变量被设置为指向当前 File 对象的指针。
4. dataStartBlock 的设定仅在创建文件时生效，对于目录，该值保持默认的 0，因为目录没有具体的内容数据。

析构函数：

1. 在 File 对象销毁前，遍历其所有的子文件/目录，调用 delete 删除它们。
2. 这是递归清理文件系统的关键步骤，确保了整个目录树的正确销毁，防止内存泄漏。
3. 析构函数仅删除子文件和目录，但不涉及对 content 字符串的删除，这是因为 content 通常作为 File 对象的一部分，会在 File 对象本身被销毁时自动清除。
4. 如果子文件或目录中还包含其他动态分配的资源，也需要在各自的析构函数中进行适当的清理
5. my\_shell.cpp



my\_shell.cpp 实现了一个简单的命令行界面，用于与前面定义的 Disk 类交互，执行文件系统的基本操作。以下是代码的详细解析：

主要函数

executeCommand(Disk& disk, std::string commandLine)

1. 功能：解析用户输入的命令行，并调用相应的 Disk 类方法执行命令。
2. 参数：
3. disk：一个 Disk 类的对象引用，用于执行文件系统操作。
4. commandLine：用户输入的一行命令字符串。
5. 流程：
6. 使用 std::istringstream 来解析命令行，提取出命令名称和参数。
7. 根据命令名称，调用 disk 对象的相应方法。例如，如果命令是 "ls"，则调用 disk.ls()。
8. 如果命令未知，则输出 "Unknown command"。

int main()

1. 功能：程序的入口点，提供一个循环的命令行界面，直到用户输入 "exit" 命令。
2. 流程：
3. 创建一个 Disk 对象 disk。
4. 进入无限循环，等待用户输入命令。
5. 输出提示符 / ，等待用户输入一行命令。
6. 如果用户输入 "exit"，则调用 disk.exit() 并退出循环。
7. 否则，将用户输入的命令行传递给 executeCommand 函数进行处理。

6.3结果展示

