|  |  |  |
| --- | --- | --- |
| **验收成绩** | **报告成绩** | **总分** |
|  |  |  |

**武汉大学计算机学院**

**本科生实验报告**

**操作系统模拟实验**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：操作系统设计

指 导 教 师 ：

学 生 学 号 ：

学 生 姓 名 ：

二○二四年七月

**郑 重 声 明**

本人呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

本人签名： 日期： 2024年7月10日

**目录**

**1**、**进程调度实验**

1.1、 实验内容及上机实验所用平台………………………………………6

1.2、 数据结构及代码段分析 ………………………………………………6

1.3、 问题和调试过程…………………………………………………………7

1.4、 实验总结…………………………………………………………………8

**2**、**虚拟页式存储管理系统实验**

2.1、 实验内容及上机实验所用平台………………………………………11

2.2、 数据结构及代码段分析 ………………………………………………11

2.3、 问题和调试过程…………………………………………………………13

2.4、 实验总结…………………………………………………………………13

**3**、**文件系统实验**

3.1、 实验内容及上机实验所用平台………………………………………16

3.2、 数据结构及代码段分析 ………………………………………………16

3.3、 问题和调试过程…………………………………………………………22

3.4、 实验总结…………………………………………………………………22

1. 进程调度实验

摘要

在本实验中，我们将深入探讨单处理器环境下的进程调度原理，通过编程实现时间片轮转调度、短进程优先调度和多级反馈队列调度算法。实验旨在模拟每个进程的到达时间、运行时间和优先级，仿真进程状态的转换过程，包括就绪、运行和阻塞状态。我们将通过仿真计算平均周转时间、平均等待时间、平均响应时间等性能指标，并比较不同调度算法的效率和公平性。实验还将探讨多级反馈队列调度的细节，设计至少三个优先级队列。测试数据集的选择将确保覆盖多种运行情况，结果分析将突出不同算法的优缺点。

**关键词：**进程调度，时间片轮转，短作业优先，多级反馈队列

* 1. 实验内容以及上机所用平台

**实验内容：**本实验通过编程实现单处理器环境下的三种经典进程调度算法：时间片轮转调度、短进程优先调度和多级反馈队列调度。每个进程具有到达时间、运行时间和优先级等属性。在仿真过程中，进程会经历就绪、运行和阻塞等状态转换。实验重点是计算并分析平均周转时间、平均等待时间、平均响应时间等性能指标，评估不同调度算法的效率和公平性。多级反馈队列调度算法将设计至少三个队列，时间片大小不同。通过选择合理的测试数据集，我们将比较各种调度算法的优缺点，并详细记录实验过程及结果分析。

**实验平台：**visual studio2022

* 1. 数据结构及代码段分析

**重要数据结构**

（1）进程控制块PCB

为了描述进程的相关信息，需要设置PCB结构体。

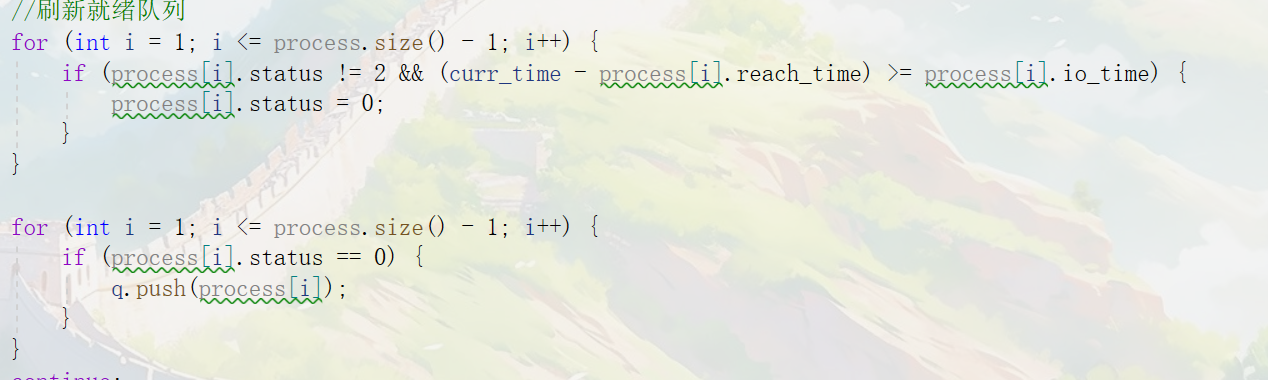
****

通过该结构体，实验可以模拟进程从到达系统到完成执行的整个生命周期，包括进程状态的转换和时间的记录。这些数据在不同调度算法中起关键作用，使得实验能够计算和比较各种性能指标，从而分析调度算法的效率和公平

**重要代码段分析**

1. 按时间刷新就绪队列

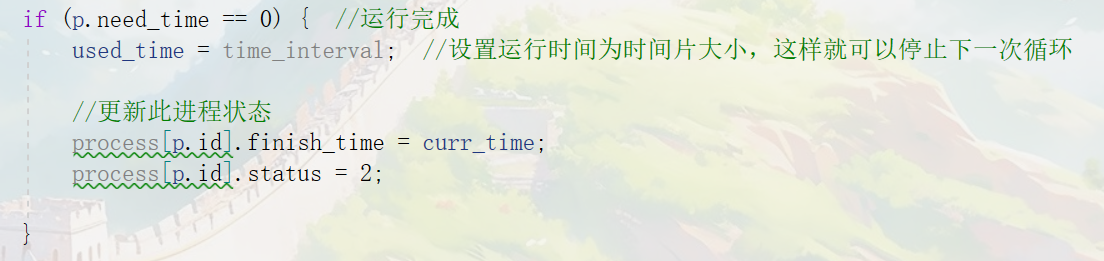
为了描述每个时刻进程状态的变化便于我们决定CPU调度的选择，需要按照时间单位进行刷新，然后修改每个进程的状态便于后续操作。

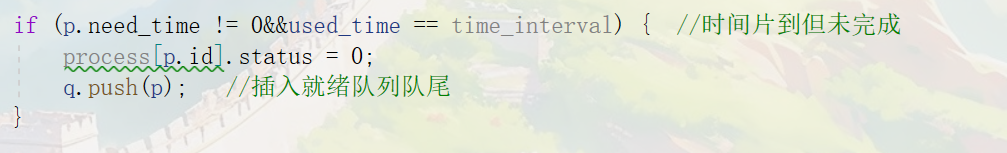


这段代通过遍历进程列表，更进程的状态为就绪状态（status = 0），并将所有就绪状态的进程加入就绪队列。这样可以确保在每个时间单位内动态调整进程状态，使调度器能够选择合适的进程进行调度，从而实现对系统中所有进程的有效管理和调度。

1. 时间片轮转算法一次循环结束时间的判断

循环结束条件应该是时间片用完或者进程完成即可结束本次循环。

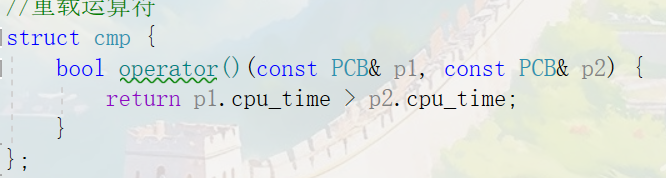




1. 最短作业优先算法中的排序

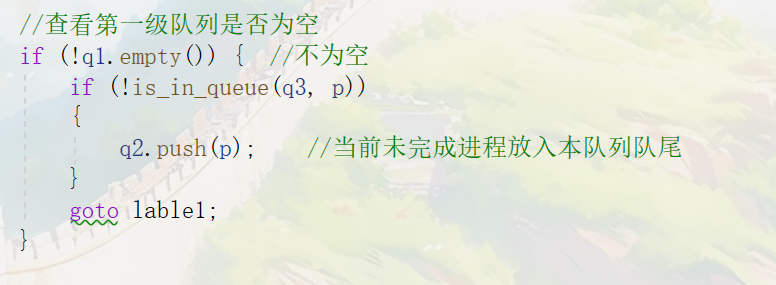
由于每次从就绪队列中取运行时间最短的进程，因此采用优先队列，按照运行时间升序排序。



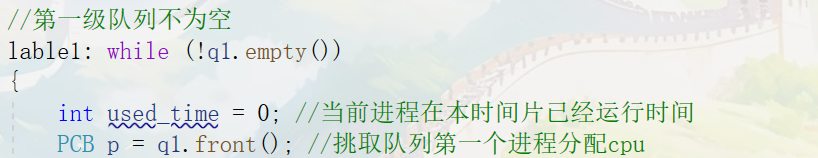


1. 多级反馈队列的抢占算法

多级反馈队列相对其它调度算法来说更关注每个进程的响应时间，因此此算法中的对抢占过程模拟十分重要。

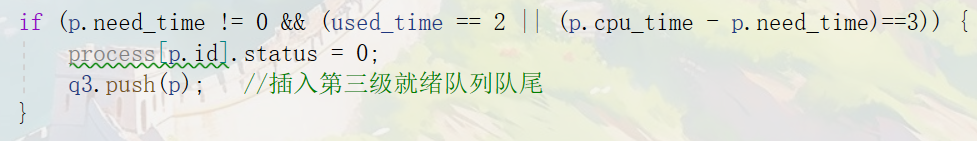


在第二级队列执行一个时间单位之后进行一级队列的刷新，并判断是否有新进程进入一级队列，有新进程时抢占处理机，回到一级队列执行中。



1.3 问题和调试过程

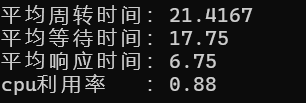
在多级反馈队列算法中，第一级队列之外的队列如果在被抢占时，应该放回本级队列队尾，但是再次运行时只能运行剩余的时间。也就是说要注意在本级队列中一共运行的时间不能超过本级队列的时间片。因此要加一个判断条件才能改正原来的问题。



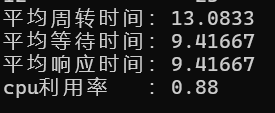
* 1. 实验总结

在本实验中，我成功实现并比较了三种经典的进程调度算法：时间片轮转调度、短进程优先调度和多级反馈队列调度。通过模拟每个进程的到达时间、运行时间和优先级，我们动态更新进程状态，并评估了不同调度算法的效率和公平性。仿真结果显示，各算法在不同情况下表现各异。时间片轮转调度算法提供了较好的响应时间，适合于交互性强的系统；短进程优先调度算法在处理短作业时效率高，但可能导致长进程饥饿；多级反馈队列调度算法兼顾了响应时间和吞吐量，但实现复杂。最终，我们通过计算平均周转时间、平均等待时间和系统利用率等性能指标，详细分析了各调度算法的优缺点，并总结出适用于不同场景的最佳调度策略。

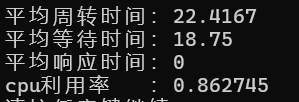
RR算法：



SJF算法：



多级反馈队列：



1. **虚拟页式存储管理系统实验**

摘要

本实验旨在通过设计和实现虚拟页式存储管理系统，理解虚拟内存和物理内存的关系及页面置换机制。通过模拟先进先出（FIFO）、最近最少使用（LRU）、时钟置换（CLOCK）和随机置换算法，观察和比较不同算法在处理缺页中断时的性能表现。实验中，我们将设计进程类和内存类，模拟逻辑地址向物理地址的转换过程，并处理缺页中断。实验结果将展示各步骤的内存分配和页面置换情况，并通过缺页中断次数和平均访问时间等指标分析各算法的优缺点，从而更好地理解内存访问的局部性原理和虚拟内存管理。

**关键词：**虚拟页式管理，时钟置换，LRU算法

2.1、 实验内容及上机实验所用平台

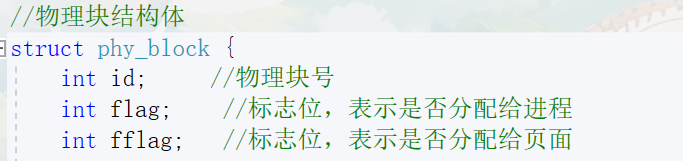
本实验通过设计并实现一个虚拟页式存储管理系统，模拟先进先出（FIFO）、最近最少使用（LRU）、时钟置换（CLOCK）和随机置换算法，观察和比较它们在处理缺页中断时的性能表现。系统将创建进程的页表，分配内存块，进行逻辑地址到物理地址的转换，并处理缺页中断。通过缺页中断次数和平均访问时间等指标，评估不同页面置换算法的优缺点和适用场景，理解内存访问的局部性原理和虚拟内存管理。

所用平台：visual studio2022

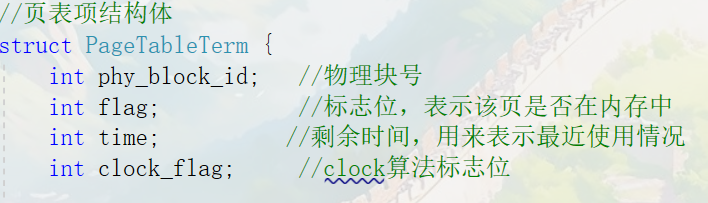
2.2、数据结构及代码段分析

**重要数据结构**

（1）物理块phy\_block，描述物理块相关信息



（2）页表项



用于描述页表中的一项。phy\_block\_id 记录物理块号，帮助实现虚拟地址向物理地址的转换；flag 表示该页是否在内存中；time 用于记录最近使用情况，支持最近最少使用（LRU）算法等；clock\_flag 则用于时钟（CLOCK）算法。该结构体在页面管理和置换过程中起关键作用。

（3）进程结构体



这个结构体用于描述一个进程的页式存储管理信息，包括进程页面个数（page\_count），页表（page\_table），分配的物理块数组（physical\_block），以及页号分配队列（page）。这些信息帮助管理进程的虚拟页到物理块的映射、内存分配和页面置换操作。

**重要代码段分析**

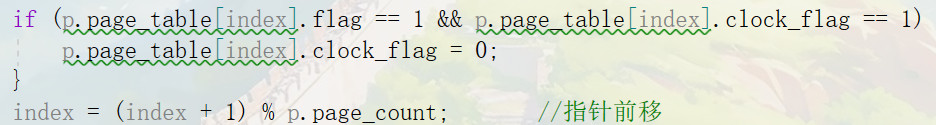
1. LRU策略中找到最近最久未使用的页面

LRU替换最重要的是找到符合要求的页面进行置换，本算法为每一个页面设置了一个time属性，每过一定的时间将其减少，再次访问时置为初值。每次置换时找time值最小的页面进行置换。



1. clock索引变化和标志位复位

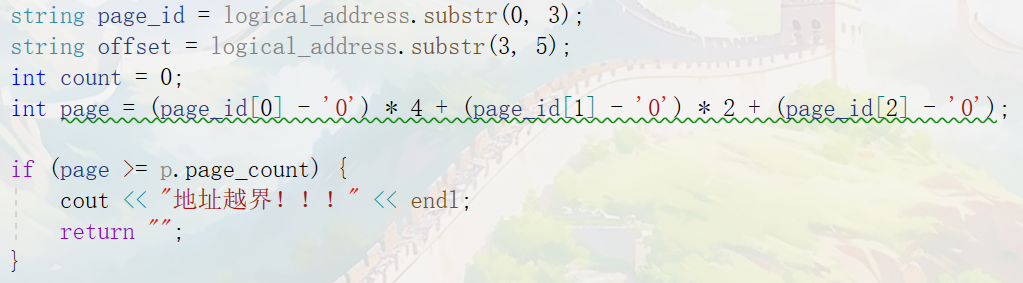
clock算法核心在于索引的变化和对标志位判断以及修改。



这段代码用于时钟置换算法。它检查页表项 p.page\_table[index] 的标志位 flag 和时钟标志位 clock\_flag，如果该页在内存中（flag == 1）且时钟标志位为1（表示最近使用过），则将 clock\_flag 置为0，表示该页不再是最近使用的页面。随后，通过 index = (index + 1) % p.page\_count 将指针前移到下一个页面，以循环方式继续检查其他页面。这一过程有助于找到可替换的页面。

1. 地址转换

由于输入的是地址，因此地址转换函数在本系统中十分重要。本系统设置八位地址长度，前三位为页号，后五位是偏移量。



2.3、 问题和调试过程

本实验最初的问题是如何将地址的输入转换成页号，并与页面置换结合起来最终得到物理地址。考虑到地址位数的不变，最后取一定的位数转换成页号，经过页面置换策略得到物理页号。并且在其中需要时刻注意错误处理，比如地址越界的问题，这其中需要不断的验证和打印输出观察结果，最终得到正确的地址转换函数，并与页面置换较好的结合起来。

2.4、 实验总结

在本实验中，我实现了虚拟页式存储管理系统，探讨了虚拟内存和物理内存的关系及页面置换算法的工作原理。通过模拟FIFO、LRU、CLOCK和随机置换算法，我们比较了它们的性能，发现不同算法在不同应用场景下有各自的优缺点。实验过程中，我们解决了地址转换和缺页中断处理等实现问题，最终通过缺页中断次数评估了各算法的性能。

结果类似下图：



1. **文件系统实验**

摘要

在本次文件系统实验中，我成功设计并实现了一个支持树状目录结构的模拟文件系统。通过自定义的数据结构和算法，实现了文件的创建、读写、删除以及目录的管理等基本操作，完全独立于操作系统的文件系统调用。我深入理解了文件系统的基本构造和工作原理，包括文件的存储方式、权限管理、空间管理和目录组织。通过模拟终端命令交互界面，用户可以方便地操作文件和目录，体验到了文件系统的核心功能和操作流程。

**关键词：**文件系统，磁盘块管理，权限管理

3.1、 实验内容及上机实验所用平台

实验包括设计文件和目录的数据结构，实现文件和目录的创建、读写、删除等基本操作，并通过终端接口与用户交互，模拟文件系统的操作过程。需要考虑文件的存储方式（如连续存储、链接存储、索引存储等）、空间管理、权限控制、文件和目录的组织结构等问题。实验中不使用与文件操作相关的系统调用，而是基于自定义的数据结构来管理文件和目录。

所用平台：visual studio2022

3.2、数据结构及代码段分析

**重要数据结构**

1. 磁盘块定义



这段结构体定义描述了模拟磁盘块的基本属性和功能：每个磁盘块具有唯一的标识号id，大小size，以及用于指示同一文件中下一个磁盘块的next\_id（如果存在）。flag标志位表示块是否已分配，capacity指示已用容量，file\_id标识所属文件，而data则存储实际数据内容。

1. 文件结构体

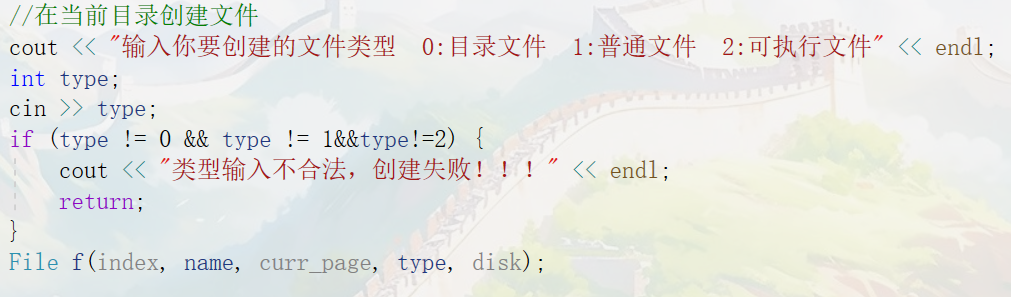


这段结构体定义描述了文件的基本属性和功能：每个文件有唯一的标识号id和名称name，大小size表示文件的大小，in\_which\_file指示所在的目录号（根目录为0，其他目录为正整数），type表示文件类型（目录文件、普通文件或可执行文件），status表示文件的状态（关闭或打开的计数），权限方面包括读、写、执行权限分别由read\_power、write\_power和exe\_power表示。block向量存储了文件占用的磁盘块编号，而file\_list向量则记录了该目录下所有文件的id。

**重要代码段分析**：

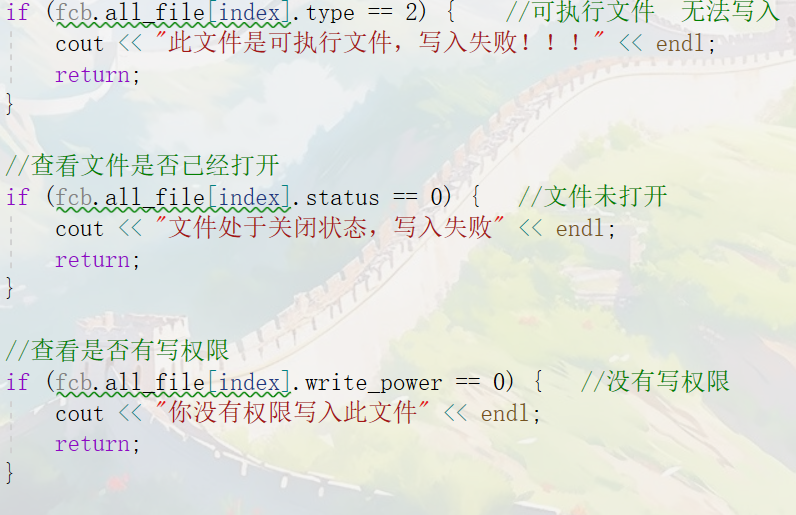
1. 创建文件

函数 create\_file 用于在虚拟文件系统中创建一个新文件。它首先检查磁盘空间是否已满，若已满则无法创建文件。然后，用户输入文件名并检查是否重名。接着，函数验证用户是否具有在当前目录下的写权限。如果目录文件大小达到1024字节（一个块大小），则需要为目录分配新的磁盘块。用户输入文件类型（目录文件、普通文件、可执行文件），创建文件后更新目录信息。最后，函数将新文件添加到文件分配表中，并增加文件索引。



（2）写入文件

函数 write\_file 实现了在虚拟文件系统中写入数据的功能。首先，它提示用户选择要写入的文件并检查该文件是否存在于当前目录中。如果文件存在且不是目录文件或可执行文件，且文件已打开且用户具有写权限，则允许写入。



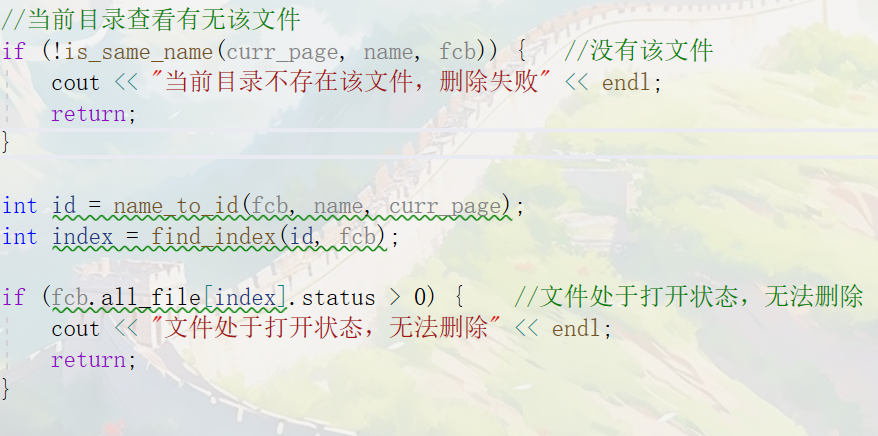
用户输入要写入的内容后，函数计算写入内容的大小并检查磁盘空间是否足够。如果最后一个磁盘块有足够的空间，则直接写入，否则需要分配新的磁盘块。当现有的最后一个块不能完全容纳新数据时，函数将剩余数据分配到一个或多个新的磁盘块中。它根据需要的块数分配足够的空闲块，并将数据分段写入这些块中，同时更新文件分配表和磁盘块的信息，包括文件大小、块使用情况和块链表指针。





1. 删除文件或目录

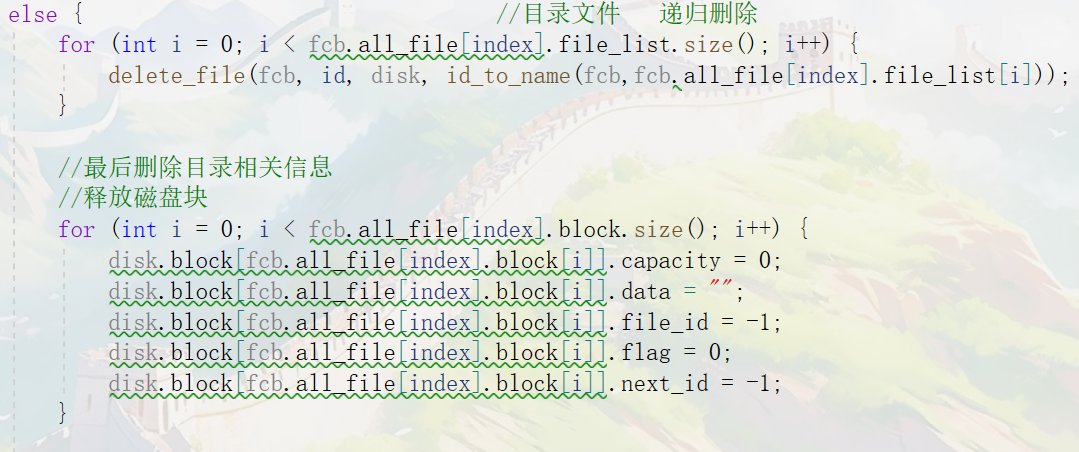
函数 delete\_file 实现了在虚拟文件系统中删除文件或目录的功能。首先，函数检查当前目录中是否存在指定名称的文件或目录。如果不存在，则输出错误信息并返回。如果文件或目录存在但处于打开状态，函数也会输出错误信息并返回。



对于普通文件，函数首先释放该文件占用的磁盘块，将块的容量、 数据、文件ID和标志位重置为初始状态。然后，它从文件控制块（FCB）中删除文件的信息，并更新目录中的文件列表和大小。



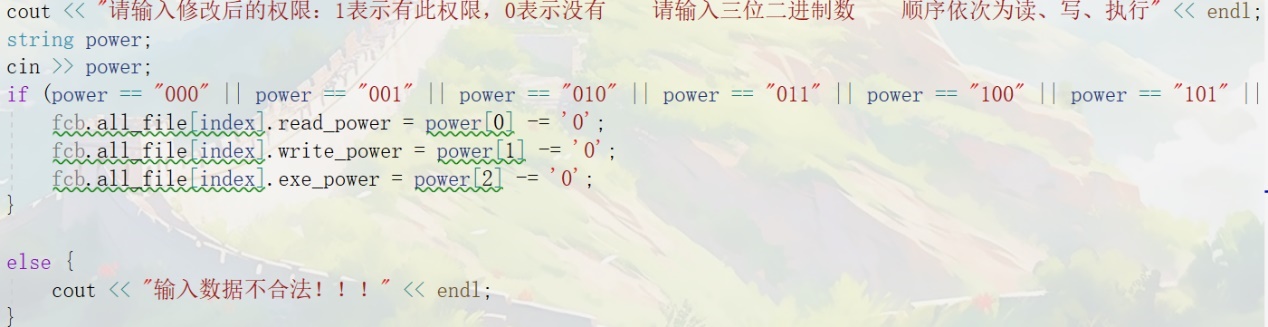
对于目录文件，函数采用递归删除的方式，首先递归删除目录下的所有文件和子目录，最后删除该目录自身。类似于删除普通文件，函数会释放目录占用的磁盘块，并更新父目录中的文件列表和大小。



1. 修改权限

函数 modify\_power 用于修改文件系统中某个文件的权限。 首先，函数提示用户输入要修改权限的文件名，并检查当前目录中是否存在该文件。如果文件不存在，则输出错误信息并返回。接着，通过文件名和当前目录获取该文件的ID和在文件控制块（FCB）中的索引位置。

然后，函数提示用户输入新的权限值，要求输入三位二进制数，依次表示读、写和执行权限。函数验证输入的权限值是否合法，即是否为有效的三位二进制数（如“000”、“001”等）。如果输入合法，函数将二进制字符串转换为整数，并分别更新文件的读、写和执行权限。如果输入不合法，函数输出错误信息并返回。



3.3、问题和调试过程

首先是磁盘块的分配与回收问题，为了高效管理磁盘空间，需要使用链表或位图来跟踪空闲和已分配的磁盘块。链表可以通过链接空闲块快速找到可用块，而位图则通过位的状态表示块的使用情况。实现一个块分配函数，在每次请求分配块时，从链表或位图中找到第一个空闲块并标记为已分配，同时实现一个块回收函数，在文件删除时，将释放的块重新标记为空闲并链接回链表或更新位图。此外，磁盘块的碎片化问题也需要解决。随着文件的创建、删除和修改，磁盘块可能会出现碎片化，影响读取性能。通过实现文件的分段存储机制，可以将文件数据重新排列，使其存储在连续的块中，提高读取效率。

其次是权限管理问题，权限管理是文件系统中确保数据安全和访问控制的关键部分。如何准确设置和检查文件的读、写、执行权限。在文件创建和权限修改过程中，需要提供用户友好的接口来输入和设置权限，并在文件操作（如读取、写入、执行）时进行权限检查。在实现权限检查时，系统需要在每次文件操作前检查用户是否具有相应的权限，并根据权限设置决定是否允许操作。

权限管理问题可以通过设计一个管理员界面来解决。管理员界面允许系统管理员对文件和目录的权限进行集中管理。管理员通过命令行界面修改各个文件和目录的读、写、执行权限。管理员可以选择目标文件或目录，并输入新的权限设置，如三位二进制数表示的读、写、执行权限。

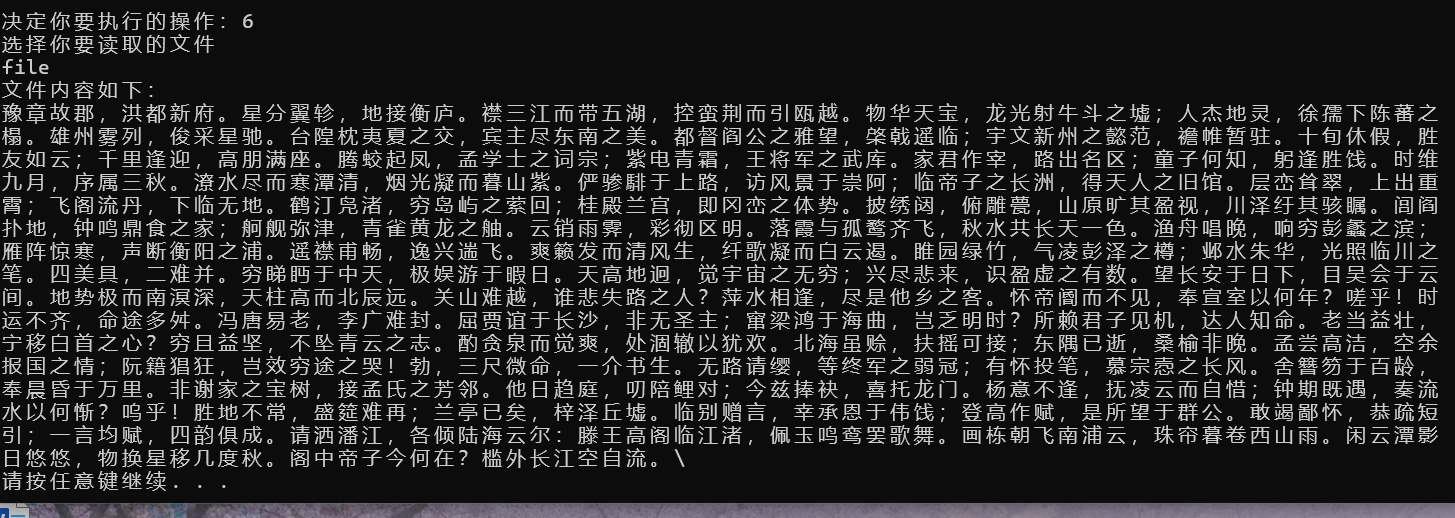
3.4、实验总结

在本次文件系统实验中，我成功实现了一个模拟文件系统，包括文件和目录的创建、读写、删除等基本操作，并通过自定义数据结构和算法来管理文件和磁盘块。实验中，我解决了多个关键问题，如磁盘块的高效分配与回收，以及文件权限的管理。为了管理磁盘块，我们设计了磁盘块结构体，包含块号、大小、下一块号、标志位、容量和数据字段，通过检查剩余磁盘块并动态分配新的磁盘块，确保了存储空间的有效利用。权限管理方面，通过设置读、写、执行权限，防止未授权操作，并通过管理员界面集中管理权限，保证了文件系统的安全性。我们设计了用户交互接口，使用户能通过终端命令进行文件系统操作，并注重输入的合法性验证和错误处理，提升了系统的可靠性。

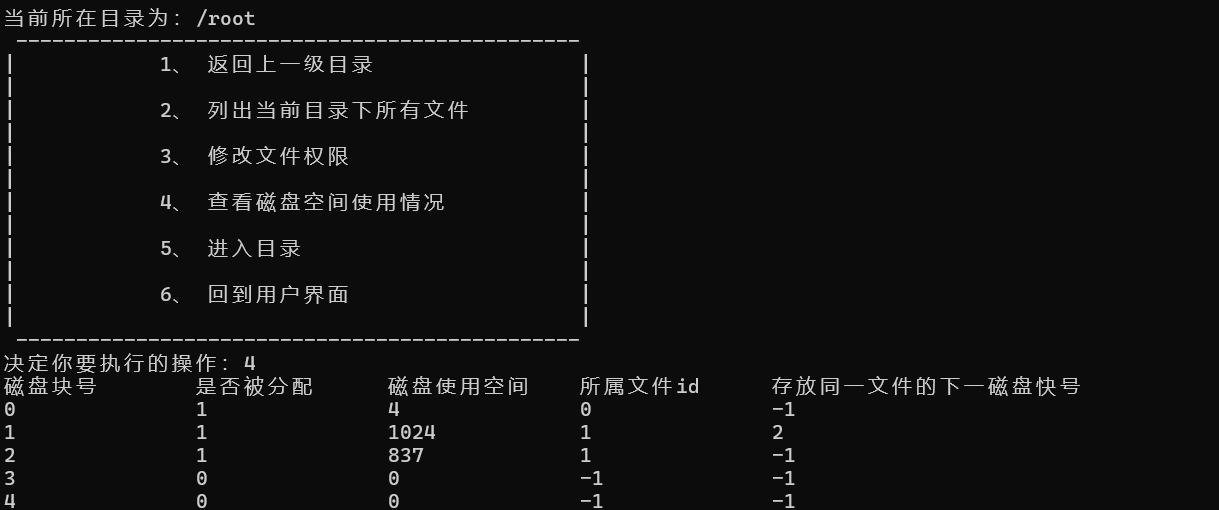
用户界面



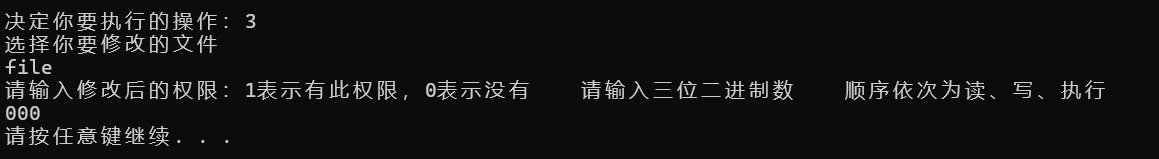
写入文件和读取文件



磁盘块管理



权限修改



文件执行模拟

