**项目说明文档**

**科目：操作系统**

**报告名称：请求调页存储管理方式模拟**

**学号：2253406**

**姓名：李跃跃**

**完成日期：2024年5月15日**

# 项目目的

1. 通过模拟请求调页存储管理方式，加深对内存块的理解；
2. 进一步加深对FIFO和LRU两种页面置换算法的理解，学会对调度算法的应用。

# 开发环境

## 操作系统平台

Windows 11+x64处理器

## 开发研究环境

编译器及版本：Visual Studio 2022

编程语言：C++

# 基本需求

## 基本任务

假设每个页面可存放10条指令，分配给一个作业的内存块为4。模拟一个作业的执行过程，该作业有320条指令，即它的地址空间为32页，目前所有页还没有调入内存。

## 功能描述

在模拟过程中：

1. 如果所访问指令在内存中，则显示其物理地址，并转到下一条指令；
2. 如果没有在内存中，则发生缺页，此时需要记录缺页次数，并将其所在页调入内存；
3. 如果4个内存块已满，则需要进行页面置换；
4. 所有320条指令执行完成后，计算并显示作业执行过程中发生的缺页率；
5. 本项目选用了FIFO和LRU两种置换算法，程序运行时用户可以根据需求进行选择，然后会执行相应的算法；
6. 本项目中作业指令的访问次序严格按照50%顺序执行，25%均匀分布在前地址部分，25%均匀分布在后地址部分的原则形成。

# 实现的基本思路

## 对数据结构进行初始化

### Page类

该类用于表示物理页，即内存块。

1. **成员变量**

该类中的成员变量包括：整型变量pageNumber，用于记录存储在该物理页中的逻辑页的编号，初始时物理页为空，该变量为-1；整型变量time，用于记录逻辑页调入物理页的时间长短，是FIFO页面置换算法执行的依据；整型变量dis，用于记录上一次使用的该页中的指令距离当前正在执行指令的距离，是LRU页面置换算法执行的依据。

1. **成员函数**

构造函数，在该函数中：将pageNumber初始化为-1，表示当前物理页中没有任何逻辑页调入；将time、dis初始化为0。

## 实现基本的功能函数

1. **input()**

用于输出输入提示，要求用户选择两种页面置换算法（1为FIFO算法，2为LRU算法）中的一个，并读取用户输入的选项，如果用户输入非法字符，即1和2以外的字符，则清空缓冲区，输出输入选项不合法的提示并要求用户重新输入，直到用户输入合法的字符为止。

1. **printMemory()**

用于打印当前内存块的使用情况。首先在该函数中对存储物理页的数组pages中的page元素进行遍历，如果page中的pageNumber变量不为-1，则当前物理页中有逻辑页装入，打印该逻辑页的编号；否则打印空格，表示当前该物理页为空。然后打印主函数传入的字符串message，如果当前要访问的指令所在的逻辑页在物理内存中，则message为对应指令的“物理页号+业内地址”，否则为“缺页！”信息，表示当前指令不在内存中，需要将该页调入内存。

1. **FIFO()**

用于在作业的4个内存块放满且访问指令不在内存中时对页面进行置换，具体实现思路见第6部分核心算法。

1. **LRU ()**

用于在作业的4个内存块放满且访问指令不在内存中时对页面进行置换，具体实现思路见第6部分核心算法。

1. **isInMemory()**

该函数接受一个整型参数page和一个物理页数组pages，用于检查当前访问指令所在的逻辑页page是否在物理内存中：遍历pages数组中的每一个元素并逐个检查元素的pageNumber变量，若该变量的值与传入参数page相等，则表示该逻辑页在物理页中，返回物理页的编号i；若遍历结束都没有找到pageNumber与page相等的元素，则代表该逻辑页不在物理页中，返回-1即可。

# 数据结构

数组，类型为自定义类Page，大小为分配给一个作业的内存块个数4，该数组用于模拟物理页，其中记录关于物理页使用情况的信息，包括存储的逻辑页页号、该逻辑页调入内存的时间以及该内存页中上一次被执行的指令与当前正在执行指令的距离。

# 核心算法

## FIFO

该算法执行的核心思想是：当物理页存满且当前访问指令不在物理页时，需要进行页面调度，首先要从物理页中选择一个逻辑页调出，然后再将所访问指令所在逻辑页调入内存，调出的依据是当前内存中逻辑页调入内存的顺序，即调出最先调入的逻辑页，也可以理解为调出存在在内存中时间最长的逻辑页。

在代码实现时，该函数接受一个存储物理页的数组pages，并对该数组中的元素page进行遍历，如果某一个page的pageNumber为-1，则该物理页当前还未被使用，直接返回该物理页的编号即可；否则逐个检查page中的time变量，最后返回time值最大的物理页的页号，该物理页中存储的逻辑页就是被选中要被调出内存的逻辑页。

## LRU

该算法执行的核心思想是：当物理页存满且当前访问指令不在物理页时，需要进行页面调度，首先要从物理页中选择一个逻辑页调出，然后再将所访问指令所在逻辑页调入内存，调出的依据是当前内存中每一个逻辑页中上一次被访问的指令与当前正在访问指令之间的距离，也可以理解为将最长时间不使用的逻辑页调出内存。该算法依据程序执行的局部性原理，即用过去代码的执行情况去预测将来的执行情况，认为越近执行的指令在未来被执行的概率更大。

在代码实现时，该函数接受一个存储物理页的数组pages，并对该数组中的元素page进行遍历，如果某一个page的pageNumber为-1，则该物理页当前还未被使用，直接返回该物理页的编号即可；否则逐个检查page中的dis变量，最后返回dis值最大的物理页的页号，该物理页中存储的逻辑页就是被选中要被调出内存的逻辑页。

## main

主函数首先调用input函数读取用户选择的调度算法，然后用一个函数指针algorithm指向执行该算法的函数（FIFO或LRU），然后情况屏幕准备对指令的访问情况以及内存的使用情况进行打印。为了使用随机数函数生成指令的逻辑地址，并保证每次生成的随机数都不同，用当前的系统时间作为生成伪随机数的种子。

接着定义一些变量，包括：整型变量count，记录执行指令的条数，取值为0到319，当count=320时，320条指令执行完成；整型变量pageFaults，用于记录指令访问过程中的缺页次数，即如果当前正在访问的指令所在的逻辑页不在物理内存块当中，则该变量+1；整型变量m，记录当前正在执行指令的逻辑地址，取值为0到319；Page型数组pages，大小为4，用于模拟内存块，其中记录一些关于物理内存块使用情况的信息。

接着开始执行指令，当count<319时执行如下操作：

1. 打印当前执行的指令条数：第count条指令，然后根据当前访问指令的逻辑地址m计算逻辑页号，即m/10（10为每一个内存块中存储的指令条数），并将该值存储在page变量中，然后将该变量和pages数组传入isInMemory函数，查看该逻辑页是否在物理内存中，用memoryAddress保存返回值；
2. 由于当前又向前执行了一条指令，因此物理内存块中当前存储的逻辑页中上一次被访问的指令与当前指令的距离又增大了，因此遍历pages数组，并对每一个元素page的dis变量执行+1的操作；
3. 根据isInMemory的返回值memoryAddress执行不同的操作：若返回值为-1，则该逻辑页不在物理内存中，需要执行页面调度算法，调用algorithm函数并获取返回值targetPage，该值即为即将使用的物理页号，然后将pages[targetPage]的pageNumber修改为page（当前指令所在逻辑页的页号），time修改为0，dis修改为0，最后将“缺页！”信息保存在message变量中；若返回值不为-1，则返回值为逻辑页所在的物理页号，计算该指令的页内地址，即m%10，修改该物理页的dis变量为0，然后将“物理地址：memoryAddress+m%10”的信息保存在message变量中；
4. 最后调用printMemory函数打印当前物理内存的使用情况以及内存访问信息message；
5. 然后根据count的值对m进行修改，即决定下一条指令的逻辑地址：若count%4=0，则下一条指令为m+1；若count%4=1，则下一条指令为0~m-1中的随机一条指令；若count%4=2，则下一条指令为m+1；count%4=3，则下一条指令为m+1~319中的随机一条指令。这样控制指令的执行顺序可以基本满足50%顺序执行，25%均匀分布在前地址部分，25%均匀分布在后地址部分的原则。

当320条指令全部执行完毕后打印缺页率：pageFaults/320，然后询问用户是否要继续：若用户输入y，则情况屏幕，从请求用户选择算法开始重新执行；若用户输入n，则直接结束程序；若用户输入其他字符，则输出输入字符非法的提示并要求用户重新输入，这样可以保证程序在运行时不会崩溃，增强程序的健壮性。

# 操作说明

## 选择页面置换算法

图片包含 文本

描述已自动生成

输入1或2选择页面置换算法，如下：

文本

中度可信度描述已自动生成

然后按下回车，会进入程序执行页面：

图片包含 文本

描述已自动生成

如果输入1、2以外的字符将会显示“输入选项不合法”的提示并请求重新输入，如下：

文本

描述已自动生成

## 程序执行页面

当前正在执行的指令编号

该指令执行时物理内存块的使用情况，空格表示空，编号代表内存块中存储的逻辑页的页号

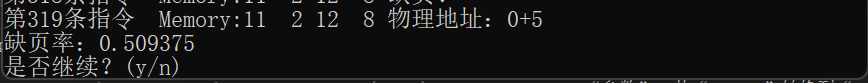
如果当前访问指令所在逻辑页在物理内存中则显示的是物理页号+页内偏移量

如果当前访问指令所在逻辑页在不在物理内存中则显示“缺页！”但此时该逻辑页已经在物理内存中存在

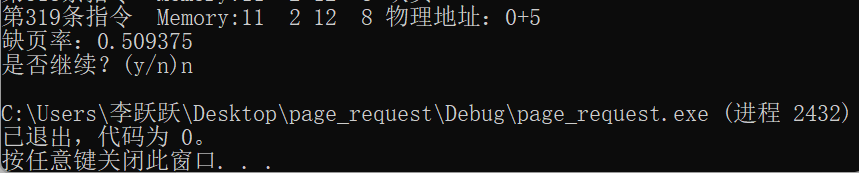
文本

描述已自动生成

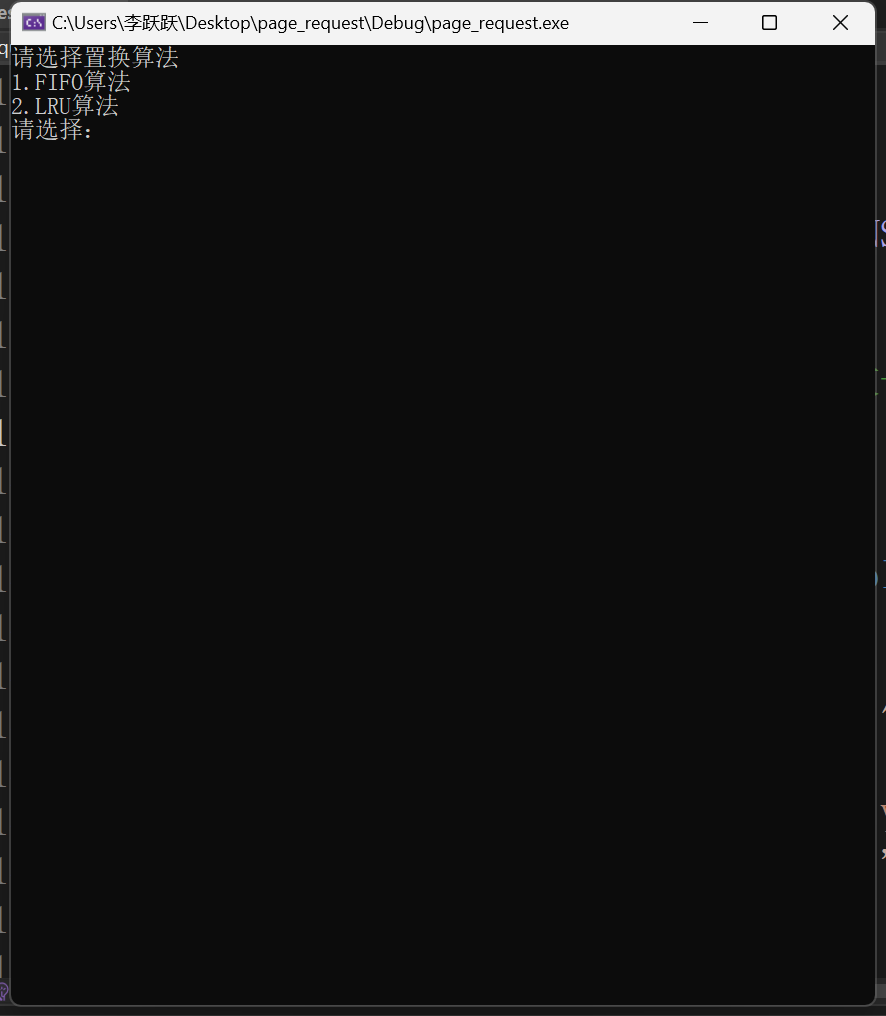
320条指令执行完毕后会计算并打印缺页率，如下：



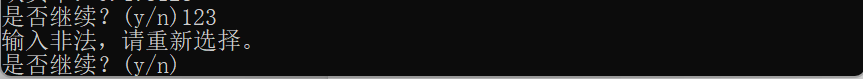
根据提示可选择输入y或n，若输入n，则程序运行结束：



若输入y，则继续跳转至算法选择界面，可以重新选择算法并再次进行模拟：



若输入y和n以外的字符则会输出“输入非法”的提示，并要求重新选择：



# 心得体会

## 总结

通过本次对请求调页存储管理方式的模拟，我对FIFO和LRU两种页面置换算法有了更加深刻的理解与认识；在修补程序漏洞的过程中，我对程序调试和debug的过程与方法也有了更深的体会。

## 反思

本程序中所有项目要求中给定的数字，如程序中指令的总条数、每一页中指令的条数、给每一个作业分配的内存块的个数都是以宏定义的方式定义在头文件下方，如果想要模拟其他规模的页面调度可以直接在源代码的此处进行修改。如果想要达到更好的用户交互效果可以增设三个整型变量，让用户自己输入这些参数。