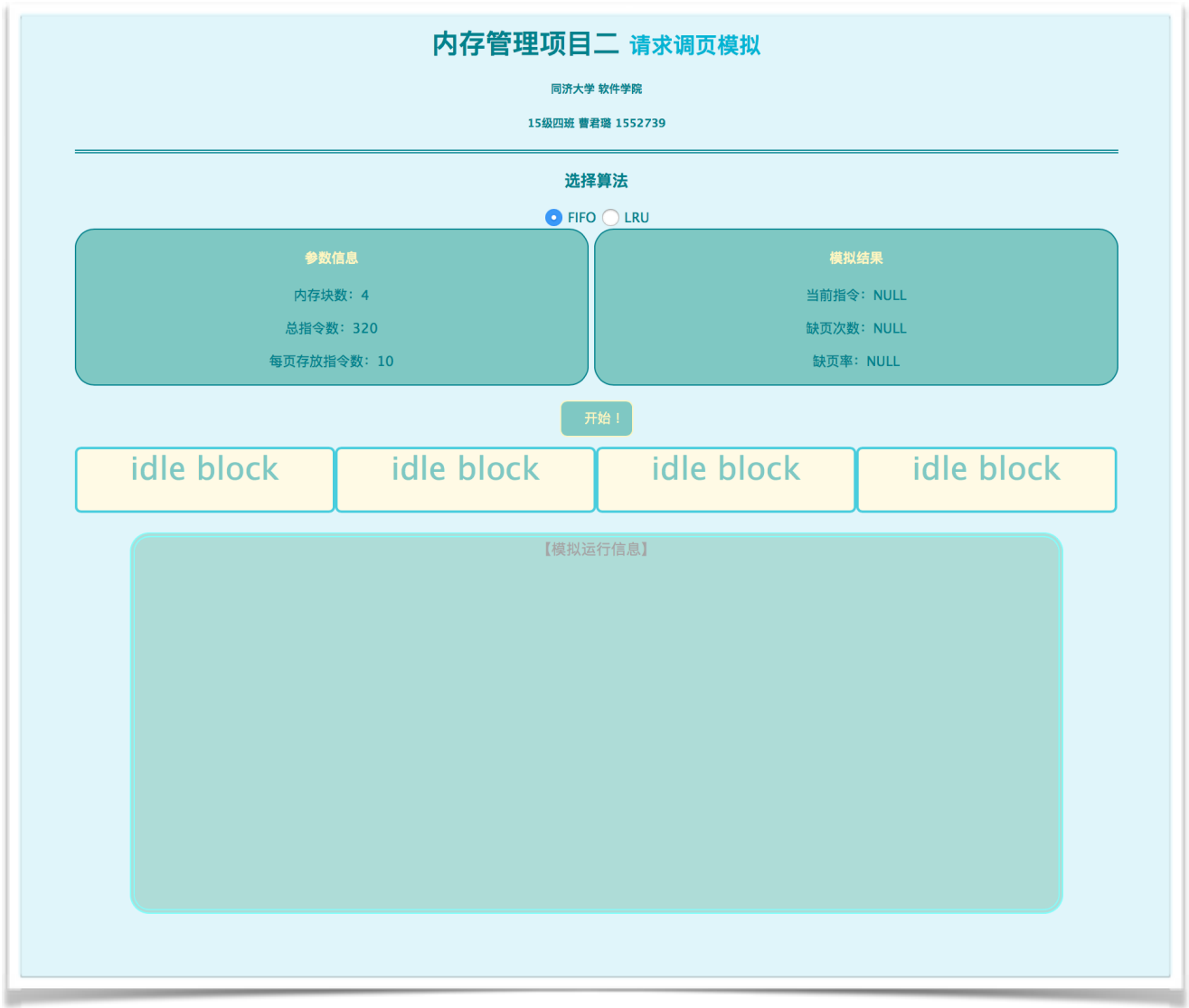
操作系统课程项目二

请求分页分配方式的模拟

1552739 软件4班 曹君璐



项目目的

- 页面、页表、地址转换
- 页面置换过程
- 加深对请求调页系统的原理和实现过程的理解。

开发环境

- 操作系统平台：MAC OS
- 开发语言：html、css、js

项目需求分析

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。

模拟过程

- 在模拟过程中，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；如果没有在内存中，则发生缺页，此时需要记录缺页次数，并将其调入内存。如果4个内存块中已装入作业，则需进行页面置换。
- 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。

置换算法

- 采用先进先出（FIFO）置换算法。
- 最近最久未使用（LRU）算法。

设计实现

主要界面

- 选择算法

- 参数信息

- 模拟结果

- 开始按钮

- 内存块

- 控制台信息

主要实现过程

- 首先由对应函数按照题目要求产生对应的320条随即指令。
- 当指令到来时，首先查询4个物理模块中是否含有当前指令，如果有则直接下一条指令，若没有就进行判定查找物理块中是否还有空闲盘块，有该指令则直接调入，无则要发生页面置换。
- 按照一定规则（FIFO/LFU）进行页面置换，知道最后一条指令完成后，显示缺页次数和缺页率。

具体实现

- 产生320条指令：

  指令的地址按如下规则生产：

  ① 50%的指令是顺序执行的；

② 25%的指令是均匀分布在前地址部分；

③ 25%的指令是均匀分布在后地址部分；

320条随机指令的产生规则如下：

① 在[0，319]的指令地址之间随机选取一起点m；

② 顺序执行一条指令，即执行地址为m+1的指令；

③ 向前跳转：在前地址[0，m+1]中随机选取一条指令并执行，该指令的地址为m′；

④ 顺序执行一条指令，其地址为m′+1的指令；

⑤ 向后跳转：在后地址[m′+2，319]中随机选取一条指令并执行；

⑥ 重复上述步骤①～⑤，直到执行320次指令。

- 初始化内存块—— initMemory()

```
function initMemory()
```

使用随机数产生页数、偏移量，将指令调入页面

- 查找物理块中是否有该页面—— isInstructionAvailable(number)

```
function isInstructionAvailable(number)
```

当指令页到达时与当前物理块中存放的页面memory[i]比较。

若相等，说明该页已经存在，没有发生缺页。

- 判断指令是否被运行过——isInstructionExecuted(number)

```
function isInstructionExecuted(number)
```

判断instructions[number]的值是否为“undefined”。

若是，将其修改为“false”。

返回instructions[number]

- 选择所使用的算法——chooseAlgorithm()

```
function chooseAlgorithm()
```

判断querySelector的值，根据其值调用FIFO或LRU算法

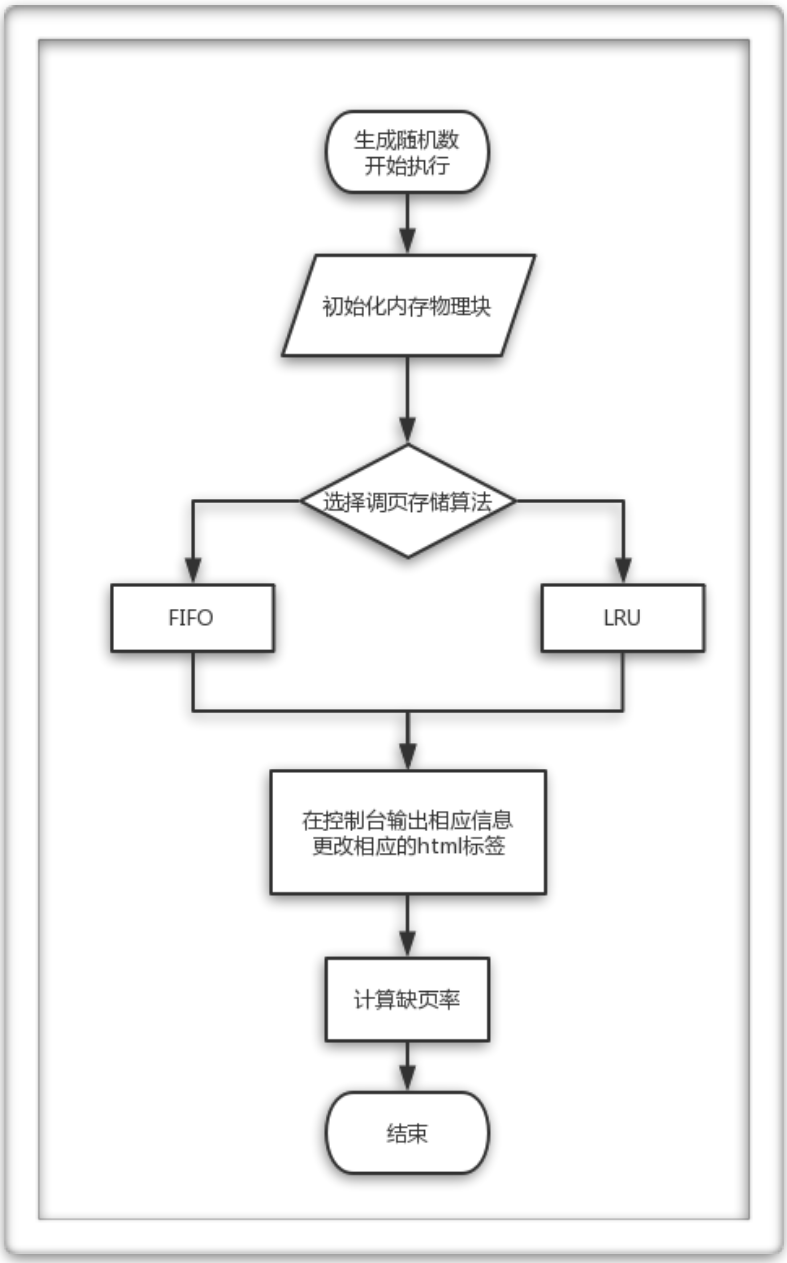
- 查找应予置换的页面

当新的指令到达且物理块已无空闲时，根据FIFO或LRU算法计算出指令所在页面调入到哪个内存块。

- 显示置换过程

使用getElementById函数，在对应的内存块内显示指令。

程序流程图

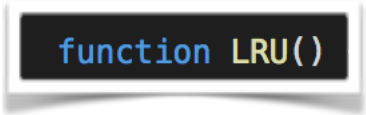


相关算法

- FIFO算法

该算法总是淘汰最先进入内存的页面，既选择在内存中驻留时间最久的页面予以淘汰。 在该算法的模拟过程中，每当页面被置换进入内存时，将置换页面所在的物理块中访问标记设为-1；并且每执行一次指令，便将物理块的访问标记自动加1，需要置换时将访问标记最大的物理块中的页面置换出去，这样能防止当物理块访问标记出现两个以上相同的值的错误执行，更好地模拟了先进先出法；

- LRU算法

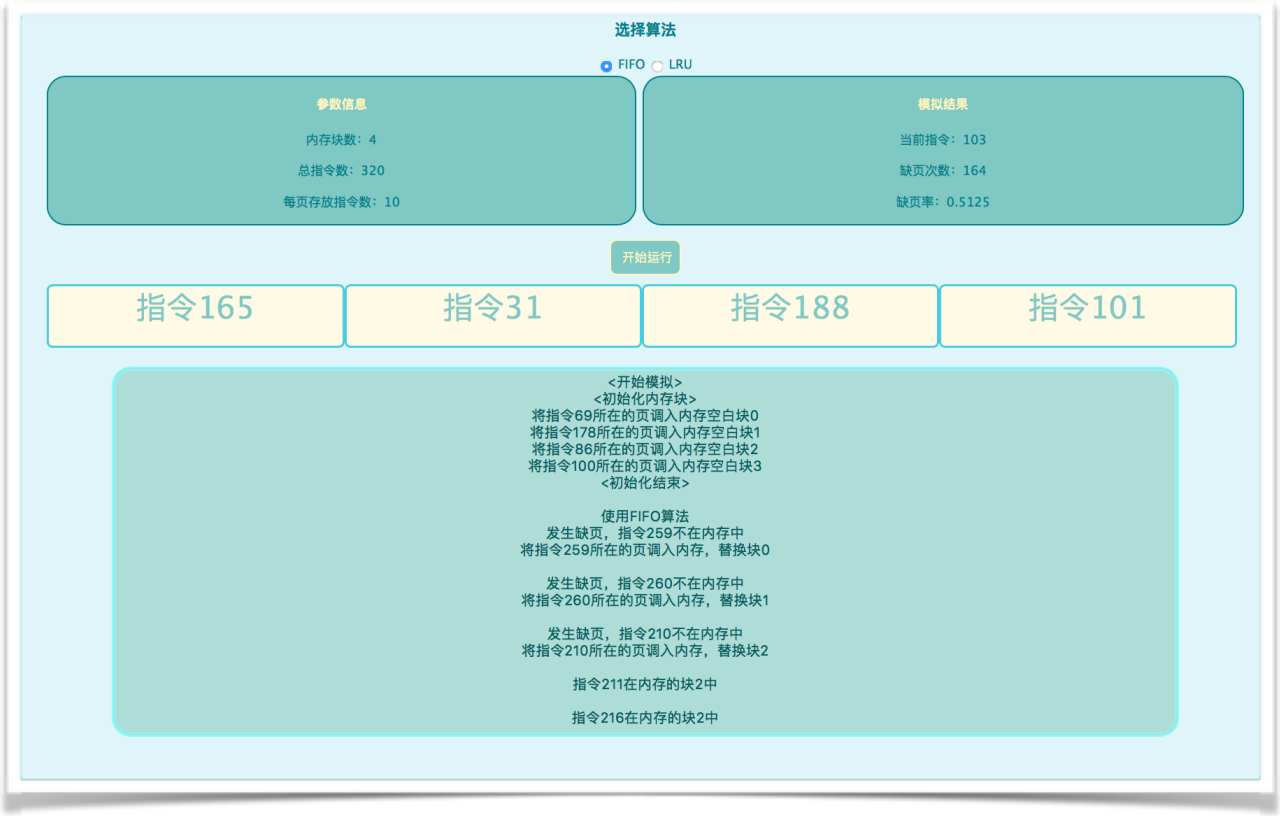


该算法以最近的过去作为不久将来的近似， 将过去最长一段时间里不曾被使用的页面置换掉。

在该算法的模拟过程中，每当物理块中的页面被访问时（包括原先存在的和后来置换进入的页面），便将其物理块访问标记置为- 1。以后每执行一条指令，便将物理块中各页面的访问标记加1，需置换时访问标记最大的便是将要被置换的。

模拟运行截图

选择FIFO算法



选择LRU算法