请求调页存储管理方式模拟

操作系统课程设计

operating system, Spring 2017

By

1552674 李源

Tongji University

School of Software Engineering

Contents

1 项目背景

1.1 项目需求

编写一个应用程序，实现请求调页存储管理方式模拟。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。采用FIFO算法或LRU算法实现置换。指令分布应该是均匀的。

1.2 项目目的

(1) 掌握页面、页表、地址转换过程；

(2) 对页面置换过程有更深的认识；

(3) 加深对请求调页系统的原理和实现过程的理解。

2 需求分析

根据项目需求，我们可以得知本项目所模拟的内存和作业分别满足如下要求：

(1) 内存：4个内存块，一个内存块中能存放10条指令；

(2) 作业：320条指令，分别放在32页中；需执行320条指令，且指令可以重复。

(3) 指令分布：50%的指令是顺序执行的，25%是均匀分布在前地址部分，25%是均匀分布在后地址部分。

(4) 置换算法：分别实现LRU算法和FIFO算法。

结合内存和作业的实际情况，我们可以设计出本项目的模拟方案。

2.1 模拟方案

该模拟方案的步骤，大致可按照如下进行：

(1) 随机从一条指令开始执行。

(2) 如果指令在内存中，则显示其物理地址，并转到下一条指令；如果不在不在内存中，则发生缺页，此时需要记录缺页次数，并将其调入内存。

(3) 如此循环，直到执行了320条指令，之后不再执行。

3 调度算法

根据前文对本模拟系统的分析，以及对两种指令情况的具体考虑，可以将该调度算法细分为两个部分：

(1) 实现内存块的置换算法，即LRU算法和FIFO算法；

(2) 实现指令的随机选择算法，可称为random算法。

3.1 FIFO算法

根据FIFO算法的定义，最先进来的页，最先被调出。则FIFO算法的关键是，将调入的页以队列的数据结构形式存储，利用队列实现先入先出。在本模拟系统中，我使用一个数组完成FIFO算法中队列的实现。FIFO算法的步骤如下：

(1) 该数组存储四个内存块进入的顺序。

(2) 初始化的时候，该数组所有的值被初始化为0。

(3) 每次执行指令时，首先检测该指令是否在某一个内存块中。

(4) 如果在某一个内存块中，则没有任何操作。

(5) 如果不在内存块中，首先查找是否有内存块为空，如果空，就放入其中。

(6) 如果没有内存块为空，则选择内存块时间为4的那一个，将其替换，替换后时间改为0，其余的内存块时间加1。

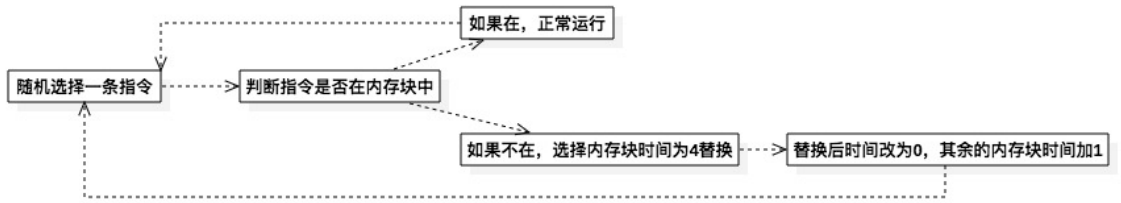


Figure 3.1: FIFO算法示意图

3.2 LRU算法

根据LRU算法的定义，使用离过去最近作为不远将来的近似，将最长时间没有使用的页置换出去。则LRU算法的关键是，选择最长时间没有使用的页，并且置换出去。在本模拟系统中，我使用一个数组完成LRU算法中对于最长时间没有使用的页寻找。LRU算法的步骤如下：

(1) 该数组存储四个内存块各自上次使用的时间。

(2) 初始化的时候，该数组所有的值被初始化为0。

(3) 每次执行指令时，首先检测该指令是否在某一个内存块中。

(4) 如果在某一个内存块中，将其该内存块的时间改为0，其余的内存块时间加1。

(5) 如果不在内存块中，找到4个内存块中时间最大的那一个，将其替换，替换后时间改为0，其余的内存块时间加1。

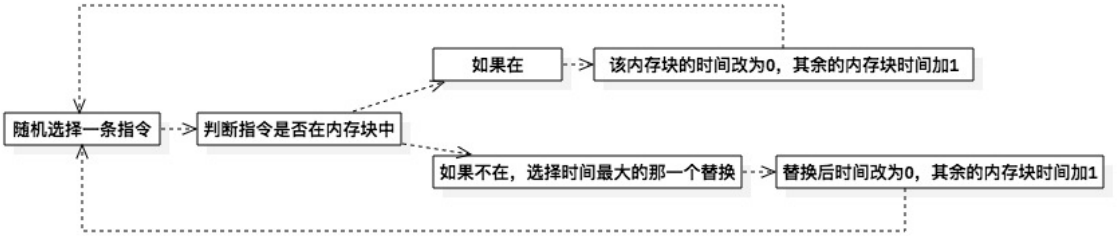


Figure 3.2: LRU算法示意图

3.3 指令随机选择算法

本算法以ppt给出的参考方法为标准，具体步骤如下：

(1) 在0至319条指令之间，随机选取一个起始执行指令。如序号为m。

(2) 顺序执行下一条指令，即序号为$m+1$的指令。

(3) 通过随机数，跳转到前地址部分0 至$m-1$中的某个指令处，其序号为m_1。

(4) 顺序执行下一条指令，即序号为m_1+1的指令。

(5) 通过随机数，跳转到后地址部分m_1+2至319中的某条指令处，其序号为m_2。

(6) 顺序执行下一条指令，即m_2+1处的指令。

(7) 重复跳转到前地址部分、顺序执行、跳转到后地址部分、顺序执行的过程，直到执行完320条指令。

其中随机数的生成，利用qsrand函数实现，具体代码如下：

```
QTime t;
t= QTime::currentTime();
qsrand(t.msec()+t.second()*1000);

int temp = qrand() % this->nowAchieve;
int temp = this->nowAchieve + qrand() % (319 - this->nowAchieve);
```

4 系统实现

根据需求分析，可以将该模拟系统的界面分为2个部分，分别为控制界面和显示界面。如下：

Figure 4.1: 系统界面图

4.1 控制界面

控制界面可选择不同的算法，”单步”表示执行一条指令，”全部”表示一次性执行完所有指令。本轮320条指令执行完后，或中途想重新开始，均可点击”重制”按钮重新开始。点击”关闭”按钮退出本模拟系统。注意，在使用一种算法后，中途不可切换算法。

Figure 4.2: 控制界面图

4.2 显示界面

显示界面可分为两部分，即内存块部分和指令部分。

Figure 4.3: 显示界面图

其中内存块部分，当某一个内存块放入了某一页后，内存块会变为”已满”的状态，且显示当前放入的逻辑页号。

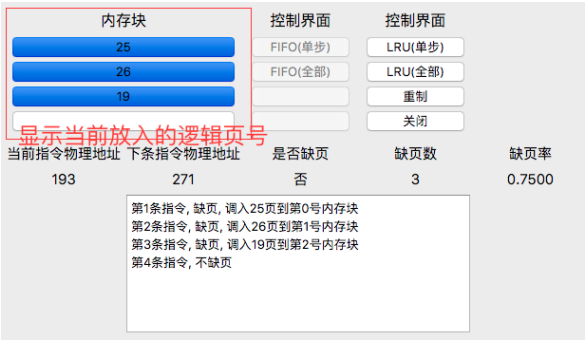


Figure 4.4: 内存块部分

而指令部分，会显示当前与下一条指令地址，当前是否出现缺页情况，总的缺页数和缺页率。同时，也可在下面查看之前的指令相关信息。 显示界面可分为两部分，即内存块部分和指令部分。

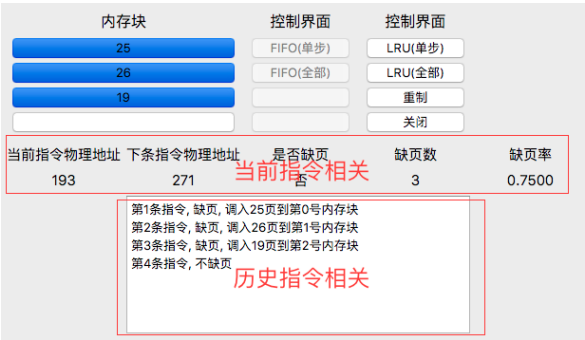
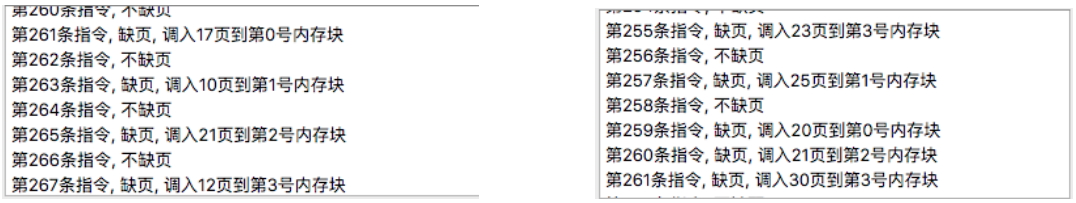


Figure 4.5: 指令部分

4.3 LRU算法与FIFO算法区别

由于本模拟系统所使用的指令随机选择算法原因，两个算法在缺页数和缺页率上的差别并不明显，其主要的差异在于，使用FIFO算法时，被替换的内存块始终按照同一顺序，如本例的0-1-2-3。而LRU算法则会没有特定的顺序。这一区别是因为其算法本质而产生的。效果如下：



(a) FIFO算法　　(b) LRU算法

Figure 4.6: 两者算法区别

5 开发环境

- 系统：macOS Sierra (version 10.12.5)
- IDE：Qt Creator 4.2.1, Based on Qt 5.8.0 (Clang 7.0 (Apple), 64 bit)
- 语言： C++

6 提交内容

- 源代码
- assignment2.zip 可执行文件压缩包（需在mac系统下使用）
- assignment2.dmg 安装包（需在mac系统下使用）
- 1552674_liyuan.pdf 说明文档