最近最久未使用LRU置换算法

在进程运行过程中，若其所要访问的页面不在内存而需要把他们调入内存，但内存已经没有空闲空间时，为了保证该进程能正常运行，系统必须从内存中调出一页程序或数据送磁盘的对换区中。应将那个页面调出，需根据一定的算法来确定；一个好的页面置换算法，应具有较低的页面更换频率。

目前存在着许多种置换算法，如最佳置换算法（Optional）、先进先出置换算法（ FIFO）、最近最久未使用置换算法（LRU）等。

①最佳置换算法（Optional）

这是一种理论上的算法，其所选的被淘汰的页面，将是以后永不使用的，或许是在最长（未来）时间内不再被访问的页面；它通常可保证获得最低的缺页率，但实际不便于实现。

②先进先出置换算法（ FIFO）

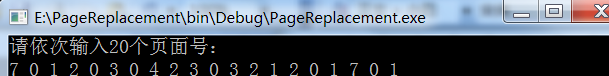
该算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰。该算法实现简单，只需把一个进程已调入内存的页面，按先后次序链接成一个队列，并设置一个指针，使它总是指向最老的页面。

③最近最久未使用置换算法（LRU）

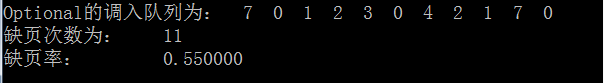
该算法是根据页面调入内存后的使用情况进行决策的，它总是选择最近最久未使用的页面予以淘汰。该算法赋予每一个页面一个访问字段，用来记录一个页面自被上次访问以来所经历的时间t，当须淘汰一个页面时选择现有页面中其t值最大的。

**三种算法的比较：**

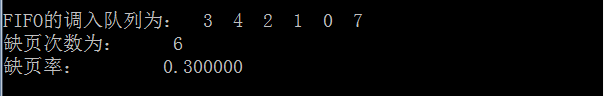
页面号：



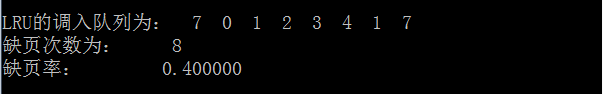
①最佳置换算法（Optional）：



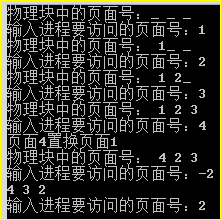
②先进先出置换算法（ FIFO）：



③最近最久未使用置换算法（LRU）：



其中LRU置换算法演示截图如下：



**四、实验总结：**

综合而来，最近最久未使用置换算法更具合理性。不过该算法需要硬件支持，即使用栈或者寄存器。其实，我认为无论是栈还是寄存器的概念都是一样的。这三个置换算法除了第一个之外，其它两个都是各有特点的。

这次实习的内容很简单，关键是理解。不过，演示时使用的是控制台人-机交互方式，可能在演示的时候很难去理解。

<https://github.com/solome/OS/tree/master/page_replacement_algorithm>