页面置换算法
(追求更少的缺页率)

最佳置换算法(OPT)

- 思想：每次选择 淘汰的页面是永不使用的页面，或最长时间不再被访问的页面
- 过程

  - 无法提前预判
  - 无法实现

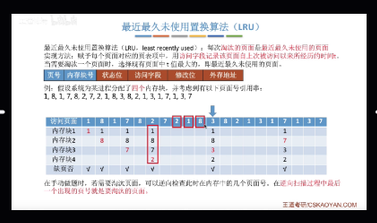
先进先出置换算法FIFO

- 思想：每次淘汰的页面是最早进入内存的页面
- 实现
  - 调入内存的页面根据调入的先后顺序排成队列，换出时选择队头元素即可
  - 队列的最大长度取决于系统为进程分配了多少内存块
- 过程

  - belady异常：进程分配的物理块数增大，缺页次数反而增加
  - 只有FIFO算法会产生belady异常，算法性能差

最近最久未使用算法LRU

- 思想：每次淘汰的是最久没有使用的页面
- 实现：每个页面的页表项中，访问字段记录该页面自上次访问以来经历的时间t。淘汰一个页面时，选择T值最大的即可
- 过程

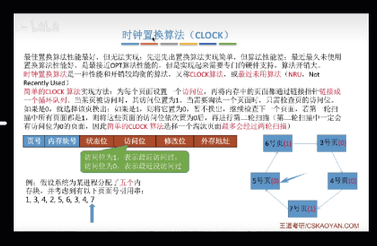


  - 性能好
  - 实现困难 开销大
  - 需要硬件实现
- 最接近最佳置换算法

时钟置换算法CLOCK (最近未用算法NRU) 一种均衡算法

简单的CLOCK

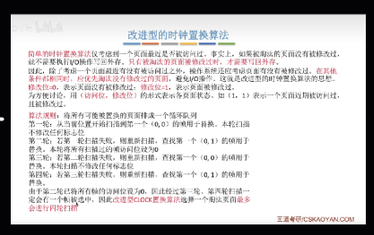
- 思想
  - 每个页面设置访问位，访问过 1 没有访问 0
  - 内存中的页面通过链接指针链接成循环队列
  - 需要淘汰页面时，检查页的访问位。
    0 换出 / 1 置为0 暂不换出继续检查下一个
    若第一轮扫描所有页面全是1，则将这些页面全置0后，进行第二轮扫描
- 过程



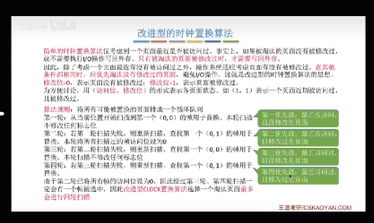
  - 只有缺页扫描的时候，指针才会移动
  - 没有缺页访问后，只会把访问为变成1，指针不变
  - 缺页换进来时，访问位为1，指针指向下一位
- 注意：如果被淘汰的页面没有被修改过，无需进行IO操作写回外存。只有淘汰的页面被修改过后，才需要写回外存
- 改进
  - 页表项中添加修改位。
  - 0表示没有被修改过1修改过
  - 优先淘汰没有被修改过的页面

改进的CLOCK

- 算法规则：扫描四轮



  - 先找00 不修改
  - 找不到 找01 并修改访问位
  - 找不到01 但是访问位已经改变 再次查找00
  - 找不到00 但是此时访问位必定有零 查找01
  - 总之 先查找修改位 然后处理访问位



回顾与思考