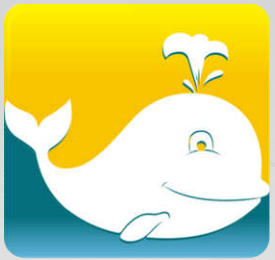


操作系统

Operating System

最优页面置换算法(OPT, optimal)

- 基本思路
- 算法实现
- 算法特征
  - 选择未来最长时间不访问的页面
  - 缺页最少，是理想情况
  - 实际系统中无法实现
  - 无法预知每个页面在下次访问前的等待时间
  - 作为置换算法的性能评价依据
    - 在模拟器上运行某个程序，并记录每一次的页面访问情况
    - 第二遍运行时使用最优算法

选择未来最长时间不访问的页面

最优页面置换算法示例

时间		0	1	2	3	4	5	6	7	8	9	10
访问请求			c	a	d	b	e	b	a	b	c	d
物理帧号	0	a	a	a	a	a	a	a	a	a	a	
	1	b	b	b	b	b	b	b	b	b	b	
	2	c	c	c	c	c	c	c	c	c	c	
	3	d	d	d	d	d	→e	e	e	e	e	
缺页状态							●					●
每页的下次访问时间							a=7 b=6 c=9 d=10					a=? b=? c=? d=?

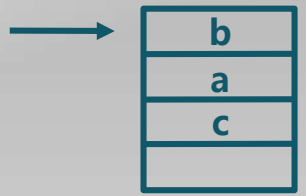
先进先出算法（First-In First-Out, FIFO）

- 思路
  - 选择在内存驻留时间最长的页面进行置换
- 实现
  - 维护一个记录所有位于内存中的逻辑页面链表
  - 链表元素按驻留内存的时间排序，链首最长，链尾最短
  - 出现缺页时，选择链首页面进行置换，新页面加到链尾
- 特征
  - 实现简单
  - 性能较差，调出的页面可能是经常访问的
  - 进程分配物理页面数增加时，缺页并不一定减少(Belady现象)
  - 很少单独使用

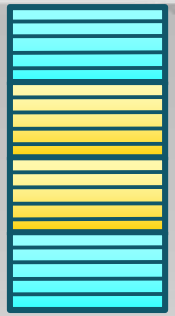
FIFO

在4个页帧中执行：

- 假定初始a- > b- > c- > d顺序



访问页面链表



进程占用物理内存

时间		0	1	2	3	4	5	6	7	8	9	10
访问请求			c	a	d	b	e	b	a	b	c	d
物理帧号	0	a	a	a	a	a	e	e	e	e	e	d
	1	b	b	b	b	b	b	b	a	a	a	a
	2	c	c	c	c	c	e	c	c	b	b	b
	3	d	d	d	d	d	d	d	d	d	c	c
缺页状态							●		●	●	●	●

最近最久未使用算法 (Least Recently Used, LRU)

- 思路
  - 选择最长时间没有被引用的页面进行置换
  - 如某些页面长时间未被访问，则它们在将来还可能会长时间不会访问
- 实现
  - 缺页时，计算内存中每个逻辑页面的上一次访问时间
  - 选择上一次使用到当前时间最长的页面
- 特征
  - 最优置换算法的一种近似

最近最未被使用算法(LRU)

置换的页面是最长时间没有被引用的

时间		0	1	2	3	4	5	6	7	8	9	10
访问请求			c	a	d	b	e	b	a	b	c	d
物理帧号	0	a	a	a	a	a	a	a	a	a	a	a
	1	b	b	b	b	b	b	b	b	b	b	b
	2	c	c	c	c	c	→ e	e	e	e	e	→ d
	3	d	d	d	d	d	d	d	d	d	→ c	c
缺页状态							●				●	●
每页的下次访问时间							a=2 b=4 c=1 d=3				a=7 b=8 e=5 d=3	a=7 b=8 e=5 c=9

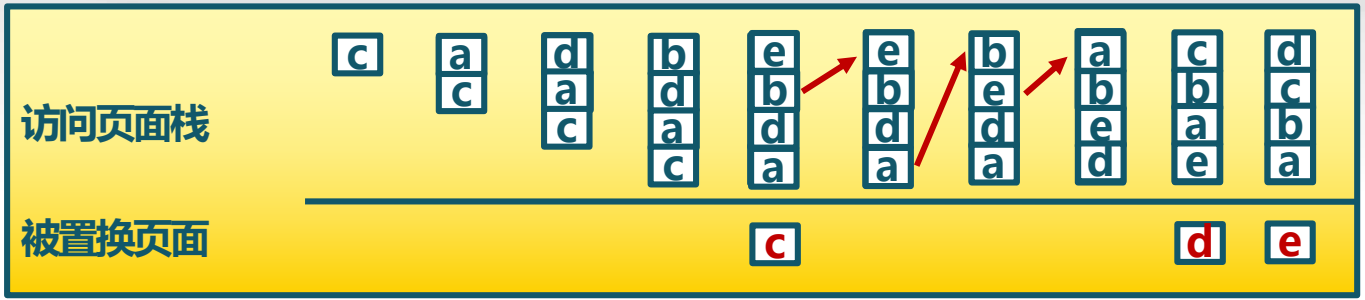
LRU算法的可能实现方法

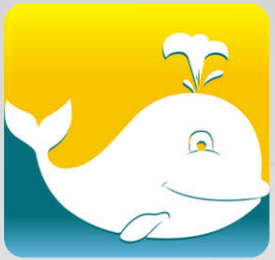
- 页面链表
  - 系统维护一个按最近一次访问时间排序的页面链表
    - 链表首节点是最近刚刚使用过的页面
    - 链表尾节点是最久未使用的页面
  - 访问内存时，找到相应页面，并把它移到链表之首
  - 缺页时，置换链表尾节点的页面
- 活动页面栈
  - 访问页面时，将此页号压入栈顶，并栈内相同的页号抽出
  - 缺页时，置换栈底的页面
- 特征
  - 开销比较大

用栈实现LRU算法

保持一个最近使用页面的“栈”

时间		0	1	2	3	4	5	6	7	8	9	10
访问请求			c	a	d	b	e	b	a	b	c	d
物理帧号	0	a	a	a	a	a	a	a	a	a	a	a
	1	b	b	b	b	b	b	b	b	b	b	b
	2	c	c	c	c	c	→ e	e	e	e	e	→ b
	3	d	d	d	d	d	d	d	d	d	→ c	c
缺页状态							●				●	●





操作系统

Operating System