

操作系统

Operating Systems

L25 内存换出

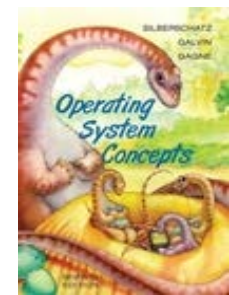
Swap out

lizhijun_os@hit.edu.cn

综合楼411室

授课教师：李治军

有换入，就应该有换出！

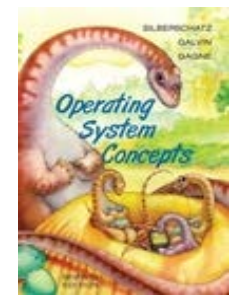


get_free_page? 还是...

```
page=get_free_page();
```

```
bread_page(page, current->executable->i_dev, nr);
```

- 并不能总是获得新的页，内存是有限的
 - 需要选择一页淘汰，换出到磁盘，选择哪一页？
 - **FIFO**，最容易想到，但如果刚换入的页马上又要换出怎么办？
 - 有没有最优的淘汰方法？ **MIN**
 - 最优淘汰方法能不能实现，是否需要近似？ **LRU**



FIFO页面置换

- 一实例: 分配了3个页框(frame), 页面引用序列为

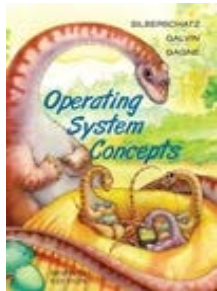
A B C A B D A D B C B

Ref: Page:	A	B	C	A	B	D	A	D	B	C	B
1	A					D				C	
2		B					A				
3			C						B		

D换A不太合适!

- 评价准则: **缺页次数**; 本实例, **FIFO**导致**7次**缺页

问题: 换谁最合适?



MIN页面置换

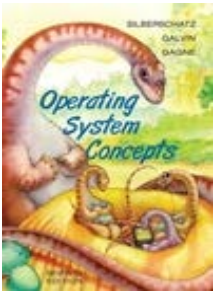
■ **MIN**算法: 选最远将使用的页淘汰, 是最优方案

■ 继续上面的实例: (3frame)**A B C A B D A D B C B**

Ref: Page:	A	B	C	A	B	D	A	D	B	C	B
1	A									C	
2		B									
3			C			D					

■ 本实例, **MIN**导致**5**次缺页

■ 可惜, **MIN**需要知道将来发生的事... 怎么办?



LRU页面置换

- 用过去的历史预测将来。LRU算法: 选最近最长一段时间没有使用的页淘汰(最近最少使用)。

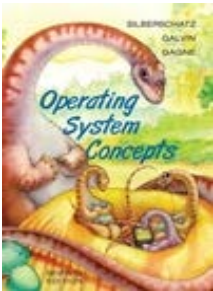
■ 继续上面的实例: (3frame) **A B C A B D A D B C B**

Ref:	A	B	C	A	B	D	A	D	B	C	B
Page:											
1	A									C	
2		B									
3			C			D					

- 本实例，LRU也导致5次缺页

和MIN完全一样!

- LRU是公认的很好的页置换算法，怎么实现?



LRU的准确实现，用时间戳

■ 每页维护一个时间戳(**time stamp**)

■ 继续上面的实例: (3frame)**A B C A B D A D B C B**

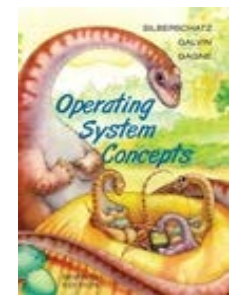
**time
stamp**

	A	B	C	A	B	D	A	D	B	C	B
A	1	1	1	4	4	4	7	7	7	7	7
B	0	2	2	2	5	5	5	5	9	9	11
C	0	0	3	3	3	3	3	3	3	10	10
D	0	0	0	0	0	6	6	8	8	8	8

选具有最小时间戳的页!

选A淘汰!

■ 每次地址访问都需要修改时间戳，需维护一个全局时钟，**需找到最小值** ... 实现代价较大



LRU准确实现，用页码栈

- 维护一个页码栈

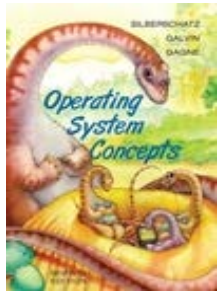
- 继续上面的实例: (3frame)**A B C A B D A D B C B**

A	B	C	A	B	D	A	D	B	C	B
		C	A	B	D	A	D	B	C	B
	B	B	C	A	B	D	A	D	B	C
A	A	A	B	C	A	B	B	A	D	D

页码栈

选栈底页淘汰!

- 每次地址访问都需要修改栈(修改10次左右栈指针) ... 实现代价仍然较大 ⇒ **LRU准确实现用的少**



LRU近似实现 – 将时间计数变为是与否

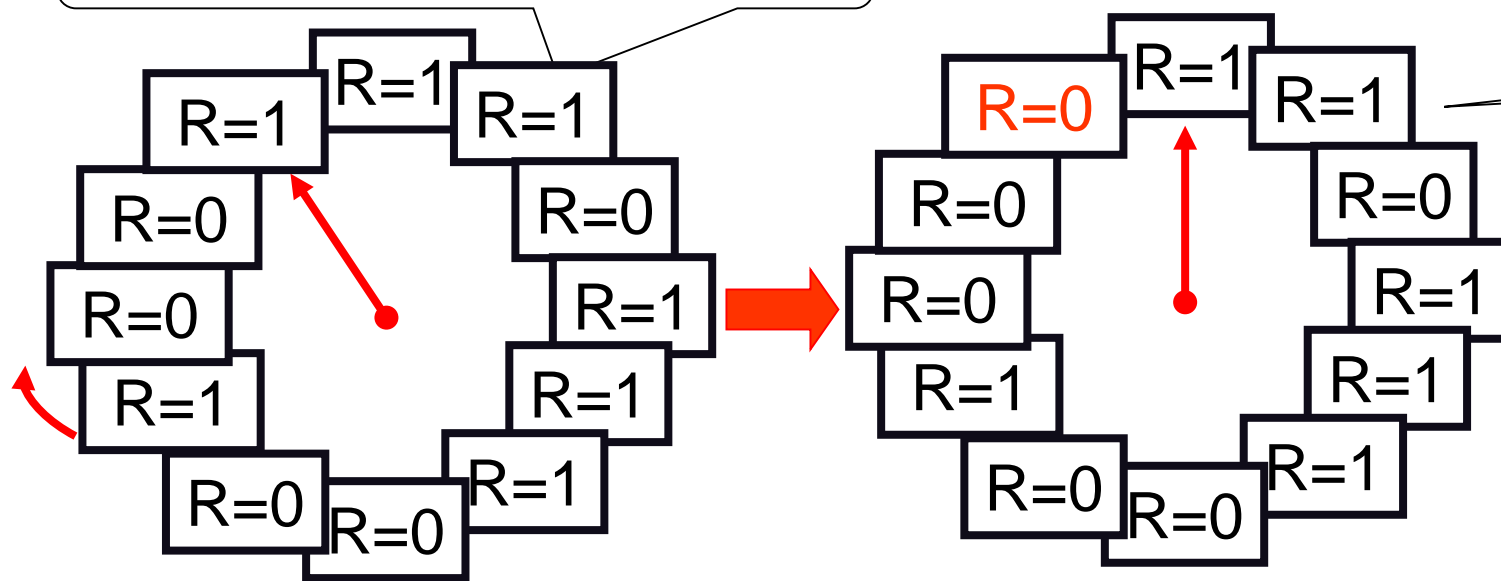
■ 每个页加一个引用位(reference bit)

■ 每次访问一页时，硬件自动设置该位

■ 选择淘汰页：扫描该位，是1时清0，并继续扫描；是0时淘汰该页

SCR这一实现方法称为
Clock Algorithm

组织成循环队列较合适！

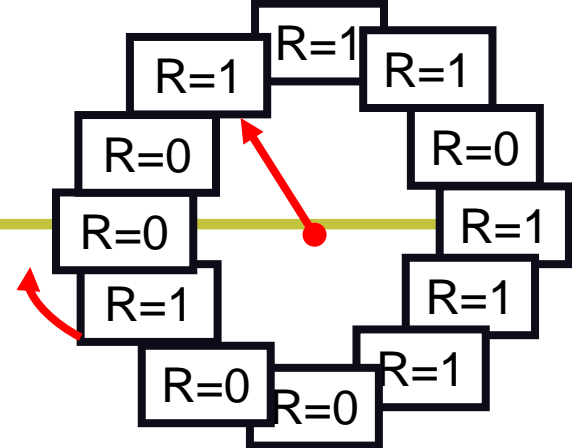


再给一次机会(**Second Chance Replacement**)



Clock算法的分析与改造

■ 如果缺页很少，会？ **所有的R=1**



■ hand scan一圈后淘汰当前页，将调入页插入hand位置，hand前移一位

■ 原因: 记录了太长的历史信息... 怎么办?

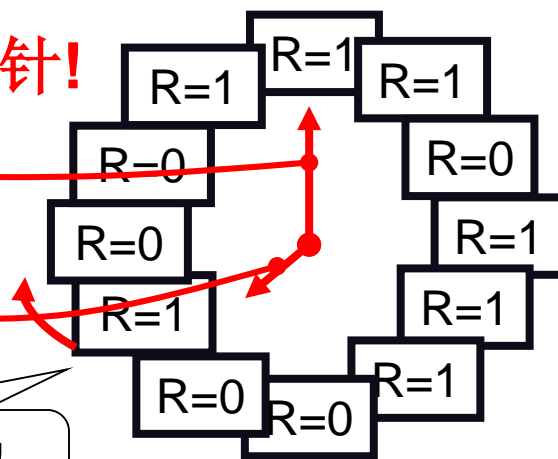
退化为FIFO!

■ 定时清除R位... **再来一个扫描指针!**

用来清除R位，移动速度要快!

用来选择淘汰页，移动速度慢!

更像Clock吧!



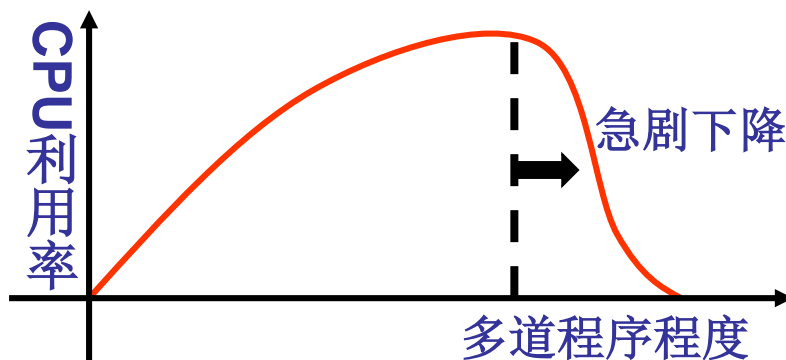
置换策略有了，还需要解决一个问题

■ 给进程分配多少页框(帧frame)

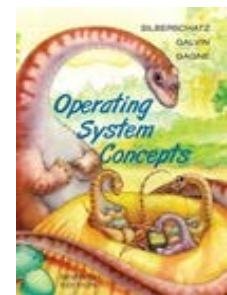
- 分配的多，请求调页导致的内存高效利用就没用了！
- 那分配的太少呢？

解释：系统内进程增多 \Rightarrow 每个进程的缺页率增大 \Rightarrow 缺页率增大到一定程度，进程总等待调页完成 \Rightarrow CPU利用率降低 \Rightarrow 进程进一步增多，缺页率更大 ...

看下面的现象：横轴是进程个数



- 称这一现象为**颠簸(thrashing)**



swap in

swap out

swap分区管理

