然而，如果R是0同时生存时间小于或等于τ，则该页面仍然在工作集中。这样就要把该页面临时保留下来，但是要记录生存时间最长（“上次使用时间”的最小值）的页面。如果扫描完整个页表却没有找到适合被淘汰的页面，也就意味着所有的页面都在工作集中。在这种情况下，如果找到了一个或者多个R＝0的页面，就淘汰生存时间最长的页面。在最坏情况下，在当前时间滴答中，所有的页面都被访问过了（也就是都有R＝1），因此就随机选择一个页面淘汰，如果有的话最好选一个干净页面。

3.4.9 工作集时钟页面置换算法

当缺页中断发生后，需要扫描整个页表才能确定被淘汰的页面，因此基本工作集算法是比较费时的。有一种改进的算法，它基于时钟算法，并且使用了工作集信息，称为WSClock（工作集时钟）算法（Carr和Hennessey，1981）。由于它实现简单，性能较好，所以在实际工作中得到了广泛应用。

与时钟算法一样，所需的数据结构是一个以页框为元素的循环表，参见图3-21a。最初，该表是空的。当装入第一个页面后，把它加到该表中。随着更多的页面的加入，它们形成一个环。每个表项包含来自基本工作集算法的上次使用时间，以及R位（已标明）和M位（未标明）。

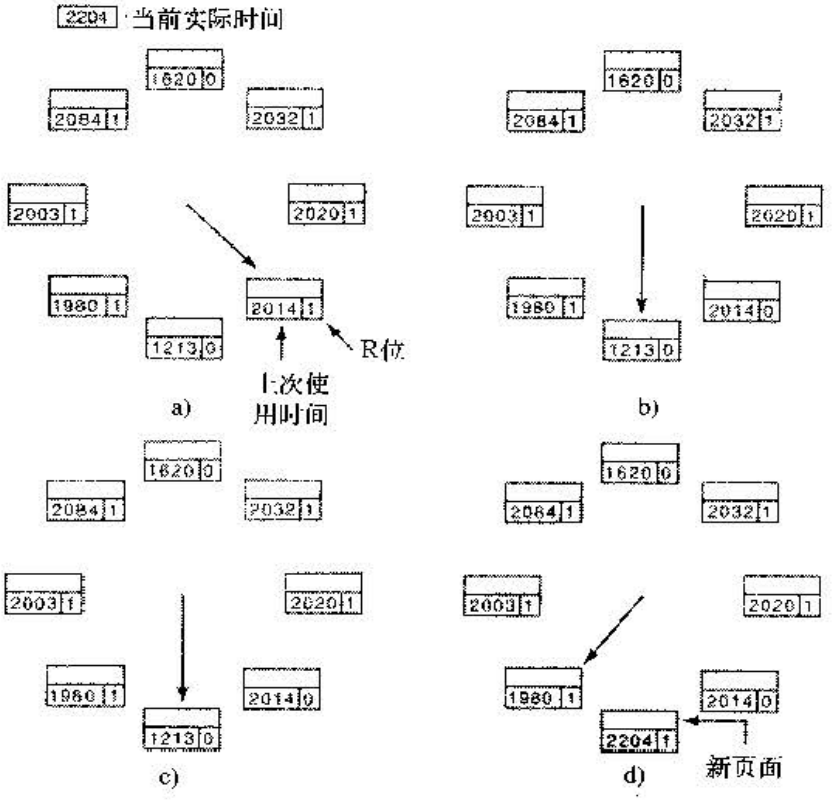

图3-21 工作集时钟页面置换算法的操作：a)和b) 给出在R=1时所发生的情形；c) 和d) 给出R=0的例子

与时钟算法一样，每次缺页中断时，首先检查指针指向的页面。如果R位被置为1，该页面在当前时钟滴答中就被使用过，那么该页面就不适合被淘汰。然后把该页面的R位置为0，指针指向下一个页面，并重复该算法。该事件序列之后的状态参见图3-21b。

现在来考虑指针指向的页面在R=0时会发生什么，参见图3-21c。如果页面的生存时间大于τ并且该页面是干净的，它就不在工作集中，并且在磁盘上有一个有效的副本。申请此页框，并把新页面放在其中，如图3-21d所示。另一方面，如果此页面被修改过，就不能立即申请页框，因为这个页面在磁盘上没有有效的副本。为了避免由于调度写磁盘操作引起的进程切换，指针继续向前走，算法继续对下一个页面进行操作。毕竟，有可能存在一个旧的且干净的页面可以立即使用。

原则上，所有的页面都有可能因为磁盘I/O在某个时钟周期被调度。为了降低磁盘阻塞，需要设置一个限制，即最大只允许写回n个页面。一旦达到该限制，就不允许调度新的写操作。

如果指针经过一圈返回它的起始点会发生什么呢？这里有两种情况：

1) 至少调度了一次写操作。