2) 没有调度过写操作。

对于第一种情况，指针仅仅是不停地移动，寻找一个干净页面。既然已经调度了一个或者多个写操作，最终会有某个写操作完成，它的页面会被标记为干净。置换遇到的第一个干净页面，这个页面不一定是第一个被调度写操作的页面，因为硬盘驱动程序为了优化性能可能已经把写操作重排序了。

对于第二种情况，所有的页面都在工作集中，否则将至少调度了一个写操作。由于缺乏额外的信息，一个简单的方法就是随便置换一个干净的页面来使用，扫描中需要记录干净页面的位置。如果不存在干净页面，就选定当前页面并把它写回磁盘。

3.4.10 页面置换算法小结

我们已经考察了多种页面置换算法，本节将对这些算法进行总结。已经讨论过的算法在图3-22中列出。

最优算法在当前页面中置换最后要访问到的页面。不幸的是，没有办法来判定哪个页面是最后一个要访问的，因此实际上该算法不能使用。然而，它可以作为衡量其他算法的基准。

NRU算法根据R位和M位的状态把页面分为四类。从编号最小的类中随机选择一个页面置换。该算法易于实现，但是性能不是很好，还存在更好的算法。

算　法	注　释
最优算法	不可实现，但可用作基准
NRU（最近未使用）算法	LRU的很粗糙的近似
FIFO（先进先出）算法	可能抛弃重要页面
第二次机会算法	比FIFO有大的改善
时钟算法	现实的
LRU（最近最少使用）算法	很优秀，但很难实现
NFU（最不经常使用）算法	LRU的相对粗略的近似
老化算法	非常近似LRU的有效算法
工作集算法	实现起来开销很大
工作集时钟算法	好的有效算法

图3-22 书中讨论过的页面置换算法

FIFO算法通过维护一个页面的链表来记录它们装入内存的顺序。淘汰的是最老的页面，但是该页面可能仍在使用，因此FIFO算法不是一个好的选择。

第二次机会算法是对FIFO算法的改进，它在移出页面前先检查该页面是否正在被使用。如果该页面正在被使用，就保留该页面。这个改进大大提高了性能。时钟算法是第二次机会算法的另一种实现。它具有相同的性能特征，而且只需要更少的执行时间。

LRU算法是一种非常优秀的算法，但是只能通过特定的硬件来实现。如果机器中没有该硬件，那么也无法使用该算法。NFU是一种近似于LRU的算法，它的性能不是非常好，然而，老化算法更近似于LRU并且可以更有效地实现，是一个很好的选择。

最后两种算法都使用了工作集。工作集算法有合理的性能，但它的实现开销较大。工作集时钟算法是它的一种变体，不仅具有良好的性能，并且还能高效地实现。

总之，最好的两种算法是老化算法和工作集时钟算法，它们分别基于LRU和工作集。它们都具有良好的页面调度性能，可以有效地实现。也存在其他一些算法，但在实际应用中，这两种算法可能是最重要的。

3.5 分页系统中的设计问题

在前几节里我们讨论了分页系统是如何工作的，并给出了一些基本的页面置换算法和如何实现它们。然而只了解基本机制是不够的。要设计一个系统，必须了解得更多才能使系统工作得更好。这两者之间的差别就像知道了怎样移动象棋的各种棋子与成为一个好棋手之间的差别。下面我们将讨论为了使分页系统达到较好的性能，操作系统设计者必须仔细考虑的一些其他问题。

3.5.1 局部分配策略与全局分配策略

在前几节中，我们讨论了在发生缺页中断时用来选择一个被置换页面的几个算法。与这个选择相关