是该算法的一个可调参数。最后，如果一个页是无效的、不在内存、共享、锁定在内存或者拥有DMA，那么它被跳过。

PFRA用一个类似时钟的算法来选择旧页面换出。这个算法的核心是一个循环，它扫描每个区域的活动和非活动列表，试图按照不同的紧迫程度回收不同类型的页面。紧迫性数值作为一个参数传递给该过程，说明花费多大的代价来回收一些页面。通常，这意味着在放弃之前检查多少个页面。

在PFRA期间，页面按照图10-18描述的方式在活动和非活动列表之间移来移去。为了维护一些启发并且尽量找出没有被引用的和近期不可能被使用的页面，PFRA为每个页面维护两个标记：活动/非活动和是否被引用。这两个标记构成四种状态，如图10-18所示。在对一个页面集合的第一遍扫描中，PFRA首先清除它们的引用位。如果在第二次运行期间确定它已经被引用，则把它提升到另一个状态，这样就不太可能回收它了。否则，将该页面移动到一个更可能被回收的状态。

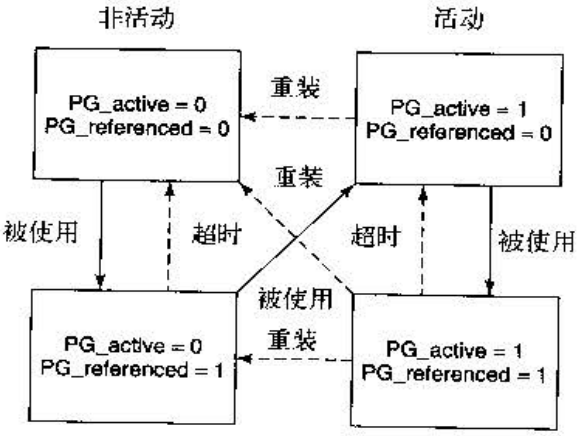

图10-18 页框置换算法中考虑的页面状态

处在非活动列表上的页面，自从上次检查未被引用过，故而是移出的最佳候选。有些页面的PG_active和PG_referenced都被置为0，如图10-18。然而，如果需要，处于其他状态的页面也可能会被回收。图10-18中的重装箭头就说明这个事实。

PFRA维护一些页面，尽管可能已经被引用但在非活动列表中，其原因是为了避免如下的情形。考虑一个进程周期性访问不同的页面，比如周期为1个小时。从最后一次循环开始被访问的页面会设置其引用标志位。然而，接下来的一个小时里不再使用它，没有理由不考虑把它作为一个回收的候选。

我们还没有提及的内存管理系统的一个方面是另一个守护进程pdflush，实际上就是一组后台守护线程。pdflush线程要么（1）周期性醒来（通常是每500ms），把非常旧的“脏（dirty）”页面写回到磁盘，要么（2）当可用的内存水平下降到一个阈值时由内核显式唤醒，把页面缓存的“脏”页面写回到磁盘。在便携模式（laptop mode）下，为了保留电池寿命，每次pdflush线程醒来，“脏”页面就被写到磁盘。“脏”页面也可以通过显式的同步请求写出到磁盘，比如通过系统调用sync、fsync或者fdatasync。更早的Linux版本使用两个单独的守护进程：kupdate，用于写回旧页面；bdflush，用于在低内存的情况下写回页面。在2.4版本内核中这个功能被整合到pdflush线程当中了。选择多线程是为了隐藏长的磁盘延迟。

10.5 Linux中的I/O系统

Linux和其他的UNIX系统一样，I/O系统都相当的简单明了。基本上，所有的I/O设备都被当作文件来处理，并且通过与访问所有文件同样的read和write系统调用来访问。在某些情况下，必须通过一个特殊的系统调用来设置设备的参数。我们会在下面的章节中学习这些细节。

10.5.1 基本概念

像所有的计算机一样，运行Linux的计算机具有磁盘、打印机、网络等I/O设备。需要一些策略才能使程序能够访问这些设备。有很多不同的方法都可以达到目的，Linux把设备当作一种特殊文件整合到文件系统中。每个I/O设备都被分配了一条路径，通常在/dev目录下。例如：一个磁盘的路径可能是“/dev/hd1”，一个打印机的路径可能是“/dev/lp”，网络的路径可能是“/dev/net”。

可以用与访问其他普通文件相同的方式来访问这些特殊文件。不需要特殊的命令或者系统调用。常用的open、read、write等系统调用就够用了。例如：下面的命令

```
cp file /dev/lp
```

把文件“file”复制到打印机“/dev/lp”，然后开始打印（假设用户具有访问“/dev/lp”的权限）。程序能够像操作普通文件那样打开、读、写特殊文件。实际上，上面的“cp”命令甚至不知道是要打印“file”