		Win32进程类优先级					
		实时	高	高于标准	标准	低于标准	空闲
Win32线程优先级	时间紧急	31	15	15	15	15	15
	最高	26	15	12	10	8	6
	高于标准	25	14	11	9	7	5
	标准	24	13	10	8	6	4
	低于标准	23	12	9	7	5	3
	最低	22	11	8	6	4	2
	空闲	16	1	1	1	1	1

图11-27 Win32优先级到Windows优先级的映射

为了使用这些优先级进行调度，系统维护一个包含32个线程列表的队列，分别对应图11-27中的0～31的不同等级。每个列表包含了就绪线程对应的优先级。基本的调度算法是从优先级队列中从31到0的从高优先级到低优先级的顺序查找。一旦一个非空的列表被找到，等待队首的线程就运行一个时间片。如果时间配额已用完，这个线程排到其优先级的队尾，而排在前面的线程就接下来运行。换句话说，当在最高的优先级有多条线程处于就绪状态，它们就按时间片轮转法来调度。如果没有就绪的线程，那么处理器空闲，并设置成低功耗状态来等待中断的发生。

值得注意的是，调度取决于线程而不是取决于线程所属的进程。因此调度程序并不是首先查看进程然后再是进程中的线程。它直接找到线程。调度程序并不考虑哪个线程属于哪个进程，除非进行线程切换时需要做地址空间的转换。

为了改进在具有大量处理器的多处理器情况下的调度算法的可伸缩性，调度管理器尽力不给全局的优先级表的数组加上一个全局的锁来实现同步访问控制。相反地，对于一个准备到CPU的线程来说，若是处理器已就位，则可以让它直接进行，而不必进行加锁操作。

对于每一个进程，调度管理器都维护了一个理想处理器（ideal processor）记录，它会在尽可能的时候让线程在这个理想处理器上运行。这改善了系统的性能，因为线程所用到的数据驻留在理想处理器的内存中。调度管理器可以感知多处理器的环境，并且每一个处理器有自己的内存，可以运行需要任意大小内存空间的程序——但是如果内存不在本地，则会花费较大的时间开销。这些系统被认为是NUMA（非统一内存地址）设备。调度管理器努力优化线程在这类计算机上的分配。当线程出现缺页错误时，内存管理器努力把属于理想处理器的NUMA节点的物理页面分配给线程。

队首的队列在图11-28中表示。这个图表明实际上有四类优先等级：实时级、用户级、零页和空闲级，即当它为-1时有效。这些值得我们深入讨论。优先级16～31属于实时级的一类，用来为构建满足实时性约束的系统。处于实时级的线程优先于任何动态分配级别的线程，但是不先于DPC和ISR。如果一个实时级的应用程序想要在系统上运行，它就要求设备驱动不能运行DPC和ISR更多的额外时间，因为这样可能导致这些实时线程错过它们的截止时间。

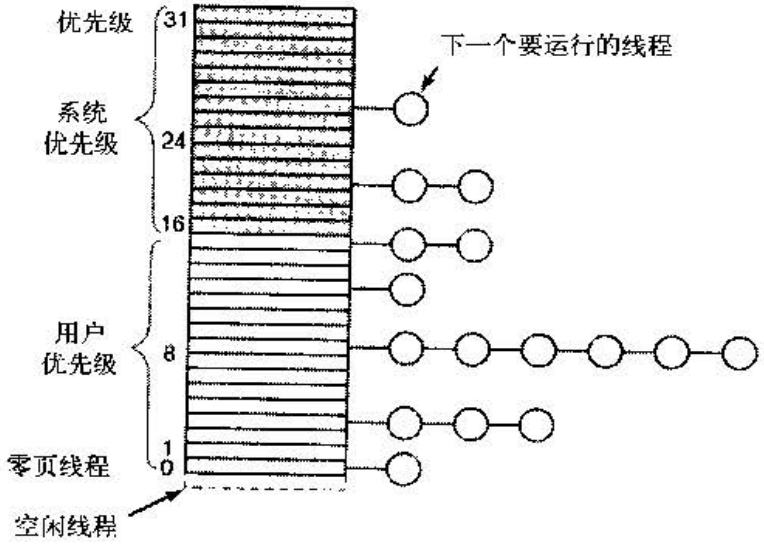

图11-28 Windows Vista为线程支持32个优先级

用户态下不能运行实时级的线程。如果一个用户级线程在一个高优先级运行，比如说，键盘或者鼠标线程进入了一个死循环，键盘或者鼠标永远得不到运行从而系统被有效地挂起。把优先级设置为实时级的权限，需要启用进程令牌中相应的特权。通常用户没有这个特权。