等待函数可使线程自愿进入等待状态，直到一个特定的内核对象变为已通知状态为止。

**WaitForSingleObject** **函数**

DWORD WaitForSingleObject(

HANDLE hObject,

     DWORD dwMilliseconds

);

第一个参数hObject标识一个能够支持被通知/未通知的内核对象（前面列出的任何一种对象都适用）。

第二个参数dwMilliseconds允许该线程指明，为了等待该对象变为已通知状态，它将等待多长时间。（INFINITE为无限时间量，INFINITE已经定义为0xFFFFFFFF（或-1））

传递INFINITE有些危险。如果对象永远不变为已通知状态，那么调用线程永远不会被唤醒，它将永远处于死锁状态，不过，它不会浪费宝贵的C P U时间。

**例子：**

DWORD dw = WaitForSingleObject(hProcess, 5000);

switch(dw)

{

   case WAIT\_OBJECT\_0:

      // The process terminated.

      break;

   case WAIT\_TIMEOUT:

      // The process did not terminate within 5000 milliseconds.

      break;

   case WAIT\_FAILED:

      // Bad call to function (invalid handle?)

      break;

}

上面这个代码告诉系统，在特定的进程（hProcess）终止运行(进程hProcess终止运行变成已经通知)之前，或者在5000m s时间结束之前，调用线程不应该变为可调度状态。

 WaitForSingleObject的返回值能够指明调用线程为什么再次变为可调度状态。

如果线程等待的对象变为已通知状态，那么返回值是WAIT\_OBJECT\_0。

如果设置的超时已经到期，则返回值是WAIT\_TIMEOUT。

如果将一个错误的值（如一个无效句柄）传递给WaitForSingleObject，那么返回值将是WAIT\_FAILED（若要了解详细信息，可调用GetLastError）。

**WaitForMultipleObjects函数**

WaitForMultipleObjects函数与WaitForSingleObject函数很相似，区别在于它允许调用线程同时查看若干个内核对象的已通知状态：

DWORD WaitForMultipleObjects(

DWORD dwCount,

   CONST HANDLE\* phObjects,

   BOOL fWaitAll,

   DWORD dwMilliseconds

);

dwCount参数用于指明想要让函数查看的内核对象的数量。这个值必须在1与MAXIMU M\_WAIT\_OBJECTS（在Windows头文件中定义为64）之间。

phObjects参数是指向内核对象句柄的数组的指针。

可以以两种不同的方式来使用WaitForMultipleObjects函数。

一种方式是让线程进入等待状态，直到指定内核对象中的任何一个变为已通知状态。

另一种方式是让线程进入等待状态，直到所有指定的内核对象都变为已通知状态。

fWaitAll参数告诉该函数，你想要让它使用何种方式。如果为该参数传递TRUE，那么在所有对象变为已通知状态之前，该函数将不允许调用线程运行。

dwMilliseconds参数的作用与它在WaitForSingleObject中的作用完全相同。如果在等待的时候规定的时间到了，那么该函数无论如何都会返回。。

WaitForMultipleObjects函数的返回值告诉调用线程，为什么它会被重新调度。可能的返回值是WAIT\_FAILED和WAIT\_TIMEOUT。如果为fWaitAl l参数传递TRUE，同时所有对象均变为已通知状态，那么返回值是WAIT\_OBJECT\_0。如果为fWaitAll传递FALSE，那么一旦任何一个对象变为已通知状态，该函数便返回。在这种情况下，你可能想要知道哪个对象变为已通知状态。返回值是WAIT\_OBJECT\_0与（WAIT\_OJECT\_0 + dwCount-1）之间的一个值。换句话说，如果返回值不是WAIT\_TIMEOUT，也不是WAIT\_FAILED，那么应该从返回值中减去WAIT\_OBJECT\_0。产生的数字是作为第二个参数传递给WaitForMultipleObjects的句柄数组中的索引。该索引说明哪个对象变为已通知状态。

下面是说明这一情况的一些示例代码

HANDLE h[3];

h[0] = hProcess1;

h[1] = hProcess2;

h[2] = hProcess3;

DWORD dw = WaitForMultipleObjects(3, h, FALSE, 5000);

switch(dw)

{

   case WAIT\_FAILED:

      // Bad call to function (invalid handle?)

      break;

   case WAIT\_TIMEOUT:

      // None of the objects became signaled within 5000 milliseconds.

      break;

   case WAIT\_OBJECT\_0 + 0:

      // The process identified by h[0] (hProcess1) terminated.

      break;

   case WAIT\_OBJECT\_0 + 1:

      // The process identified by h[1] (hProcess2) terminated.

      break;

   case WAIT\_OBJECT\_0 + 2:

      // The process identified by h[2] (hProcess3) terminated.

      break;

}

转自<http://www.cnblogs.com/fangyukuan/archive/2010/09/03/1817095.html>