What's the point of passing a never-signaled event to MsgWaitForMultipleObjects?

devblogs.microsoft.com/oldnewthing/20181114-00

November 14, 2018

Raymond Chen

In the Quake source code, there is this variable `tevent` whose usage is rather strange.

```
 49 static HANDLE   tevent;
```

It is initialized at program startup to a newly-created unsignaled event.

```
660     tevent = CreateEvent(NULL, FALSE, FALSE, NULL);
661
662     if (!tevent)
663         Sys_Error ("Couldn't create event");
```

and it is cleaned up a program shutdown:

```
267         if (tevent)
268             CloseHandle (tevent);
```

and the only use of it is in this call to `MsgWaitForMultipleObjects`:

```
535         MsgWaitForMultipleObjects(1, &tevent, FALSE, time, QS_ALLINPUT);
```

In true angry developer fashion, this is in a function with the banner

```
520 ============================================================
521
522 WINDOWS CRAP
523
524 ============================================================
```

Anyway, when the `bWaitAll` parameter is `FALSE`, the `MsgWaitForMultipleObjects` function waits for one of three things to happen:

- One of the handles is signaled,
- The queue is in a state specified by the filter, or
- The timeout elapses.

Since the code never signals the event, the first case neve occurs, so the only things that will cause `MsgWaitForMultipleObjects` to return are the second or third cases.

The dummy event is not actually necessary.

```
MsgWaitForMultipleObjects(0, NULL, FALSE, time, QS_ALLINPUT);
```

If `bWaitAll` is `TRUE`, then the `MsgWaitForMultipleObjects` function waits for one of two things to happen:

- All of the handles is signaled and the queue is in a state specified by the filter, or
- The timeout elapses.

If you pass no handles, then the first part of the first case is vacuously satisfied (due to the magic properties of the empty set), so the things that will cause the function to return are either that the queue is in a required state or the timeout elapses.

The fact that the handle count can be any value up to `MAXIMUM_WAIT_OBJECTS` *minus one* gives you some insight into the internal implementation of the `MsgWaitForMultipleObjects` function: It takes the handle array you pass, and adds another handle that is signaled when the queue is in the desired state. It then calls the `WaitForMultipleObjects` with the same `bWaitAll` parameter. That explains why passing `bWaitAll = TRUE` requires all the handles to be signaled *and* the queue to be in the requested state.

If you don't want to rely on the magical properties of the empty set, you could instead use a handle that you already know will never be signaled: You can use `GetCurrentProcess()` or `GetCurrentThread()`. The current process pseudohandle and current thread pseudohandle will become signaled when the process or thread terminates, but this is code running on that thread in that process. The thread cannot outlive itself.

Bonus chatter 2: Here's why I'm in the Quake credits.

Raymond Chen

Follow