MsgWaitForMultipleObjects and the queue state

devblogs.microsoft.com/oldnewthing/20050217-00

February 17, 2005

Raymond Chen

One danger of the `MsgWaitForMultipleObjects` function is calling it when there are already messages waiting to be processed, because `MsgWaitForMultipleObjects` returns only when there is a **new** event in the queue.

In other words, consider the following scenario:

- `PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE)` returns `TRUE` indicating that there is a message.
- Instead of processing the message, you ignore it and call `MsgWaitForMultipleObjects`.

This wait will **not** return immediately, even though there is a message in the queue. That's because the call to `PeekMessage` told you that a message was ready, and you willfully ignored it. The `MsgWaitForMultipleObjects` message tells you only when there are new messages; any message that you already knew about doesn't count.

A common variation on this is the following:

- `MsgWaitForMultipleObjects` returns that there is a message.
- You call `PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)` and process that message.
- You call `MsgWaitForMultipleObjects` to wait for more messages.

If it so happens that there were **two** messages in your queue, the `MsgWaitForMultipleObjects` does not return immediately, because there are no new messages; there is an old message you willfully ignored, however.

When `MsgWaitForMultipleObjects` tells you that there is a message in your message queue, you have to process **all** of the messages until `PeekMessage` returns `FALSE`, indicating that there are no more messages.

Note, however, that this sequence is not a problem:

- `PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE)` returns `FALSE` indicating that there is no message.

- A message is posted into your queue.
- You call `MsgWaitForMultipleObjects` and include the `QS_ALLPOSTMESSAGE` flag.

This wait does return immediately, because the incoming posted message sets the "There is a new message in the queue that nobody knows about" flag, which `QS_ALLPOSTMESSAGE` matches and therefore causes `MsgWaitForMultipleObjects` to return immediately.

The `MsgWaitForMultipleObjectsEx` function lets you pass the `MWMO_INPUTAVAILABLE` flag to indicate that it should check for previously-ignored input.

Armed with this knowledge, explain why the observed behavior with the following code is "Sometimes my program gets stuck and reports one fewer record than it should. I have to jiggle the mouse to get the value to update. After a while longer, it falls two behind, then three…"

```
// Assume that there is a worker thread that processes records and
// posts a WM_NEWRECORD message for each new record.
BOOL WaitForNRecords(HANDLE h, UINT cRecordsExpected)
{
  MSG msg;
  UINT cRecords = 0;
  while (true) {
    switch (MsgWaitForMultipleObjects(1, &h,
                        FALSE, INFINITE, QS_ALLINPUT)) {
    case WAIT_OBJECT_0:
      DoSomethingWith(h); // event has been signalled
      break;
    case WAIT_OBJECT_1:
      // we have a message - peek and dispatch it
      if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
      }
      if (SendMessage(hwndNotify, WM_GETRECORDCOUNT,
                      0, 0) >= cRecordsExpected) {
        return TRUE; // we got enough records
      }
      break;
    default:
      return FALSE; // unexpected failure
    }
  }
}
```

Raymond Chen

Follow