

Re: The basics of Windows' messages

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2006-06/msg00115.htm>

- *From:* "William DePalo [MVP VC++]" <wildd.no.spam@xxxxxxxx>
 - *Date:* Tue, 6 Jun 2006 01:50:57 -0400
-

"Ney André de Mello Zunino" <zunino@xxxxxxxx> wrote in message news:u8M4dKSIGHA.1600@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Up to this point, the thread owning the queue would do nothing else in its available time slice other than check for messages on its queue (with, e.g., `/SendMessage()`).

That's close to the truth but not required. Windows applications must pump messages or they appear "hung" or "stuck" to the user. But some applications when there are no messages pending may do some other processing that does not take much time between messages.

As soon as it realizes there is a message, the thread removes it from the queue and dispatches it (calls `/DispatchMessage()`). The code in `/DispatchMessage()` will find out which window procedure should be called and then invoke it. All this happens in the context of the thread which initially removed the message from the queue. As the message processing is over, the thread will resume its queue watching activity and the whole cycle starts over.

Yes. That's pretty much it as far as messages that are posted go. Messages may be sent as well. In the case of messages sent `_across_` processes, the sender is blocked until the receiver processes the message. When the receiver next calls `GetMessage()` or `PeekMessage()`, Windows will note that there is a sent message pending, and it will route it to the target window before fetching the next message from the queue.

Regards,
Will