The dangers of sleeping on a UI thread

devblogs.microsoft.com/oldnewthing/20060210-00

February 10, 2006

Raymond Chen

If you have a thread that owns a window, you should not be using the `Sleep` function, because that causes your thread to stop responding to messages for the duration of the sleep. This is true even for sleeps of short duration, such as sleeping for a few seconds and waking up in order to poll the state of something in the system. As we noted earlier, polling is bad for system performance, impairing the system's ability to conserve energy in low power scenarios and suffering from the magnifying effects of Terminal Server. If you're idle, stay idle. If you're busy, do your work and then go idle.

Unfortunately, I occasionally see code like the following:

```
// code in italics is wrong
// bad polling message loop
bool fQuit = false;
while (!fQuit) {
 Sleep(2000);
 CheckIfSomethingHappened();
 MSG msg;
 while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
  if (msg.message == WM_QUIT) {
   fQuit = true;
   break;
  }
  TranslateMessage(&msg);
  DispatchMessage(&msg);
 }
}
...
```

Observe that this message loop goes up to two seconds without processing messages. People aren't crazy enough to insert two-second sleeps into threads that are responsible for interacting with the end user, but they often do it for background worker threads which created hidden windows for cross-thread communication purposes. Since the thread has no visible UI, hanging for a few seconds at a time is invisible to the end user.

Until it isn't.

If the system needs to broadcast a message, it will have to wait for this sleeping thread to finally wake up and process the broadcast message. In the meantime, the component that is issuing the broadcast continues to wait. For example, the user may have double-clicked a document that requires DDE to open. The DDE process begins with a broadcast of the `WM_DDE_INITIATE` message, which stalls behind your window. Your non-responsive hidden window has just created a "Windows seems to hang for a few seconds at random intervals" bug.

Note that many people overlook that calling `CoInitialize` (possibly indirectly) to initialize a thread for STA creates a hidden window in order to perform marshalling. Consequently, a thread that is running in a single-threaded apartment must pump messages. Failing to do so will result in mysterious system-wide stalls due to the unresponsive window.

But what if you want to sleep for a period of time while processing messages? We looked at this a little while ago.

Raymond Chen

Follow