Watching thread messages disappear

devblogs.microsoft.com/oldnewthing/20050427-10

April 27, 2005

Raymond Chen

We saw last time that thread messages are eaten by modal loops. Today we'll illustrate, and then we'll try to fix it next time.

Start with our scratch program and make the following changes:

```
#include <shellapi.h>
BOOL IsThreadMessage(MSG *pmsg)
{
 if (pmsg->hwnd == NULL) {
  switch (pmsg->message) {
   case WM_APP: MessageBeep(-1); return TRUE;
  }
 }
 return FALSE;
}
// For illustration, we'll post a thread message every two seconds
DWORD CALLBACK ThreadProc(void *lpParameter)
{
 DWORD dwThread = PtrToUint(lpParameter);
 for (;;) {
  Sleep(2000);
  PostThreadMessage(dwThread, WM_APP, 0, 0);
 }
 return 0;
}
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 // Start the timer that posts the thread message
 DWORD dwThread;
 HANDLE hThread = CreateThread(NULL, 0, ThreadProc,
       UintToPtr(GetCurrentThreadId()), 0, &dwThread);
 if (!hThread) return FALSE;
 CloseHandle(hThread);
 // create some content - just to make things interesting
 g_hwndChild = CreateWindow(WC_LISTVIEW, NULL,
                            WS_CHILD | WS_VISIBLE | LVS_ICON, 0, 0, 0, 0,
                            hwnd, (HMENU)1, g_hinst, 0);
    if (!g_hwndChild) return FALSE;
    SHFILEINFO sfi;
    HIMAGELIST himl = (HIMAGELIST)SHGetFileInfo(TEXT("C:\\"), 0,
            &sfi, sizeof(sfi),
            SHGFI_SYSICONINDEX | SHGFI_DISPLAYNAME | SHGFI_LARGEICON);
    if (!himl) return FALSE;
    ListView_SetImageList(g_hwndChild, himl, LVSIL_NORMAL);
    for (int i = 0; i < 50; i++) {
     LVITEM item;
     item.iItem = i;
     item.iSubItem = 0;
     item.mask = LVIF_TEXT | LVIF_IMAGE;
     item.pszText = sfi.szDisplayName;
     item.iImage = sfi.iIcon;
     if (ListView_InsertItem(g_hwndChild, &item) < 0) return FALSE;
    }
    return TRUE;
}
void OnClose(HWND hwnd)
```

```
{
 if (MessageBox(hwnd, TEXT("Really?"), TEXT("Title"),
     MB_YESNO) == IDYES) {
  DestroyWindow(hwnd);
 }
}
// add to window procedure
    HANDLE_MSG(hwnd, WM_CLOSE, OnClose);
    while (GetMessage(&msg, NULL, 0, 0)) {
      if (!IsThreadMessage(&msg)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
      }
    }
```

Run this program and notice that it beeps every two seconds, or at least it does most of the time. If you right-click on the caption bar or grab the edge of the window to start resizing it or grab the scollbar or start a drag-selection or display a message box, the beeps stop. That's because all of those actions are modal operations, and the modal message loop is eating the thread messages.

Save this program because we'll come back to it.

The obvious solution is to post the message to the main window itself rather than to the thread. You have a window handle; use it!

```
DWORD CALLBACK ThreadProc(void *lpParameter)
{
 HWND hwnd = reinterpret_cast<HWND>(lpParameter);
 for (;;) {
  Sleep(2000);
  PostMessage(hwnd, WM_APP, 0, 0);
 }
 return 0;
}
 HANDLE hThread = CreateThread(NULL, 0, ThreadProc,
       reinterpret_cast<void*>(hwnd), 0, &dwThread);
// add to window procedure
 case WM_APP: MessageBeep(-1); return 0;
    while (GetMessage(&msg, NULL, 0, 0)) {
      ~~if (!IsThreadMessage(&msg)) {~~
        TranslateMessage(&msg);
        DispatchMessage(&msg);
      ~~}~~
    }
```

Now that that problem has been solved, I'm going to tempt fate and solve the problem **the wrong way** because I want to illustrate message filters. Next time.

Raymond Chen

Follow