Why is GetWindowLongPtr returning a garbage value on 64-bit Windows?

devblogs.microsoft.com/oldnewthing/20131226-00

December 26, 2013

Raymond Chen

A customer was running into problems with their application on 64-bit Windows 8. They claimed that on Windows 8, the `GetWindowLongPtr` is returning a garbage pointer, which causes their program to crash. The same program works fine on 64-bit Windows 7. They asked the Windows team why they broke `GetWindowLongPtr`.

An investigation of the customer's code quickly turned up the issue:

```
INT_PTR CALLBACK AwesomeDialogProc(
    HWND hdlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
  Awesome *pThis = (Awesome*)GetWindowLongPtr(hdlg, DWLP_USER);
  switch (uMsg) {
  case WM_INITDIALOG:
    pThis = (Awesome*)lParam;
    SetWindowLongPtr(hdlg, DWLP_USER, (LONG)pThis);
    ...
    return TRUE;
   case WM_COMMAND:
     if (pThis != nullptr) {
       // This line crashes with pThis = garbage nonzero value
       return pThis->OnCommand(wParam, lParam);
     }
     return FALSE;
   ...
  }
  return FALSE;
}
```

See if you can spot the problem.

The error is in the line that calls `SetWindowLongPtr`. It takes the 64-bit pointer value `pThis` and casts it to a `LONG`, which is a 32-bit integer type. This truncates the pointer and throws away the upper 32 bits of data. Therefore, when read back, the pointer looks like garbage because the top 32 bits were set to zero (or to `0xFFFFFFFF`, depending on the value of bit 31).

Windows 8 made some improvements to the memory manager, and a side effect was a seemingly harmless change to the way memory is allocated in 64-bit processes. As a result of the change, pointer values greater than 4GB are much more common, which means that the pointer truncation will actually destroy data. (In Windows 7, the default heap tended to hang out below the 2GB boundary, so the code merely truncated zeros, which is mostly harmless.)

What I found particularly interesting about this error is that the `DWL_USER` window long was specifically renamed to `DWLP_USER` in 64-bit Windows in order to force a build break. Therefore, developers had to go in and convert each separate use of `[GS]etWindowLong` with `DWL_USER` to a version that used `[GS]etWindowLongPtr` with `DWLP_USER`, being careful not to truncate the pointer.

This customer missed that last little bit about not truncating the pointer, and all they did was a global search/replace:

```
s/\bGetWindowLong\b/GetWindowLongPtr/g;
s/\bSetWindowLong\b/SetWindowLongPtr/g;
s/\bDWL_USER\b/DWLP_USER/g;
```

"There, I fixed it."

Raymond Chen

Follow