Why did the Win64 team choose the LLP64 model?

devblogs.microsoft.com/oldnewthing/20050131-00

January 31, 2005

Raymond Chen

Over on Channel 9, member Beer28 wrote, "I can't imagine there are too many problems with programs that have type widths changed." I got a good chuckle out of that and made a note to write up an entry on the Win64 data model.

The Win64 team selected the LLP64 data model, in which all integral types remain 32-bit values and only pointers expand to 64-bit values. Why?

In addition to the reasons give on that web page, another reason is that doing so avoids breaking persistence formats. For example, part of the header data for a bitmap file is defined by the following structure:

```
typedef struct tagBITMAPINFOHEADER {
        DWORD      biSize;
        LONG       biWidth;
        LONG       biHeight;
        WORD       biPlanes;
        WORD       biBitCount;
        DWORD      biCompression;
        DWORD      biSizeImage;
        LONG       biXPelsPerMeter;
        LONG       biYPelsPerMeter;
        DWORD      biClrUsed;
        DWORD      biClrImportant;
} BITMAPINFOHEADER, FAR *LPBITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

If a `LONG` expanded from a 32-bit value to a 64-bit value, it would not be possible for a 64-bit program to use this structure to parse a bitmap file.

There are persistence formats other than files. In addition to the obvious things like RPC and DCOM, registry binary blobs and shared memory blocks can also be used to transfer information between processes. If the source and destination processes are different bitness, any change to the integer sizes would result in a mismatch.

Notice that in these inter-process communication scenarios, we don't have to worry as much about the effect of a changed pointer size. Nobody in their right mind would transfer a pointer across processes: Separate address spaces mean that the pointer value is useless in any process other than the one that generated it, so why share it?

Raymond Chen

Follow