What happens to WPARAM, LPARAM, and LRESULT when they travel between 32-bit and 64-bit windows?

devblogs.microsoft.com/oldnewthing/20110629-00

June 29, 2011

Raymond Chen

The integral types `WPARAM`, `LPARAM`, and `LRESULT` are 32 bits wide on 32-bit systems and 64 bits wide on 64-bit systems. What happens when a 32-bit process sends a message to a 64-bit window or vice versa?

There's really only one choice when converting a 64-bit value to a 32-bit value: Truncation. When a 64-bit process sends a message to a 32-bit window, the 64-bit `WPARAM` and `LPARAM` values are truncated to 32 bits. Similarly, when a 64-bit window returns an `LRESULT` back to a 32-bit sender, the value is truncated.

But converting a 32-bit value to a 64-bit value introduces a choice: Do you zero-extend or sign-extend?

The answer is obvious if you remember the history of `WPARAM`, `LPARAM`, and `LRESULT`, or if you just look at the header file.

The `WPARAM` is zero-extended, while `LPARAM` and `LRESULT` are sign-extended.

If you remember that `WPARAM` used to be a `WORD` and `LPARAM` and `LRESULT` used to be `LONG`, then this follows from the fact that `WORD` is an unsigned type (therefore zero-extended) and `LONG` is a signed type (therefore sign-extended).

Even if you didn't know that, you could look it up in the header file.

```
typedef UINT_PTR WPARAM;
typedef LONG_PTR LPARAM;
typedef LONG_PTR LRESULT;
```

`UINT_PTR` is an unsigned type (therefore zero-extended) and `LONG_PTR` is a signed type (therefore sign-extended).

Raymond Chen

Follow