Why do I sometimes see redundant casts before casting to LPARAM?

devblogs.microsoft.com/oldnewthing/20041103-00

November 3, 2004

Raymond Chen

If you read through old code, you will often find casts that seem redundant.

```
SendMessage(hwndListBox, LB_ADDSTRING, 0, (LPARAM)(LPSTR)"string");
```

Why was `"string"` cast to `LPSTR`? It's already an `LPSTR`!

These are leftovers from 16-bit Windows. Recall that in 16-bit Windows, pointers were near by default. Consequently, `"string"` was a near pointer to a string. If the code had been written as

```
SendMessage(hwndListBox, LB_ADDSTRING, 0, (LPARAM)"string");
```

then it would have taken the near pointer and cast it to a `long`. Since a near pointer is a 16-bit value, the pointer would have been zero-extended to the 32-bit size of a `long`.

However, all pointers in window messages must be far pointers because the window procedure for the window might very well be implemented in a different module from the sender. Recall that near pointers are interpreted relative to the default selector, and the default selector for each module is different. Sending a near pointer to another module will result in the pointer being interpreted relative to the **recipient's** default selector, which is not the same as the **sender's** default selector.

The intermediate cast to `LPSTR` converts the near pointer to a far pointer, `LP` being the Hungarian prefix for far pointers (also known as "long pointers"). Casting a near pointer to a far pointer inserts the previously-implied default selector, so that the cast to `LPARAM` captures the full 16:16 far pointer.

Aren't you glad you don't have to worry about this any more?

Raymond Chen

Follow