A window can't have two timers with the same ID, so how do I assign an ID that nobody else is using?

devblogs.microsoft.com/oldnewthing/20191009-00

October 9, 2019

Raymond Chen

The `SetTimer` function creates a timer associated with a window. Timer IDs need to be unique, but if you have multiple pieces of code that all want to register a timer on the same window, you need to make sure they all come up with different timer IDs.

One way is to carve up the timer ID space so different components are assigned different ranges of timer IDs. But this means that if you add a new component, you'll have to assign it a new range of IDs, and you might run into the case where you've given away all the available values, and there are no more left to hand out.

And then there are components which may want to create timers on windows they didn't create. For example, you may have a windowless controls framework, and those windowless controls may need a timer, but they don't have a window to associate with the timer. They'll have to somehow share the timer ID space of the windowed host without explicitly coordinating with each other.

A common solution is to use a pointer to ensure a unique number.

According to this convention, if you need a unique ID for a timer, just allocate some memory. You don't need to allocate memory specifically for this purpose. You probably already allocated some memory, such as memory for your `this` pointer.

For as long as the memory is allocated, that pointer is uniquely yours. No other object can be assigned the same address, and you can use the pointer as the unique ID.

This technique generalizes to other ID assignment problems, as long as they are constrained to a single address space: For the ID, use the address of something that is uniquely yours.

The memory manager has unwittingly become the ID number registrar.

Raymond Chen

Follow