What can or should I do with the cursor handle returned by SetCursor?

devblogs.microsoft.com/oldnewthing/20221026-00

October 26, 2022

Raymond Chen

A customer was a bit confused by the return value of the `SetCursor` function. Why does it return the previous cursor? Does the caller own it now? Is the caller responsible for destroying it?

First we'll answer the questions: Ownership of the cursor does not change. Whoever was responsible for the cursor before you called `SetCursor` is still responsible for it.

Okay, with the answers out of the way, let's take a step back.

There are two general use cases for `SetCursor`.

One of them is for setting the cursor in response to a `WM_SETCURSOR` message. The system is telling you, "Okay, you're in charge of the cursor. Pick something." In this case, you set the cursor, and you don't care what the previous cursor was, because you're choosing the cursor now. Any old cursors are losers.

The other pattern is where you are temporarily changing the cursor (typically to an hourglass), and you want to change it back when you're done.

```
void DoLongRunningThingOnTheUIThread()
{
    HCURSOR oldCursor = SetCursor(hourglass);
    /* Do stuff that DOES NOT PUMP MESSAGES */
    SetCursor(oldCursor);
}
```

It is essential that you not pump messages because

1. If you pump messages and the user moves the mouse, then the `WM_SETCURSOR` message will change the cursor, and your hourglass will be lost. Even worse, you will restore the wrong cursor.
2. If you pump messages, then that creates the opportunity for the cursor owner to destroy the old cursor while you still have a handle to it.

The point is that the code that had set the cursor (the cursor you are temporarily replacing) has to keep the cursor handle valid until it gets a chance to change the cursor to something else. And as long as you don't pump messages, that code is not going to get another `WM_SETCURSOR` message, and therefore won't get a chance to change the cursor.

Mind you, locking up the UI thread for a long period of time is not a great idea, so even though it is a common pattern, it's a sign that your program should probably be moving the expensive work to a background thread.

Bonus reading: The effect of SetCursor lasts only until the next SetCursor.

Raymond Chen

Follow