The default error mode (SetErrorMode) is not zero

devblogs.microsoft.com/oldnewthing/20130626-00

June 26, 2013

Raymond Chen

A customer put the following code at the start of their program:

```
// If this assertion fires, then somebody else changed the error mode
// and I just overwrote it with my error mode.
ASSERT(SetErrorMode(SEM_FAILCRITICALERRORS) == 0);
```

The customer wanted to know whether it was a valid assumption that the initial error mode for a process is zero.

No it is not, and this is called out in the documentation for `SetErrorMode`:

> **Remarks**
>
> Each process has an associated error mode that indicates to the system how the application is going to respond to serious errors. A child process inherits the error mode of its parent process.

The assumption that the initial error mode is zero is therefore false.

There's another error in the above code: The call to `SetErrorMode` is placed inside an assertion. This means that in the retail build, the call *disappears*. The debug build has the error mode set to `SEM_FAILCRITICALERRORS`, but the retail build has the default error mode. They are changing the semantics in the debug build, and are headed down the slippery slope that leads to them being forced to deploy the debug version of the program into production because that's the only build that works.

Unfortunately, they may have already reached that point, because the customer asked, "Is it possible for the user to set the default error code to something other than zero, in which case this assertion would crash the client?" (Emphasis mine.)

Bonus chatter: Note that you can override error mode inheritance by passing the `CREATE_DEFAULT_ERROR_MODE` flag to the `CreateProcess` function.

Raymond Chen

Follow