

Re: Oh boy, how did we miss this...

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2004-11/0207.html>

From: L D Blake (*not_at_any.adr*)

Date: 11/08/04

Date: Mon, 08 Nov 2004 05:56:15 -0500

On 08 Nov 2004 08:14:32 GMT, vbdis@aol.com (VBDis) wrote:

*>Now I've tried to explore the error handling, by dereferencing an nil pointer.
>This error seems to trigger SEH immediately, but no handler responded because I
>had intentionally excluded SysUtils.*

Borland made a mistake with their exception handling... They are trying to use the OS's own reporting mechanism in a way that was never intended (at least not by M\$).

There are several different error handling mechanisms in Windows...

- 1) Structured Exceptions
- 2) Error reporting
- 3) Function returns
- 4) Vectored Exceptions

The last one is new in 2k/XP only but the others have been in use for a very long time. They are well established windows functions that (one would hope) are very well understood.

Structured exceptions are used to report Asynchronous errors... stuff that happens outside our code, out there in the real world... here's a list of some of the stuff windows thinks it should report by SEH...

```
STATUS_ACCESS_VIOLATION = $C0000005;
STATUS_NO_MEMORY = $C0000017;
STATUS_ARRAY_BOUNDS_EXCEEDED = $C000008C;
STATUS_FLOAT_DENORMAL_OPERAND = $C000008D;
STATUS_FLOAT_DIVIDE_BY_ZERO = $C000008E;
STATUS_FLOAT_INEXACT_RESULT = $C000008F;
STATUS_FLOAT_INVALID_OPERATION = $C0000090;
STATUS_FLOAT_OVERFLOW = $C0000091;
STATUS_FLOAT_STACK_CHECK = $C0000092;
STATUS_FLOAT_UNDERFLOW = $C0000093;
STATUS_INTEGER_DIVIDE_BY_ZERO = $C0000094;
STATUS_INTEGER_OVERFLOW = $C0000095;
STATUS_PRIVILEGED_INSTRUCTION = $C0000096;
```

comp.lang.pascal.delphi.misc: Re: Oh boy, how did we miss this...

```
STATUS_STACK_OVERFLOW = $C00000FD;  
STATUS_CONTROL_C_EXIT = $C000013A;
```

Now take a look at this... none of these are exactly trivial errors, especially in a system of modern complexity. Divide by 0 is not a runtime error (no matter what Borland says it is) it's the CPU telling you it's in trouble... look at the list, access violations, overflows, underflows, priveledged instructions... not trivial at all.

The second level is that of error reporting (GetLastError). The concept here is simple. Each subroutine generates an error code and stores it for the procedure to call up on demand. This is a loosely synchronous error handling as the errors can only arise from programmed actions but we are not under any immediate obligation to check the results ... bad flags, uninitialized pointers and such. These codes generally i/o related do not really signal 'system in trouble' and don't generate exceptions...

The third level is enacted by the return values of almost every WinApi call. The general convention is that a returned 0 means "oh oh", although there are a few situational differences. This is tightly synchronous error checking, it must be done immediately after calling the procedure.

As much as I hate to agree with M\$, the fact is they've chosen wisely in the way they've set this up. They could have simply had *everything* generate exceptions, but there's really no much point stopping a system because of trivial errors. Exceptions are errors that windows thinks are of such severity as to need *immediate* attention... and if they don't get the attention they need your appy will be shut down by Windows.

Borland has chosen to overlay this important error reporting mechanism with a layer of *deliberate obfuscation* that in fact hides a lot of sins in their code. Take a look at the way objects are created... why do they need an entirely separate exception handler in their new instance code? Then to make it worse they split the handlers across two files --something any programmer worth salt would never do-- leaving the potential for disaster when exceptions don't get handled by Delphi. No Delphi doesn't translate exceptions into runtime errors... It IGNORES them and windows shuts down the appy.

Error handling by objects that, by the way, don't even have the error codes in them... is a ludicrous perversion of a perfectly sound and necessary error reporting scheme. It is even worse when it quits working if you don't load the right units...

If I wrote code like that even 10 years ago I'd have been fired by everyone in the company who could fire me. Today it's a "feature"... really, I don't care how much you like Delphi, you seriously do need to ask yourself... what's wrong with this picture?

Laura

Re: Oh boy, how did we miss this...