

Re: Try Finally...

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2004-10/0993.html>

From: VBDis (vbdis_at_aol.com)

Date: 10/26/04

Date: 26 Oct 2004 21:27:47 GMT

Im Artikel <ql6sn0lbsj1tak9ee030snd8ou27bpjdvj@4ax.com>, L D Blake <not@any.adr> schreibt:

>Nope... *it's bad design... there's no excuse for releasing a language in which keywords can and do simply stop working.*

Some of the "keywords" are not hard coded. Writing procedures named Halt, Break or Exit can result in strange effects, but sometimes it's nice to override the default behaviour. Just recently I wrote my own Halt procedure, when porting a legacy command line application. Now a Halt doesn't really halt the program, instead it raises an exception. This exception can be caught at the main program level, and with a breakpoint in my Halt procedure I can inspect the stack to determine the location that caused the halt in the application code.

An Exit procedure also would be nice, to emulate a Return(value) construct, but this had to be a local subroutine, with access to the Result variable of the procedure-to-exit. You are much more informed on low level procedures, so would you see an way to code an Exit(value) procedure that would behave like Exit, i.e. exit the calling procedure, and /additionally/ would make the given value the Result of the function? In most cases it would be sufficient to put that value into EAX, where it may already reside depending on the calling convention, but then no further compiler created code should override that register with the Result variable...

>>*Why don't you just provide the necessary routines to replace the ones found in SysUtils ?*
>
>*And put them where? In another unit, so that try/except can once again drop dead because you didn't include this new unit?*

Okay, this is a problem. But not a big one, as long as any types or constants, used in exception handling, are defined in exactly that unit. Every "on Exception do" requires a definition for "Exception". You also prove that user defined exception classes must not necessarily derive from Exception, at least not in the current Delphi versions. IMO this is also very uncommon coding style, asking for trouble. Such a situation is worth to note, of course, but please in such a context that everybody can see what's wrong with such code, and why it is.

DoDi