Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by David Fanning on Thu, 05 Aug 2004 17:31:48 GMT
View Forum Message <> Reply to Message

M. Katz writes:

> So thank goodness for the control-C in Unix. What do Windows IDL
> programmers do when the mouse cursor disappears, and the IDL window
> becomes unresponsive? -- Besides "End Task"?

I usually kick the dog who is always lying at my feet,
but the dog bite, combined with the sucky interface,
doesn't really make my day any better. :-(

> I also wrestle with the issue that every time I have to close and
> re-launch IDLDE on Windows I have to re-open all of the program
> libraries I was just working with. It's a time-consuming pain.

I usually just go down to the Recent Projects menu item
and everything miraculous reappears just like I left it.
I don't work with "program libraries", I just use the Path. :-)

> But in
> Unix (where I don't use IDLDE) I just keep all of the text editor
> windows open and running in a different application (BBEdit on Mac).
> Is there a Windows setting for this I'm unaware of.

I've heard of people using EMACS on Windows and doing
all their editing there. You just have to get used to
IDL asking you if you want to "reload" all the live
long day!

Cheers,

David

P.S. Let's just say YES I want to reload...a gun!!
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by Michael Wallace on Thu, 05 Aug 2004 19:48:46 GMT
View Forum Message <> Reply to Message

Personally, I do ALL of my development work on *nix. Once the program

has been completely tested out on *nix, I will then check that it runs
the same way on Windows. All of the core logic has been debugged by
this point, so the only errors which would arise would be ones if
platform specific code managed to accidentally creep in. I also do the
same thing for my Java, Python, et al. programs -- develop on Linux and
test completely, then move them to Windows and make sure they still work
the same way. In short, I just develop by the credo "Spend as little
time as possible on Windows."

-Mike

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by David Fanning on Thu, 05 Aug 2004 20:00:20 GMT

View Forum Message <> Reply to Message

Michael Wallace writes:

> Personally, I do ALL of my development work on *nix. Once the program
> has been completely tested out on *nix, I will then check that it runs
> the same way on Windows. All of the core logic has been debugged by
> this point, so the only errors which would arise would be ones if
> platform specific code managed to accidentally creep in.

Well, I do just the opposite and it works out about
the same. Except my code always seems to have "core
logic" problems on *both* platforms. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by Michael Wallace on Fri, 06 Aug 2004 03:47:58 GMT

View Forum Message <> Reply to Message

> Well, I do just the opposite and it works out about
> the same. Except my code always seems to have "core
> logic" problems on *both* platforms. :-(

Reminds me of the old Java programmer's mantra:
Write once, debug everywhere.

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by R.G. Stockwell on Sat, 07 Aug 2004 01:24:08 GMT
View Forum Message <> Reply to Message

"M. Katz" <MKatz843@onebox.com> wrote in message
news:4a097d6a.0408050919.3874969a@posting.google.com...
...

> I do all of my development on Unix (Apple) and then run experiments on
> a Windows machine.

That is the exact opposite of me. I develop under windows in the
awesome development environment, and run them on several unix machines.
Which I peak in on with vnc.
I never use the development environment under *nix, which is, um, less awesome. [1]

> So thank goodness for the control-C in Unix. What do Windows IDL
> programmers do when the mouse cursor disappears, and the IDL window
> becomes unresponsive? >

You do know that in windows it is Control-Break, not control-C right?
I assume you meant that. I find that it often stops when you ask it to,
under windows. However a tight loop will ignore you. Perhaps a
wait command deep in the loop, with the smalles wait possible would help
[in a debuggin situation].
A print command almost always catches a break. And of course, the actual
solution in windows is to put a stop command at the offending place, before
the function call or so, and step into it.

Cheers,
bob

[1] the phrase "blows monkeychunks" comes to mind

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by Ben Panter on Sat, 07 Aug 2004 08:38:01 GMT
View Forum Message <> Reply to Message

R.G. Stockwell wrote:

> You do know that in windows it is Control-Break, not control-C right?
> I assume you meant that. I find that it often stops when you ask it to,
> under windows. However a tight loop will ignore you. Perhaps a

> wait command deep in the loop, with the smalles wait possible would help
> [in a debuggin situation].

I've used the following to stop runaway code in the IDLDE...

press the "break" button (which gets ignored)
attempt to close the IDLDE environment
reply "no"

It then appears to process the break command *before* it restarts the program. This method has worked for me a number of times, although no guarantee is offered!

Ben

--
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Ben Panter, Edinburgh, UK
email is false. Use my name (no spaces) at bigfoot which is a com.

Subject: Re: IDL on Windows vs. Unix, debugging consideration
Posted by Mark Hadfield on Sun, 08 Aug 2004 22:26:07 GMT

View Forum Message <> Reply to Message

Ben Panter wrote:
> R.G. Stockwell wrote:
>
>> You do know that in windows it is Control-Break, not control-C right?
>> I assume you meant that. I find that it often stops when you ask it to,
>> under windows. However a tight loop will ignore you. Perhaps a
>> wait command deep in the loop, with the smalles wait possible would help
>> [in a debuggin situation].
>
>
> I've used the following to stop runaway code in the IDLDE...
>
> # press the "break" button (which gets ignored)
> # attempt to close the IDLDE environment
> # reply "no"
>
> It then appears to process the break command *before* it restarts the
> program. This method has worked for me a number of times, although no
> guarantee is offered!

Clever!

If you know beforehand that your code might run away and you want to put in an interruptible point (as distinct from a break point) the following will do nicely:

```
    void = widget_event(/NOWAIT)
```

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.
More included text than new text. Blah blah.