Folks,

I'm not sure if this is a question or a report. But a couple of
things I've discovered:

1. I'm writing a file with a rather long filename. When I get
finished with it, I compile it, and go down to the command
line to run it. I type the first couple of letters of its
name, then hit CNTL-space. I see a list of the modules in the
file. I choose the appropriate one. The name of the program
is completed on the command line. Now I hit the CR.

Nothing. The "command" disappears. But nothing appears
in the Console View, and my program doesn't run. The command
doesn't show up in my command history. Is this a bug?

2. In Windows IDLDE in 6.4, if I put a breakpoint in a file
and run the file, I hit the breakpoint and I can step through
the file. Typically, I discover the thing I was trying to find,
make a couple of changes in the file, hit the Save and Compile
icon, hit the remove all breakpoints button, and off I go again.

In the IDL Workbench, my file pointer, even after I recompile
the file, is still inside the file. I can't do much of anything
until I remember I have to type RETALL. I last typed RETALL this
much back in the 1960s. Is this possibly a bug? I really prefer
when I re-compile that my file pointer be removed from the file.

Oh, wait a minute. Maybe this is a problem with the file pointer
not being updated correctly. I think the recompile actually
removes the file pointer, but I don't get visual confirmation
of that fact. (I think we talked about this before and agreed
it was a bug, now that I think about it.) Anyway, RETALL
refreshes the file enough to remove the file pointer in the file.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

On Jan 10, 4:45 pm, Paul van Delst <Paul.vanDe...@noaa.gov> wrote:
> Bob Crawford wrote:
>> On Jan 10, 3:43 pm, Abraham campbell <abra...@ittvis.com> wrote:
>>> There is a difference between multi-line paste and single-line paste.
>>> When you paste multiple lines into the command line window, it
>>> immediately executes each line, one at a time, without displaying the
>>> lines in the command line, and without you being able to modify the
>>> command before it's executed. (NOTE: The last line of a multi-line
>>> paste is also executed immediately.) However, with a single-line paste,
>>> the command is left in the command line window, and not immediately
>>> executed, so that you can modify it before executing it.
>
>>> What you might be running into is the case where you have a single
>>> statement, but you have also copied a newline character at the end of
>>> the statement. When you paste this into the command line, it will
>>> behave like a multi-line paste and immediately execute the (single)
>>> command and not give you a chance to edit the command before execution.
>
>>> The presence of a carriage return/newline character anywhere in the
>>> pasted text determines the difference between multi-line and single-line
>>> paste behavior.
>
>>> Abraham- Hide quoted text -
>
>>> - Show quoted text -
>
>> If that is the case, then the copy/paste behaviour could be made more
>> intuitive/consistent if when there is only one carriage return/newline
>> character in the text (ie it's a single-line paste that just happens
>> to have a newline at the end) it gets stripped off before pasting.
>
> I don't see how that follows. Sometimes I copy/paste single lines and I want the newline
> included. Other times I don't. It depends.

It was merely a suggestion on how to get the copy/paste function to
work as intended (as given by Abraham's first paragraph description)
by handing the case of accidental newline characters being copied on
otherwise single-line copy/paste functions.

... and imho hitting the <enter> key after all single-line copy/pastes
is much less hassle than having a single-line pasted command
unintentionally execute (when I wanted to modify it).

> All the c.l.i-p ng traffic about the Workbench would seem to indicate that letting the
> application make the choices behind-the-scenes is just bad mojo of the 10ft pole variety.

I would agree with that generally - but in this case it's the user who
may be inadvertantly telling the Workbench the wrong intention - cr/nl
characters being largely invisible and easily copied by accident (at
least on Windows)

Bob.

Subject: Re: IDL Workbench Nits
Posted by Paul Van Delst[1] on Fri, 11 Jan 2008 17:55:38 GMT

View Forum Message <> Reply to Message

Bob Crawford wrote:
> On Jan 10, 4:45 pm, Paul van Delst <Paul.vanDe...@noaa.gov> wrote:

[snip]

>> All the c.l.i-p ng traffic about the Workbench would seem to indicate that letting the
>> application make the choices behind-the-scenes is just bad mojo of the 10ft pole variety.
>
> I would agree with that generally - but in this case it's the user who
> may be inadvertantly telling the Workbench the wrong intention - cr/nl
> characters being largely invisible and easily copied by accident (at
> least on Windows)

Ah, o.k. I live in the *nix world and 99.999% of the time it's very obvious if the cr/nl
is included in a copy/paste; if the highlighted portion extends all the way to the right
edge of the window or text box, then the cr/nl is included. If it just extends to the end
of the selected text, then no cr/nl. Quite simple really.

But then, when I sit in front of a Windows machine, I'm (a) lost, and (b) dumbfounded how
difficult these sorts of little things are (e.g. for default Windows behaviour at least,
why on earth do I have to type ctrl-C to copy text when I just selected it? Isn't the act
of selecting text enough to tell the system "Hey! Copy this into the buffer!" :o)

Maybe that is a fundamental issue with stuff like eclipse - to be a truly cross-platform
GUI must it compromise on typical usage/interaction patterns and habits that people have
with their system of choice?

cheers,

paulv