Subject: Re: IDLWAVE 4.5
Posted by John-David T. Smith on Wed, 18 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Kenneth Mankoff wrote:
>
> On Wed, 18 Oct 2000, Martin Schultz wrote:
>> Karsten Rodenacker wrote:
>>> Carsten Dominik schrieb:
>>>> IDLWAVE 4.5 fixes a few minor bugs in the object support.
>>>> (reported by JD Smith, as usually - is anybody else using this?).
>>>> http://www.strw.leidenuniv.nl/~dominik/Tools/idlwave
>>>> - Carsten
>>>
>>> I am surprised that there is so little feedback on the tremendous work
>>> Carsten is doing with that (X)emacs mode idlwave! I think most users are
>>> happy with the state reached with idlde and are not aware of the
>>> possibilities of this emacs mode.
>>
>> Ohhhh. I for one just looove the idlwave emacs mode, and I am not
>> using anything else for editing of my IDL programs. However, I am
>> almost ashamed to reply to this thread, because I still haven't found
>> the time to play around with the shell within idlwave, and so I miss
>> out probably at least 80% of what idlwave can offer.
>
> Hi,
>
> I've spent some time useing the shell. The main problem that i run across
> is that now i have my emacs keys and my idl keys messed up. up-arrow
> doesn't recall my last IDL command anymore, it just moves up in the
> emacs text-buffer. Other than a few problems like that, it works quite
> nicely. Does anyone out there have a good workaround for this?
>
> -ken.

Like you, I was loathe to give up my basic xterm running IDL for the built-in emacs shell. Here's what finally convinced me to switch (among many, many others):

1. Compiling: No more typing ".run myfile" a million times. C-c C-d C-c. Instant compilation. Nothing more natural when you've just edited a file. Compile error? The line is highlighted. Fix my dumb syntax mistake, run C-c C-d C-c again. Repeat until it compiles successfully. This one is an *enormous* time savings.

2. Abbreviations: All of the abbreviations you love (like \ine -- see http://www.strw.LeidenUniv.nl/~dominik/Tools/idlwave/idlwave .html#TOC14 if you don't know what I mean) are available in the shell. You can

easily add your own too.

3. Routine info, and instant online help: The same help features you love from IDLWAVE buffers are available in the shell, as is the routine info. No more fishing through manuals to recall the ordering of those arguments. Instant, and I mean instant, access is available whether you're writing a program or running one. example:

IDL> a=histogram([C-c ?] and you get:

Usage: Result = HISTOGRAM(Array)
Keywords: BINSIZE INPUT MAX MIN NAN OMAX OMIN REVERSE_INDICES
Source: Builtin

The Keywords are all blue, so I can right click on them (the color lets me know they have a corresponding topic in the help file). Suppose I right-click on "REVERSE_INDICES"... up pops the help buffer with:

> REVERSE_INDICES
> Set this keyword to a named variable in which
> the list of reverse indices is returned.
> This list is returned as a longword vector .....

all queued up. You can just as easily see routine information and view the source of any IDL procedure, builtin, shipped with IDL, written by you or others... anything.

4. Completion: Hit tab in the shell and a routine name, system variable, field, filename, etc. is completed. It's even smart enough to complete class names after
IDL> a=obj_new('[Tab]

and keywords to that classes' Init function after

IDL> a=obj_new('myClass',[Tab]

Those of you who use a shell like bash or tcsh that does completion usually find that you hit Tab instinctively about every 10 keystrokes, and the same is true in IDLWAVE's shell. This is a *major* time saver. Examples.

IDL> a=readf[Tab]

and instantly I get

IDL> a=readfits(

A quick check to [C-c ?] tells me the file name is expected first, so I

begin typing it:

IDL> a=readfits('~/myfiles/[Tab]

now all my files are popped into a completion buffer (exactly the same as if you were loading the file into emacs). I can choose one, or get better completion by typing a few more characters, etc.

5. Compiling regions: Ever wanted to test out a bit of code but didn't because it couldn't fit on one IDL command line (like an entire if then begin endif else ... block)? You can simply compile and run *regions* if you use the shell. Just highlight it and [C-c C-d C-e].

6. Debugging. I admit it. I too used to be a "when in doubt sprinkle print and help statements liberally" debugger. No longer (ok, usually no longer). Setting, deleting, viewing, and modifying breaks is so easy with IDLWAVE and the shell that I just don't have an excuse anymore. And being able to Shift-middle click on any variable name to have it's value printed saves time by the bushel. Not to mention quickly navigating the calling stack upon break or error (I suppose I already mentioned that in a prior post).

7. History. The comint mode IDLWAVE uses for its shell saves the entire history (doesn't use IDL's history). This means the history survives restarting IDL (an unfortunately common occurence). And powerful history searching (among many other features) is built right into comint. I too missed my arrow keys for history recall, which is why I added:

```
(setq comint-scroll-to-bottom-on-input t)
(setq comint-scroll-show-maximum-output t)
(define-key idlwave-shell-mode-map [up]   'comint-previous-input)
(define-key idlwave-shell-mode-map [down] 'comint-next-input)
```

into my idlwave-shell-mode-hook (see http://www.strw.LeidenUniv.nl/~dominik/Tools/idlwave/idlwave .html#TOC40 for info on how this works). For help on any of these variables do [C-h v] and type its name -- don't forget to use tab completion to make your life easy. Now I get the best of both worlds. I can use Emacs' powerful editing features on my old input and output, collecting things together to construct a new command, for instance. I can keep commands in registers and pop them out whenever needed, etc., etc. And when on the command line, I can use up arrow just like I used to. There are many more options to get comint to behave just as you like. See the info (C-h i) for emacs->Shell Mode.

8. Shadow listing with likeliness ranking: a fancy way of saying IDLWAVE detects those eminently bothersome times when more than one

routine of the same name is on your path or compiled in (or builtin).
You see this immediately with routine_info, and can even generate a full
shadow listing for everything idlwave knows about (which is a fair bit
more than IDL knows about). This works even without the shell running,
but if it is, it checks *compiled* routines also! Especially nice when
someone dumps a big package on you to test out and you just can't seem
to get it running, or can't run your old stuff alongside it.

I could go on and on but I won't. Emacs and all this lisp business is
intimidating at first, but you really don't need to know any of that to
get all the benefits. Customization usually just involves cut and paste
to your .emacs file. Don't let the parentheses scare you. Give it a
try, and you'll see why I spend so much time raving about it.

Thanks Carsten!

Good luck,

JD

P.S. And for those of you afraid you won't be able to install idlwave...
it is simple. All you do is download:

ftp://ftp.strw.leidenuniv.nl/pub/dominik/idlwave/idlwave.tar .gz
ftp://ftp.strw.leidenuniv.nl/pub/dominik/idlwave/idlwave-hel p.tar.gz

to /tmp (or anywhere), then do (as root):

```
% tar xzvf idlwave.tar.gz
% cd idlwave-version
% tar xzvf ../idlwave-help.tar.gz
% make
% make install-all
```

where "version" is something like 4.5. This by default puts idlwave in
/usr/local/share/emacs/site-lisp/, which is a fine place. It also puts
the help files in /usr/local/etc, which is also fine. (Carsten had to
separate these for copyright issues, not to punish us).

Now simply add the following to the top of your .emacs file. (Note
possible changes):

```
;;;; IDLWAVE setup
; Comment the following out if emacs already knows about
; where idlwave is installed.  Try M-x idlwave-mode to see.
; You can also change the path if you've put it somewhere strange.
(setq load-path (cons "/usr/local/share/emacs/site-lisp/" load-path))
```

```
; Change the following to point to the idlwave help files,
; if installed somewhere other than the default.
(setq idlwave-help-directory "/usr/local/etc")

; Font lock, not just for IDLWAVE... mmmm colors
; comment out if you're boring.
(global-font-lock-mode 1)
; Uncomment the following if you commented the above,
; but would like font-lock to operate in IDLWAVE mode.
;(add-hook 'idlwave-mode-hook 'turn-on-font-lock)

; Here we ensure sure IDLWAVE is loaded for .pro files...
(autoload 'idlwave-mode "idlwave" "IDLWAVE Mode" t)
(autoload 'idlwave-shell "idlw-shell" "IDLWAVE Shell" t)
(setq auto-mode-alist
 (cons '("\\.pro\\'" . idlwave-mode) auto-mode-alist))
;;;;;;;;;;;;;;;;;;
```

That's it to get going. See
http://www.strw.LeidenUniv.nl/~dominik/Tools/idlwave/idlwave .html#SEC40
for more customization fun. (I for one insist on (setq
idlwave-main-block-indent 3) -- with due deference to David Fanning's
coding style). And don't forget to build your personal catalog of
routines (menu IDLWAVE->Routine Info->Select Catalog Directories) for
maximum enjoyment!

Hope you made it this far.

--
J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |
