Subject: Re: Saving IDL History Across Sessions...
Posted by Ken Mankoff on Thu, 05 May 2005 04:43:02 GMT
View Forum Message <> Reply to Message

On Thu, 4 May 2005, Robi wrote:
> who died and made emacs the required editor for an observer to use?
vi? Just kidding {dons flame-retardant suit}.

What platform are you on? What environment do you use normally? xterm? idlde? Terminal.app? I'm going to assume xterm on *nix.

Who said your users need to be aware that they are in emacs? How is the IDLWAVE shell any different from an xterm + IDL session? (Answer: it is much better!) What about this in an xterm?:
emacs -nw -f idlwave-shell

The only problem with this is if a routine crashes, they will find themselves looking at source code (the line the error occurred on) rather than a normal IDL error message. You could probably turn this feature off (ask JD), or just put a big sticky-note on the monitor telling them to hit the three necessary keys to return to the shell. And some would say that is a feature not a problem.

> But, point is, you definitely can't fill the recall buffer when
> IDL is started up, so unless you're using IDLWAVE you're outta
> luck. Just wanted to be sure there were no clever ways around it.
> Thanks for the responses.

I can think of some other solutions, much uglier hacks than the above, but i'll list them just for brainstorming sake:

Write a wrapper (AppleScript, windows Macro, something else?) that takes the journal and simulates a user re-typing each command.

This could work, or might not work (if a file you try to read has been removed), or could be slow (if some commands are computationally long) or could be disastrous (if you overwrite files). But it can be done.

If commands are slow, and you don't want to actually execute them, have your wrapper precede each command by a bogus character (line number of journal, letter 'x', whatever (ex: 42_print, 'foo'). Now they will all fail (and therefore run very quickly) to run a previous command, I have to press up-arrow to get to it, and then delete the 42_. Better than your current situation maybe?

If you want to run the actual command, so that your data/vars are valid, this would be an improvement on IDLWAVE. But now you run into

issues with non-existent files you want to load, or writing files
you don't want to overwrite.

To stop yourself from overwriting files,

```
cd, 'safe_dir'
reload_cmds    ; external macro/AppleScript inputter
cd, '..'
```

and that will stop files from getting destroyed. Unless, of course,
someone wrote something out with an absolute path.

You could re-mount the disk in read-only mode, reload commands, and
then re-mount in read-write mode. Ack! Ugly.

You could tell people not to quit IDL. Now you don't need to reload
the commands!

You could build a wrapper to IDL that pipes everything through the
Unix tee command (or RPC?) to a 2nd IDL session. Now even if they
quit there is a 'backup' IDL that is identical. I'm not sure how you
could restore the backup.

In short, I think there are multiple solutions out there. Some
elegant, some ugly. Some are easier for you to implement (IDLWAVE)
than others (re-mounting disks) and others are easier for your users
(IDLWAVE) than others (not quitting).

-k.
