

Re: Fortran77 & GUI

Source: <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2004-05/0564.html>

From: FCC (fcc509_at_netscape.net)

Date: 05/14/04

Date: Fri, 14 May 2004 11:52:52 +0200

All is well... But, assuming the user makes a mistake in the input file, he has to run the code first, check the error message that pops up on the screen (or in the log file), understand it well enough to know how to correct his mistake.

But this is not all that simple for everyone, including people who are very familiar with developing similar codes. Imagine an error message composed of only 5 words: How could you possibly be well informed based on such small amount of information, particularly when so many input parameters are interacting (say, for large enough codes which address many phenomena) ? Even a person who is perfectly literate in say, finite element codes, can be confused with the input file, and also with the error message he receives, unless the error message is a paragraph describing all that could possibly be causing it.

A GUI, on the other hand, is incremental, that is, it will analyze the options selected so far and prevent all the following irrelevant ones from being selected (by e.g. dimming them), which is sort of "driving" the user to the correct input. It can, and should, display important information regarding the last selected option as well. I think such an approach will save many users a lot of effort and time.

Besides, there is nothing that prevents a GUI front end from saving all this information selected by the user to a valid, error-free input file, in which case all your arguments below will apply as well :) .

Regards,

FCC

Kevin G. Rhoads wrote:

```
>>>The reason being the interactive command line based input is time  
>>>consuming and warrants errorfree input (and hard to change them)  
>  
>  
> I have run into this issue numerous times. My preferred solution,  
> which may not be the best for you, is to make interactive command
```

comp.lang.fortran: Re: Fortran77 & GUI

- > *line input the fall-back which the program attempts if a file of*
- > *input data is not specified. By moving input off to a file, there*
- > *are several advantages:*
- > *1) by use of various OPENs in conditionals, the input unit can be*
- > *bound to either the console window or to the file -- no changes*
- > *are needed to existing READ statements, provided they already specify*
- > *a unit and only trivial changes are needed to add this if they don't,*
- > *meanwhile -- the file contains the responses the user would have supplied*
- > *interactively*
- > *2) this is easy to test using command-line redirection with NO*
- > *code changes on many platforms*
- > *3) input errors are reduced since the file can be edited in advance*
- > *and new, modified runs can be made by copying an input file and*
- > *modifying it*
- > *4) with a trivial extension, the program can be made to take multiple*
- > *command files and do each run in order*
- > *5) the existing interactive input is preserved as an option, which may*
- > *be an advantage in some situations*
- >
- > *Putting a GUI front-end on what is essentially a batch number crunch*
- > *program is, to my taste, not a good approach.*
- >
- > *YMMV*
- > *Kevin*