Subject: Re: file paths
Posted by Tim Patterson on Wed, 02 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

I have a similar problem. What I ended up doing was
to have subdirectories below my source code directory
and defining an environment variable $CASPER_SRC_DIR
to point at the top level, then set up a file
that would define my !path like so...

```
!path = !path+":"+$
  getenv('CASPER_SRC_DIR')+'modules/' + ':'+$
  getenv('CASPER_SRC_DIR')+'fov/' + ':'+$
  getenv('CASPER_SRC_DIR')+'kernels/' + ':'+$
  getenv('CASPER_SRC_DIR')+'plot/' + ':'+$
  getenv('CASPER_SRC_DIR')+'batch/' + ':'+$
  getenv('CASPER_SRC_DIR')+'utils/' + ':'+$
  getenv('CASPER_SRC_DIR')+'spice/' + ':'+$
[....]
  getenv('CASPER_SRC_DIR')+'target/' +':'+$
  getenv('CASPER_SRC_DIR')+'includes/'
```

Then I have a compilation routine, which executes this
at the very beginning and has a list of files to compile
as:

```
.size 65000 65000
@$CASPER_SRC_DIR/casper_path
.run casper_main
.run casper_makefiles
.run casper_widget
.run casper_toplevel
[...]
.run casper_final_routine
print, 'All compiled - starting Casper'
Casper_Main
```

and so on. I can move the files around as much as I want
as long as I make sure all the subdirectories are in my
path definition file and as I create new routines I
just tack them on the end of the compilation routine.
(Which also saves eveything as a precompiled binary, btw.
As I have over 200 routines it takes a fair while to
have to compile it ech time)

I tried a lot of fancier things, but this was the simplest

and easiest method I could come up with. Unfortunately
.run *.pro doesn't seem work which would have made my life
much easier, and you can't use EXECUTE to .RUN a file, and
then RESOLVE routines only work if you have one routine per
file.

If you find something better, let me know!

Tim

PS Yep, the tool is called Casper.

David wrote:
>
> hi,
>
> I want to add some code at the top of a fairly large project that will
> tell IDL where to look for all the subroutines it'll need. They are in
> various different folders, organized by their function. I want the code
> to compile everything automatically, even if I move the source around.
> (It would be messy to put all the subroutines in the same folder).
>
> Is there any way other than help, /source to show the path of the
> currently executing file? If I knew the path to the executing file, I
> could append it (with a +) to !path to make this work, right?
>
> Thanks in advance for any help!
>
> cheers!
> dave
>
> ***
> David Katz
> Esquimalt Defence Research Detachment
> co-op student
>

Subject: Re: file paths
Posted by Phil Williams on Thu, 03 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David wrote:

>
> hi,
>
> I want to add some code at the top of a fairly large project that will
> tell IDL where to look for all the subroutines it'll need. They are in
> various different folders, organized by their function. I want the code
> to compile everything automatically, even if I move the source around.
> (It would be messy to put all the subroutines in the same folder).
>
> Is there any way other than help, /source to show the path of the
> currently executing file? If I knew the path to the executing file, I
> could append it (with a +) to !path to make this work, right?
>
> Thanks in advance for any help!
>
> cheers!
> dave

You could search a given directory, and any subdirs off that one for the file, get the path once, and if found, and then append that to the !path var. "Boy," you say, "that's a lot of work Phil." Yes, but I already have a routine that does just that!

Check out <http://www.irc.chmcc.org/idl/philsIDL.html> and take a look at fileExists.pro. You'll also need direxists.pro (which is a hack of some of David Fanning's code). Here's the header:

```
function fileExists, file, path, $
                FOLLOW_DIR = FOLLOW_DIR, $
                GET_PATH = GET_PATH
;+
; NAME: FILEEXISTS.PRO
;
; PURPOSE: Determines the existence of a file.
;
; CATEGORY: File I/O
;
; CALLING SEQUENCE: result = fileExists(file[, path, /follow_dir,
/get_path])
;
; INPUTS:
;    file : a string containing the file name
;
; OPTIONAL INPUTS:
;   path : The given path to search. If not given then searching
;          begins at the current directory
;
; KEYWORD PARAMETERS:
```

```
;  FOLLOW_DIR : If set then subdirectories are recursively searched.
;  GET_PATH   : If set and file exists then the path to the file is
;              returned.
; OUTPUTS:
;   If GET_PATH is specified and the search was successful then the
;   path to the file is returned. Otherwise 1 is returned if the file
;   exists and 0 if the file does not.
;
; OPTIONAL OUTPUTS: none
;
; COMMON BLOCKS: none.
;
; SIDE EFFECTS: none.
;
; RESTRICTIONS: none.
;
; PROCEDURE:
;    IDL> t = fileExists('file.pro',/follow,/get)
;
; MODIFICATION HISTORY:
;    13 Dec 96 Initial Coding. PMW
;    29 Dec 96 Added FOLLOW_DIR keyword for recursive searches
;    30 Dec 96 Added GET_PATH keyword to return path if file is found.
;-
```

Just a suggestion. Hope this is what you had in mind.

Phil

```
--
 /*********************************************************** ********/
  Phil Williams, Ph.D.
  Research Instructor
  Children's Hospital Medical Center    "One man gathers what
  Imaging Research Center                another man spills..."
  3333 Burnet Ave.                      -The Grateful Dead
  Cincinnati, OH 45229
  email: williams@irc.chmcc.org
  URL: http://scuttle.chmcc.org/~williams/
 /*********************************************************** ********/
```

Subject: Re: file paths
Posted by David Foster on Thu, 03 Apr 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David wrote:
>

> hi,
>
> I want to add some code at the top of a fairly large project that will
> tell IDL where to look for all the subroutines it'll need. They are in
> various different folders, organized by their function. I want the code
> to compile everything automatically, even if I move the source around.
> (It would be messy to put all the subroutines in the same folder).
>
> Is there any way other than help, /source to show the path of the
> currently executing file? If I knew the path to the executing file, I
> could append it (with a +) to !path to make this work, right?
>

Maybe I'm really missing the boat on this one, but I think you would
be better off organizing your source code so that everything is
included in your definition of !PATH from the start. Part of the
beauty of IDL is that you don't have to tell it where to find
routines. Once you do this, you can pre-compile routines in an
application by just using:

```
 @filename.pro
```

If you really need to add dirs to !PATH "on the fly", I like
Tim's approach using getenv() (maybe just once though!).

I don't understand the last statement, since if a file is executing
then IDL must have found it, which means that it's directory
is *already* in !PATH.
--

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~
  David S. Foster         Univ. of California, San Diego
   Programmer/Analyst     Brain Image Analysis Laboratory
   foster@bial1.ucsd.edu  Department of Psychiatry
   (619) 622-5892         8950 Via La Jolla Drive, Suite 2200
                          La Jolla, CA  92037
                          [ UCSD Mail Code 0949 ]
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~
```