Subject: Re: Which like command for IDL?
Posted by davidf on Fri, 05 Jan 2001 00:33:54 GMT
View Forum Message <> Reply to Message

Jason P. Meyers (jpm7934@cis.rit.edu) writes:

> I was recently working with some examples from Dave Fanning's book
> (nice book Dave) when I realized it would be nice to have an IDL program
> that works like the unix "which" command. I had typed in the
> Display.pro program on page 66 only to find out that there is some other
> program (I suspect an RSI demo) also called Display.pro elsewhere on my
> IDL path. It would sure be nice to be able to quickly identify which
> program IDL is using from the path.
>
> Does anyone out there know of such a beast for IDL?

Try this:

```
IDL> Help, /Source
```

That should do it. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Which like command for IDL?
Posted by Richard French on Fri, 05 Jan 2001 01:51:53 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Jason P. Meyers (jpm7934@cis.rit.edu) writes:
>
>> I was recently working with some examples from Dave Fanning's book
>> (nice book Dave) when I realized it would be nice to have an IDL program
>> that works like the unix "which" command. I had typed in the
>> Display.pro program on page 66 only to find out that there is some other
>> program (I suspect an RSI demo) also called Display.pro elsewhere on my
>> IDL path. It would sure be nice to be able to quickly identify which

>> program IDL is using from the path.
>>
>>   Does anyone out there know of such a beast for IDL?
>
> Try this:
>
>   IDL> Help, /Source
>
>
That does the job for a routine you've already compiled, but what if you
want to find out BEFORE you compile it which display.pro you would end
up running? Or even better (JD can probably do this with a recursive
one-line routine), how about a procedure that goes through your full
path and finds all of the duplicate filenames? I've often wanted to do
this so that I could do some preemptive file renaming, but I haven't
taken
on the task yet. I am sure that I have about six different routines
called CIRCLE.PRO in libraries that I have gotten from people, and
having
a routine that figured this out in advance would be a nice thing.
 I got burned a year ago (UNIX system) by storing old versions
of programs in a subdirectory called OLD. It turns out that the !Path
was set up in such a way that the procedures in the OLD directory
had precedence over the new ones. Took me a long time to figure
out why it was that the changes I was putting in my program never got
executed!
Dick

Subject: Re: Which like command for IDL?
Posted by davidf on Fri, 05 Jan 2001 02:55:27 GMT
View Forum Message <> Reply to Message

Richard G. French (rfrench@wellesley.edu) writes:

> That does the job for a routine you've already compiled, but what if you
> want to find out BEFORE you compile it which display.pro you would end
> up running?

Can't be done. Heisenberg uncertainty principle applies.
You can't possibly know WHICH routine you are using until
the wave function collapses. Or, was that an electron? Humm.
Can't remember ... :-(

> Or even better (JD can probably do this with a recursive
> one-line routine), how about a procedure that goes through your full
> path and finds all of the duplicate filenames?

I'm shocked that you think JD can write better recursive
functions than me. But I'm busy. I'll leave it to him. :-)

> I am sure that I have about six different routines
> called CIRCLE.PRO in libraries that I have gotten from people, and
> having
> a routine that figured this out in advance would be a nice thing.

How about this WHICH program. *Very* quick and dirty.
I'll leave it to others to make perfect. (I can already
think of a couple of ways it can be greatly improved.)
But this finds the *first* program with the given name
of all the names I tested it on. Of course, it can
only find library routines. :-)

```
  IDL> Which, "arrow"
     D:\RSI\IDL54\lib\arrow.pro

  IDL> Which, "congrid", /Func
     D:\RSI\IDL54\lib\congrid.pro
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

```
 ************************************************************ ******
PRO WHICH, routineName, Funct=funct

Catch, theError
IF theError NE 0 THEN BEGIN
   Catch, /Cancel
   answer = Dialog_Message("Can't find: " + StrUpCase(routineName) + $
      '. Is this a function?', /Question)
   IF StrUpCase(answer) EQ 'YES' THEN BEGIN
      Catch, theError
      IF theError NE 0 THEN BEGIN
         Catch, /Cancel
         ok = Dialog_Message("Sorry. Can't find: " + $
            StrUpCase(routineName) + '. Returning')
         RETURN
      ENDIF
      Which, routineName, /Funct
   RETURN
   ENDIF ELSE BEGIN
      ok = Dialog_Message("Sorry. Can't find: " + $
         StrUpCase(routineName) + '. Returning')
      RETURN
   ENDELSE
ENDIF

Resolve_Routine, routineName, Is_Function=Keyword_Set(funct)
Help, /Source, Output=text

array = StrPos(STRUPCASE(text), STRUPCASE(routineName) + "  ", -1)
index = Where(array NE -1, count)
IF count GT 0 THEN Print, text[index[0]] ELSE Print, 'Undetermined'
END
 ************************************************************ ******
```

Subject: Re: Which like command for IDL?
Posted by davidf on Fri, 05 Jan 2001 03:32:00 GMT

View Forum Message <> Reply to Message

David Fanning (davidf@dfanning.com) writes:

> How about this WHICH program. *Very* quick and dirty.
> I'll leave it to others to make perfect. (I can already
> think of a couple of ways it can be greatly improved.)

Oh, bother! :-(

Those guys at RSI have gone and done it again. They
have already improved on my wonderful Which wesult.
See the FILE_WHICH routine in the IDL 5.4 library,
which finds the right file *without* collapsing
the wave function.

Those guys really take all the fun out of it, don't
they. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Subject: Re: Which like command for IDL?
Posted by Martin Schultz on Fri, 05 Jan 2001 09:42:32 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> David Fanning (davidf@dfanning.com) writes:
>
>> How about this WHICH program. *Very* quick and dirty.
>> I'll leave it to others to make perfect. (I can already
>> think of a couple of ways it can be greatly improved.)
>
> Oh, bother! :-(
>
> Those guys at RSI have gone and done it again. They
> have already improved on my wonderful Which wesult.
> See the FILE_WHICH routine in the IDL 5.4 library,
> which finds the right file *without* collapsing
> the wave function.
>
> Those guys really take all the fun out of it, don't
> they. :-(

Hi David,

this brings up an interesting legal/buisiness question: If
file_which is an IDL file (.pro) distributed with 5.4 is it then
permitted to use this routine with earlier versions of IDL (provided
it works)? Legally, I would think this should be doable at least as
long as you are under a maintenance contract, i.e. you could
theoretically run 5.4 but haven't installed it for whatever reason.
Without a current maintenance contract, there would be the question if
such a new library routine belongs to IDL5.4 or to IDL in general.
Buisineswise, it might be a smart move of RSI to put their library
routines under the open source license and post them own their web
site. Nobody without a licensed IDL program can use these routines
anyway, and it could be regarded as a nice service to the users if
they can profit from new developments in the library routines.

Just a thought,

Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[

Subject: Re: Which like command for IDL?
Posted by davidf on Fri, 05 Jan 2001 15:15:06 GMT
View Forum Message <> Reply to Message

Martin Schultz (martin.schultz@dkrz.de) writes:

>   this brings up an interesting legal/buisiness question: If
> file_which is an IDL file (.pro) distributed with 5.4 is it then
> permitted to use this routine with earlier versions of IDL (provided
> it works)?

I've no idea about this, although I have to imagine that
the good folks at RSI have better things to do than run
around harassing customers about using RSI-supplied
programs written in IDL. :-)

It is a moot point anyway, in this case, since the program
uses some of the neat new SWITCH, BREAK, etc. stuff that
comes in IDL 5.4, and will not compile in earlier versions.

Cheers,

David

P.S. Let's just say for all of you poor relations out
there, we have the Coyote library and its crummy WHICH
program. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Which like command for IDL?

Posted by John-David T. Smith on Fri, 05 Jan 2001 15:21:49 GMT
View Forum Message <> Reply to Message

I hate to be a one tune band, but the best way I have of doing this kind of shadow mapping (finding files defining procedures which shadow each other), is with IDLWAVE. Why is this superior to anything offered internally in IDL?

For one, it doesn't need IDL! It uses it's own notion of IDL builtins, system library files (the ones in !DIR/lib), and the user catalog it scanned. If IDL is running in the shell, it also queries it for all routine info. Since I've scanned into my catalog almost everything on !PATH (IDLWAVE makes it trivial), I can easily see which files define a procedure. I can even do a full shadow scan of the entire system, or all routines in the current buffer, or all routines compiled in the shell.

The most trivial way to see multiple sources, however, is with routine info (which you'll probably be using for other things all the time anyway). For instance, here's the routine info (invoked via [C-c ?] when near a pro/func), for print, which I've unwisely redefined several times:

```
Usage:    PRINT [, Expr1, ..., Exprn]
Keywords: AM_PM DAYS_OF_WEEK FORMAT MONTHS REWRITE STDIO_NON_FINITE
Sources:  - Builtin
          - Other      [-SB] ~/foo.pro
          - Library    [C--] ~/idl/scrap/print.pro
          - Library    [C--] ~/idl/scrap/printf.pro
```

We see 4 sources, in order of likelihood of usage. Here, the built-in print is always used. You can't override it. Then we have a file foo.pro in a buffer I'm visiting (the B), which has a version of "print" compiled in the shell (S) (will I never learn). Then there's a pair of "Library" files (which just means they're in !PATH), which have been scanned and put into my catalog "C".

What about something you can override? How about one of the !DIR/lib procedures which comes with IDL? Here's an example shadow listing for one of those (notice it's the same as routine info, but without the usage/keyword stuff).

```
ValidateManagedWidgets
   - SystemLib   [C--] /usr/local/rsi/idl/lib/xmanager.pro
   - Library     [C--] ~/idl/scrap/xmanager.pro
```

So, I've redefined ValidateManagedWidgets somewhere, but ~/idl comes after !DIR/lib on the !PATH, so it won't ever be automatically compiled. How cheeky.

You get the idea. You can also find out interesting things about your
path ordering, like:

```
CW_COLOR_INDEXE()
   - Obsolete    [C--] /usr/local/rsi/idl/lib/obsolete/pwidget.pro
   - SystemLib   [C--] /usr/local/rsi/idl/lib/cw_clr_index.pro
```

I.e. this file defines cw_color_indexe in obsolete, and is on the path
before cw_clr_index. Which one gets called depends on which one of
these .pro's gets compiled, but if the first is compiled before the
second, that's a silent routine shadow... watch out.

Oh by the way, middle clicking on any of the .pro's listed above would
take you immediately to the routine definition in the source code, so
you can see for yourself with no fuss why you thought you'd override
print, for instance.

Good luck,

JD

Subject: Re: Which like command for IDL?
Posted by davidf on Fri, 05 Jan 2001 16:06:41 GMT
View Forum Message <> Reply to Message

David Fanning (davidf@dfanning.com) writes:

> It is a moot point anyway, in this case, since the program
> uses some of the neat new SWITCH, BREAK, etc. stuff that
> comes in IDL 5.4, and will not compile in earlier versions.

Interestingly, the FILE_WHICH program supplied in IDL 5.4
calls a built-in, but undocumented, program STRTOK, which
appears to separate the path subdirectories based on
a delimiter supplied to the function. I'll leave it
to the expert sleuths in the group to tell us what it
really does. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Which like command for IDL?
Posted by Pavel A. Romashkin on Fri, 05 Jan 2001 16:57:11 GMT
View Forum Message <> Reply to Message

Oh, STRTOK has been around for a while. Any illegal call to STRSPLIT (my
common one, passing an array as a parameter) would crash it and expose a
call to STRTOK. I think I even asked a question about it (long time
ago), but I can't remember what kind of answer did I get. A real IDL
hacker (like Craig) probably would know.
Cheers,
Pavel

David Fanning wrote:
>
> David Fanning (davidf@dfanning.com) writes:
>
>> It is a moot point anyway, in this case, since the program
>> uses some of the neat new SWITCH, BREAK, etc. stuff that
>> comes in IDL 5.4, and will not compile in earlier versions.
>
> Interestingly, the FILE_WHICH program supplied in IDL 5.4
> calls a built-in, but undocumented, program STRTOK, which
> appears to separate the path subdirectories based on
> a delimiter supplied to the function. I'll leave it
> to the expert sleuths in the group to tell us what it
> *really* does. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting
> Phone: 970-221-0438 E-Mail: davidf@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Which like command for IDL?
Posted by John-David T. Smith on Fri, 05 Jan 2001 19:03:10 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:

>
> Oh, STRTOK has been around for a while. Any illegal call to STRSPLIT (my
> common one, passing an array as a parameter) would crash it and expose a
> call to STRTOK. I think I even asked a question about it (long time
> ago), but I can't remember what kind of answer did I get. A real IDL
> hacker (like Craig) probably would know.
> Cheers,
> Pavel
>
> David Fanning wrote:
>>
>> David Fanning (davidf@dfanning.com) writes:
>>
>>> It is a moot point anyway, in this case, since the program
>>> uses some of the neat new SWITCH, BREAK, etc. stuff that
>>> comes in IDL 5.4, and will not compile in earlier versions.
>>
>> Interestingly, the FILE_WHICH program supplied in IDL 5.4
>> calls a built-in, but undocumented, program STRTOK, which
>> appears to separate the path subdirectories based on
>> a delimiter supplied to the function. I'll leave it
>> to the expert sleuths in the group to tell us what it
>> *really* does. :-)

from /usr/local/rsi/idl/lib/strsplit.pro:

```
; NAME:
;       STRSPLIT
;
; PURPOSE:
;       Wrapper on the build in system routine STRTOK that implements
exactly
;       the same interface as STRTOK, but with the STRSPLIT name.
;
;       The reason for doing this is so that if a user has their own
;       STRSPLIT in their local user library, their version will
superceed
;       this one. RSI does not recommend this practice, but it is
;       allowed for backwards compatability reasons. See the
;       documentation for STRSPLIT in the IDL Reference manual
;       for details on arguments, keywords, and results.
;
```

The lesson for the day has been: built-ins cannot be overridden, unless
RSI sets up a silly hack such as this to allow you to.

JD

Subject: Re: Which like command for IDL?
Posted by Vapuser on Fri, 05 Jan 2001 21:42:04 GMT
View Forum Message <> Reply to Message

Sorry for superseding my post, but I had left a rant in about
routine_info/resolve_routine that I really didn't want to send,
since I'd discovered some information about those two routines that
made my rant a bit too splenetic, if you know what I mean.

Anyway, I do have some problems with those two routines which I'll
indicate below.

davidf@dfanning.com (David Fanning) writes:

> David Fanning (davidf@dfanning.com) writes:
>
>> It is a moot point anyway, in this case, since the program
>> uses some of the neat new SWITCH, BREAK, etc. stuff that
>> comes in IDL 5.4, and will not compile in earlier versions.
>
> Interestingly, the FILE_WHICH program supplied in IDL 5.4
> calls a built-in, but undocumented, program STRTOK, which
> appears to separate the path subdirectories based on
> a delimiter supplied to the function. I'll leave it
> to the expert sleuths in the group to tell us what it
> *really* does. :-)
>

<snip>

If it's like the C function of the same name, it 'tokenizes' the
string using any delimiter which appears in a particular set, which
is input to the function. It's like repeated calls to strsplit with
different delimiters.

So, *IIRC* you could say 'stuff=strtok(path,':/\') and it would
split the string up regardless of whether you were on a Windows of
Unix machine. (I forget what the delimiter is for Vaxen)

By the way, here's my entry into the (pre 5.4) field. It works by
trying it as a system routine first, then it looks in the output
from help,/source for an *exact* match of the input name (stopping
at the first, see my <rant> below), then an object (if it has a ::
in it) then procedure, a function and, if all these fail, it appends
a '__define' on the input name and tries that, just in case someone
just passed the name of the object it.

It will even work if the object method is defined in it's own file, provided one follows the obj__method.pro naming convention.

It has a *whole* slew (well, two actually) of GOTOs which I couldn't
find a way to get rid of, mostly because
resolve_routine/routine_info need to know whether the thing being
resolved/asked-about is a procedure or a function beforhand.

<rant>

After I rewrote this routine to be a bit smarter I came to a better
understanding of the problems associated with
resolve_routine/routine_info. But I still think that the proper way
to do this sort of thing is to ere on the side of accomodating the
user and let them resolve necessary ambiguities rather then
requiring them to do it *before* the call. (of course, in order to
follow my own advice, I'll have to rewrite my `which.pro', which I
am going to do in my copious free time!) If the user askes for
information about two routines with the same name, one a function
and one a procedure, I think routine_info should return information
about *both* along with some way to tell which is which and let the
user decide which he/she wants. Similarly, I wonder why routine_info
doesn't resolve the routine(s) itself, instead of requiring it be
done by the user before hand. If there is ambiguity, *resolve both*
and default to the previous lemma.

If anyone can tell me why this wouldn't be a better way to do it,
please do so but I don't see any *real* reason to do it except that
it's harder to write the code. (and that's only a quasi-real reason ;->)

</rant>

William Daffer

```
;+
; NAME:  Which
; $Id: which.pro,v 1.2 2001/01/05 21:03:04 vapuser Exp $
; PURPOSE:  Like the Unix 'which' program. Tells you which source file
;          a given routine is in.
;
; AUTHOR:  William Daffer
;
; CATEGORY:  Utility
;
; CALLING SEQUENCE: which,'routine'
;
; INPUTS:  routine:  An IDL procedure/function
;
; OPTIONAL INPUTS:  None
;
```

```
; KEYWORD PARAMETERS:  None
;
; OUTPUTS:  Prints one line with the following info
;
;  "routine: System routine" if it's a system routine. -- or --
;  "routine: path" if it finds the routine -- or --
;  "routine: Doesn't exist" if the previous two fail.
;
;
; OPTIONAL OUTPUTS:  none
;
; COMMON BLOCKS:  none
;
; SIDE EFFECTS: The routine is compiled along with any possible
;               routines contained in the object definition, if this
;               circumstance applies.
;
; RESTRICTIONS:
;
; PROCEDURE: Look in the system routines for this name, if not there,
;            look in the output from help,/source, if it isn't there,
;            try various calls to resolve_routine and routine_info.
;            If `routine' has a '::' in it (e.g. foo::bar), `which'
;            will resolve will be foo__define and see if bar is a
;            method defined in that file, otherwise it will assume
;            that the routine is defined in the file `foo__bar.'
;
;            If these no '::' and `routine' doesn't resolve either as
;            a procedure or a function, `which' will attempt to
;            revolve 'routine__define' and see if someone just passed
;            an object name in.
;
;
;
; EXAMPLE:
;
; IDL> which,'foo'
;           foo: /path/to/foo.pro
;
; IDL> which,'foo::init'
;       foo::init: /path/to/foo__define.pro
;
;    if init is defined in foo__define.pro
;
;       -- or --
;
; IDL> which,'foo::init'
;       foo::init: /path/to/foo__init.pro
```

```
;
;       if init is defined in foo__init.pro
;
; IDL> which,'contour'
;      contour: SYSTEM ROUTINE!
;
; IDL> which,'foobar'
;      foobar: DOESN'T EXIST!
;
; MODIFICATION HISTORY:
;
; $Log: which.pro,v $
; Revision 1.2  2001/01/05 21:03:04  vapuser
; Reworked completely
;
; Revision 1.1  1999/10/06 21:54:32  vapuser
; Initial revision
;
;
;Copyright (c) 1999, William Daffer
;-

PRO which, procname
  usg = "Usage: which,`procname' (with `procname' a nonempty STRING)"
  IF n_params() LT 1 OR n_elements(procname) EQ 0 THEN BEGIN
    Message,USG,/cont
    return
  ENDIF

  IF size(procname,/type) NE 7 THEN BEGIN
    Message,usg,/cont
    return
  ENDIF

  tproc =  strupcase(strtrim( procname,2))

  IF strlen(tproc) EQ 0 THEN BEGIN
    Message,usg,/cont
    return
  ENDIF
  savequiet = !quiet
  !quiet = 1
  system_routines = routine_info(/system)

  catch,/cancel
  errcnt = -1
  is_func = 0
  is_obj = 0
    ;; Look in the SYSTEM routines first
  pos = strpos( system_routines, tproc)
  x = where(pos NE -1,nx)
  IF nx NE 0 THEN BEGIN
    found = 0
    ii = 0
    REPEAT BEGIN
    ;; check for possible false positives!
      tmp = strcompress(system_routines[x[ii]])
      tmp = strsplit(tmp,' ',/extract)
      test = tmp[0]
      IF test EQ  tproc THEN found = 1
      ii = ii+1
    ENDREP UNTIL found OR ii GE nx
    IF found THEN BEGIN
      outmsg = procname + ': SYSTEM ROUTINE!'
      print,outmsg
      !quiet = savequiet
      return
    ENDIF
  ENDIF

  ;; Then in the already compiled routines

  help,/source,out=out
  out = strupcase(out)
  pos = strpos(out,tproc)
  x = where(pos NE -1, nx )
  found = 0
  ii = 0

  IF nx NE 0 THEN BEGIN
    REPEAT BEGIN
    ;; check for false positives!
      tmp = strcompress(out[x[ii]])
      tmp = strsplit(tmp,' ',/extract)
      test = tmp[0]
      IF test EQ  tproc THEN found = 1
      ii = ii+1
    ENDREP UNTIL found OR ii GE nx
    IF found THEN BEGIN
      catch, error
      IF error NE 0 THEN BEGIN
        catch,/cancel
        is_func = 1
      ENDIF
      info = routine_info(tproc,/source,FUNC=is_func)
      outmsg = info.path
    ENDIF
  ENDIF

    ;; And finally, try to compile it!

  errcnt = -1
  is_func = 0
  is_obj = 0

  IF NOT found THEN BEGIN

    IF strpos(procname,'::') NE -1 THEN BEGIN

        ;; Damn! object reference!

      tmp = strsplit(tproc,':',/extract)
      procs_to_resolve = [tmp[0] + "__DEFINE", procname]
      message,/reset

      errcnt2 = -1
      is_func2 = 0

      catch, error1
      IF error1 NE 0 THEN BEGIN
        errcnt2 = errcnt2 + 1
        CASE errcnt2 OF
          0: BEGIN
            is_func2 = 1
            message,/reset
          END
          1: GOTO, own_file
        ENDCASE
      ENDIF
      IF errcnt2 LT 0 THEN $
        resolve_routine,procs_to_resolve[0] ; the __define routine, always a proc

      info = routine_info(procname,/source,func=is_func2)

        ;; If we've made it this far, it's defined in the
        ;; tmp[0]__define file, so go to the end

      outmsg = info.path
      GOTO, endit

      OWN_FILE:

      errcnt2 = -1
      is_func2 = 0
      catch,error2
      IF error2 NE 0 THEN BEGIN
        error2 = 0
        errcnt2 = errcnt2 + 1
        CASE errcnt2 OF
          0: BEGIN
            is_func2 = 1
            message,/reset
          END
          1: BEGIN
            print, procname + ": DOESN'T EXIST!"
            return
          END
        ENDCASE
      ENDIF
      resolve_routine,procs_to_resolve[1],is_func=is_func2 ;
      info = routine_info(procname,/source,func=is_func2)
      outmsg = info.path

    ENDIF ELSE BEGIN

      ;; Doesn't have a "::" in it. May still be an object name, though!
      catch, error
      IF error NE 0 THEN BEGIN
        errcnt = errcnt+1
        CASE errcnt OF
          0: BEGIN
            ; won't compile as a procedure,
            ; try as funtion
            is_func = 1
            message,/reset
          END
          1: BEGIN
            is_obj = 1
            is_func = 0
            tproc  =  tproc + "__DEFINE"
            message,/reset
            ;resolve_routine, tproc[jj]
          END
          ELSE : BEGIN
            ;; can't resolve it as either procedure
            ;; function or object.
            ;; Must not exist!
            !quiet = savequiet
            print, procname + ": DOESN'T EXIST!"
            return
          END
        ENDCASE
      ENDIF
      resolve_routine, tproc, is_func= is_func

      info = routine_info(tproc,/source,FUNC=is_func)
      IF !error_state.code NE 0 THEN BEGIN
        !quiet = savequiet
        outmsg = procname + ": DOESN'T EXIST!"
        print, outmsg
        return
      ENDIF
      outmsg = info.path
    ENDELSE
   ENDIF

   ENDIT:
   outmsg =  procname + ': ' + outmsg
   print, outmsg
   !quiet = savequiet
   return

  END
```

--
William Daffer: 818-354-0161: William.Daffer@jpl.nasa.gov

Subject: Re: Which like command for IDL?
Posted by Vapuser on Fri, 05 Jan 2001 22:29:14 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:

> I hate to be a one tune band,

 It's alright. It's a good note!

 [...]

>
> For one, it doesn't need IDL! It uses it's own notion of IDL builtins,
> system library files (the ones in !DIR/lib), and the user catalog it
> scanned. If IDL is running in the shell, it also queries it for all
> routine info. Since I've scanned into my catalog almost everything on
> !PATH (IDLWAVE makes it trivial)

 I just went and got the helpfile and then scanned my installation
 into my catalog.

Man, you and Carsten are *wizards*.

To paraphrase and line from a nutshell book:

"Any sufficiently advanced technology is indistinquishable from well written emacs lisp package!"

whd
--
William Daffer: 818-354-0161: William.Daffer@jpl.nasa.gov
