Subject: Re: How to call fortran subroution?
Posted by web on Mon, 16 Apr 2001 02:35:54 GMT
View Forum Message <> Reply to Message

Thank you very much. I use IDL under windows.

I have generated vecadd.dll from visual fortran. Then I use

result = call_external('f:\fortran\vecadd.dll','f:\fortran\vecadd ', a,
n_elements(a), x, n_elements(x), b)

to call the dll. The error message is as following:

% CALL_EXTERNAL: Error loading sharable executable.
Symbol: f:\fortran\vecadd, File = f:\fortran\vecadd.dll
ERROR_PROC_NOT_FOUND
But I have dll file under f:\fortran. How to do then? May be I have set
wrong parameter.
This is explanation of call_external:

Result = CALL_EXTERNAL(Image, Entry [, P0, ..., PN-1])

Entry:A string containing the name of the symbol in the library which is the
entry point of the routine to be called.

I donot know how to set entry.

Where can I find The IDL External Development Guide?

Jiali

"Liam Gumley" <Liam.Gumley@ssec.wisc.edu> wrote in message
news:9bd2rbcgm1@news.doit.wisc.edu...
> "Craig Markwardt" <craigmnet@cow.physics.wisc.edu> wrote in message
> news:onwv8m6xpf.fsf@cow.physics.wisc.edu...
>> I think Stefano is right, you can't call FORTRAN directly from IDL,
>> especially with the CALL_EXTERNAL method. That method requires your
>> program to conform to a very specific interface for your function
>> parameters (two parameters named argc and argv).
>>
>> I'm looking at the External Development Guide for IDL 5.2. They
>> recommend a "wrapper" function written in C, which in turn calls the
>> FORTRAN. That manual does give an example where the FORTRAN
>> subroutine can be called directly, but the semantics appear to be
>> present only on a few platforms (VMS or IBM AIX), and the subroutine
>> must still conform to the argc and argv convention.
>>

>> Questions:
>> * What example are you looking at?
>> * Does Ronn Kling's book on external linking address FORTRAN?
>
> If your Fortran compiler supports the %VAL and LOC extensions, you can call
> a Fortran subroutine or function from IDL via CALL_EXTERNAL using a Fortran
> wrapper. I developed the routines shown below to vectorize the following
> operation in IDL:
>
> for i = 0L, n_elements(x) - 1L do a[x[i]] = a[x[i]] + b[i]
>
> ---begin vecadd.f---
> c ... This is the wrapper routine called by IDL
> subroutine vecadd(argc, argv)
> integer*4 argc, argv(*), j
> j = loc(argc)
> call vecadd1(%val(argv(1)), %val(argv(2)), %val(argv(3)),
> & %val(argv(4)), %val(argv(5)))
> end
>
> c ... This is the routine which does the work.
> c ... The arguments are defined exactly the same way as in the
> c ... call to CALL_EXTERNAL in vecadd.pro
> subroutine vecadd1(a, na, x, nx, b)
> integer*4 na, nx
> real*4 a(na), b(nx)
> integer*4 x(nx), i
> do i = 1, nx
> a(x(i)) = a(x(i)) + b(i)
> end do
> end
> ---end vecadd.f
>
> ---begin vecadd.pro---
> FUNCTION VECADD, ARRAY, INDEX, VALUE
>
> ;- Check arguments
> if (n_elements(array) eq 0) then $
> message, 'Argument A is undefined'
> if (n_elements(index) eq 0) then $
> message, 'Argument X is undefined'
> if (n_elements(value) eq 0) then $
> message, 'Argument B is undefined'
> if (n_elements(index) ne n_elements(value)) then $
> message, 'Arguments X and B must have the same number of elements'
>

```
>  ;- Create copies of the arguments with correct type
>  if (size(a, /tname) ne 'FLOAT') then begin
>    a = float(array)
>  endif else begin
>    a = array
>  endelse
>  x = ((long(index) > 0L) < (n_elements(a) - 1L)) + 1L
>  b = float(value)
>
>  ;- Call the external routine
>  result = call_external('vecadd.so', 'vecadd_', $
>    a, n_elements(a), x, n_elements(x), b)
>  if (result ne 1) then message, 'Error calling external routine'
>
>  ;- Return result
>  return, a
>
>  END
>  ---end vecadd.pro---
>
>  To compile the Fortran source on SGI IRIX 6.5:
>  % f77 -n32 -KPIC -c vecadd.f
>  % ld -n32 -shared -o vecadd.so vecadd.o
>  (see  /usr/local/rsi/idl_5.3/external/call_external/Fortran/Makefi le for
the
>  syntax on other UNIX platforms).
>
>  To call the routine in IDL 5.3:
>  a = findgen(10)
>  x =[3, 5, 7]
>  b = [2, 4, 6]
>  print, vecadd(a, x, b), format='(10f5.1)'
>    0.0  1.0  2.0  5.0  4.0  9.0  6.0 13.0  8.0  9.0
>
>  The IDL External Development Guide is quite helpful in explaining how this
>  works.
>
>  Cheers,
>  Liam.
>  http://cimss.ssec.wisc.edu/~gumley/
>
>
>
```