Subject: Calling fortran
Posted by hto on Thu, 10 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Does anyone have an example of calling Fortran from IDL?
Howard Onishi

Subject: Re: Calling fortran
Posted by dors on Thu, 10 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Examples of how to use call_external come WITH idl. The are located
in a path something like: /usr/local/rsi/idl_4/external/examples/sharelib/

Also check out the IDL FORTRAN-FAQ:

ftp://eos.crseo.ucsb.edu:/pub/idl/idl-fortran.Z

Note that the call external example given uses a C interface program,
it is also possible to use a FORTRAN interface program.

A very simple shell of a FORTRAN interface program would look like:
(check out the post I made last week about this with a more
complicated example, I erased it, maybe someone else will be kind
enough to repost it. :)

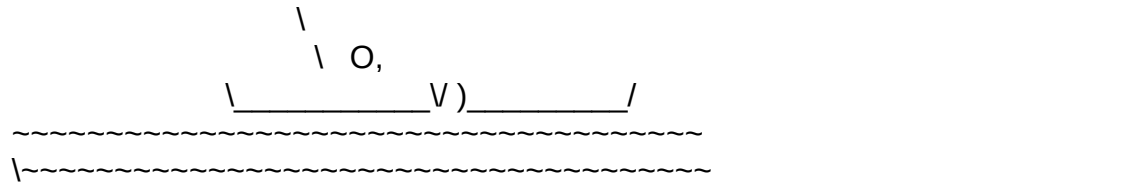
```
c      Interface routine
       SUBROUTINE pass (argc, argv)
c note size of intergers on the next line depends on your system pointer size
       INTEGER*8 argc, argv(*)
       CALL mysub (%VAL(argv(1)),%val(argv(2)))
       RETURN
       END

       SUBROUTINE mysub (a, b)
       integer*4 a, b

       print *, a, b

       end
```

idl:

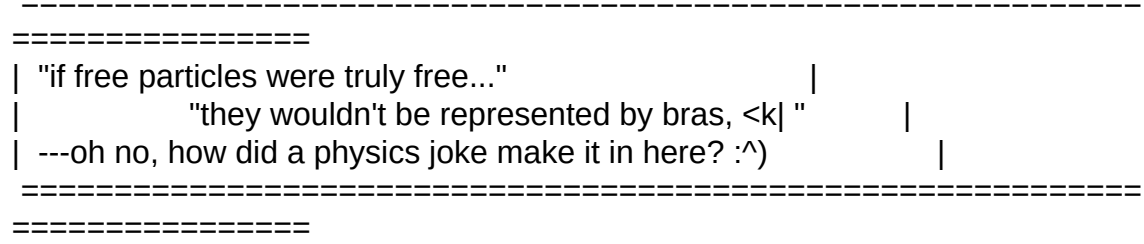
```
j=CALL_EXTERNAL ('mylib.so','pass_',NBLOCKS,len,VALUE=[0,0])
```

Eric

--



Eric E. Dors | Internet: eric.dors@unh.edu
203 Van Allen Hall	
Iowa City, IA 52242	



Subject: Re: Calling fortran
Posted by Andy Loughe on Fri, 11 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

On Thu, 10 Oct 1996, Howard Onishi wrote:

> Does anyone have an example of calling Fortran from IDL?
> Howard Onishi
>

Yep! You will have to chage the link statement to fit your needs.

```
;============================================================ ============
;  Originator:  Andrew F. Loughe
;
;  *** SUN SOLARIS 2.3 TEST ***
;  Procedure to call a FORTRAN program from within IDL.
;  The first 100 primes are computed in the FORTRAN program and
;  passed into an IDL vector called prime_nums.
;  We pass into the subroutine the number of primes desired.
;
;  NOTE:  A large number of "nonsense" function calls are made
;         from within the FORTRAN subroutine in order to test
;         that the link is robust.  It found a problem with
;         atan2 and datan2.

;  **** Set these for yourself ****
   DIR = '/home/afl/ensemble/weights/src/idl/'
```

```
   LIB_DIR = '/usr/lib/'

;  This doesn't quite look like a shared object library!!
;  May not need all three libraries, but for other tests I did need them.
compile = 1
if (compile eq 1) then begin
   SRC = DIR + 'primes.f  '
   OBJ = DIR + 'primes.o  '
   OUT = DIR + 'primes.so '
   LIB = LIB_DIR + 'libV77.a  ' + LIB_DIR + 'libF77.a  ' + $
         LIB_DIR + 'libsunmath.a '

   spawn, 'f77 -c -Kpic ' + SRC
   spawn, 'ld -G -o ' + OUT + OBJ + LIB
endif

num_primes = 100L                ; Want 100 primes.
prime_nums = lonarr(num_primes)    ; Initialize the prime_nums vector.

;  Call a FORTRAN program to do the computation.
a = CALL_EXTERNAL(DIR + 'primes.so', 'primes_', num_primes, prime_nums)

print, prime_nums(*)

end


 C============================================================ ===========
C  Originator:  Andrew F. Loughe
C
C  *** SUN SOLARIS 2.3 TEST ***
C  A rather simple, inefficient, poorly nested, quickly written,
C  subroutine (apology accepted?) to compute the first N primes.
C  It is used to demonstrate the ability of IDL to call a FORTRAN
C  subroutine to accomplish some task, accepting an input paramter,
C  and returning some values.
C  num_primes is passed into this subroutine from IDL.
C
C  NOTE:
C  Some nonsense function calls are added to see if our link is robust.
C  From this test I learned that atan2 and datan2 are symbols which
C  could not be found.

      subroutine primes1(num_primes, prime)

      implicit none

      integer i, j, icount, num_primes
```

```
      integer prime(num_primes)
      real r, r2
      double precision d, d2


      prime(1) = 2          !  By definition 1 is not prime.
      prime(2) = 3
      prime(3) = 5          !  This simple method requires
      prime(4) = 7          !  specification of primes under 10.
      icount   = 4


C  Loop through a large number of integers.
C  Return only "num_primes" primes.
      do 100 i = prime(icount)+2, 1e8, 2

C  Test for an even divisor.
         do 200 j = 3, int( sqrt( float(i) ) ), 2
           if ( mod(i,j) .eq. 0 ) goto 100       ! Number not prime.
 200     continue

C  A prime has been found!
         icount = icount + 1
         prime(icount) = i
         if (icount .gt. num_primes-1) goto 300   ! Only want num_primes

 100  continue

C  SOME NONSENSE FUNCTION CALLS:
C  Sometimes a particular symbol is not found, so the CALL_EXTERNAL
C  routine fails.  Let's do some nonsense function calls to see if
C  our link is robust.  Sorry, not all FORTRAN functions are tested.

 300  i = 100
      r = 100.
      d = 100.

      i = iabs(i)
      r = abs(r)
      d = dabs(d)

      i = max0(i, 2)
      r = amax1(r, 3.)
      d = dmax1(d, d*d)

      i = min0(i, 2)
      r = amin1(r, 3.)
      d = dmin1(d, d*d)
```

```
      r = sqrt(r)
      d = dsqrt(d)

      r = exp(r)
      d = dexp(d)

      r = alog( abs(r) )
      d = dlog( dabs(d) )

      r = alog10( abs(r) )
      d = dlog10( dabs(d) )

      r = sin(r)
      d = dsin(d)

      r = cos(r)
      d = dcos(d)

      i = 100
      r = 100.
      d = 100.
      r2= .5
      d2= .5

      r = tan(r)
      d = dtan(d)

      r = asin(r)
      d = dasin(d)

      r = acos(r)
      d = dacos(d)

      r = atan(r)
      d = datan(d)

C  COULD NOT FIND THESE 2 SYMBOLS
C       r = atan2(r, r2)
C       d = datan2(d, d2)

      r = sinh(r)
      d = dsinh(d)

      r = cosh(r)
      d = dcosh(d)

      r = tanh(r)
```

```
    d = dtanh(d)

    return
    end
```