Subject: Help on calling Fortran routines from IDL under linux
Posted by isoaga2 on Wed, 08 Oct 2003 06:45:35 GMT
View Forum Message <> Reply to Message

For the past few days i have been trying to call basic fortran
routines from IDL (IDL 6.0 and Gentoo Linux on a PC). First i tried
using the CALL_EXTERNAL approach using both fortran and C wrappers as
shown in the IDL documentation however the output i was getting from
the sum_array example was incorrect. Here is what i did first:

```
 ------------------------------------------------------------- ---------
test_call_external.pro - the IDL calling routine
 ------------------------------------------------------------- ---------
pro test_call_external

 x = [1.0,2.0,3.0]
 sum = 0.0
  n = n_elements(x)
 s = call_external('/home/david/PhD/Fortran/sum_array.so',
'sum_array', $
    x, n, sum)
 print, s
 help, s

end

 ------------------------------------------------------------- ---------
sum_array.c - the C wrapper as in the IDL documentation
 ------------------------------------------------------------- ---------
#include <stdio.h>

void sum_array(int argc, void *argv[])
{
 extern void sumd_(); /* Fortran Routine */
 int *n;
 float *s, *f;

 f = (float *) argv[0]; /* Array pntr */
 n = (int *) argv[1]; /* Get # of elements */
 s = (float *) argv[2]; /* Pass back result a parameter */

 sumd_(f, n, s);  /* Compute Sum */
}

 ------------------------------------------------------------- ---------
sumd.f - the Fortran routine as in the documentation
 ------------------------------------------------------------- ---------
      SUBROUTINE sumd(array, n, sum)
```

```
      INTEGER*4 n
      REAL*4 array(n), sum

      sum=0.0
      DO i=1,n
      sum = sum + array(i)
      ENDDO

      RETURN
      END
```

I compiled both the sumd.f and sum_array.c files to objects then linked them together into a shared object (sum_array.so) using gcc (i think, i'm a gcc ultra newbie) as follows:

```
gcc -c sumd.f
gcc -c sum_array.c
gcc -shared -o sum_array.so sumd.o sum_array.o
```

However, when i compile and run test_call_external i get the output:

```
-1073746820
S               LONG      =  -1073746820
```

This, i don't understand, so i figured i'd try the fortran wrapper like this...

```
 ------------------------------------------------------------ ---------
dave.f - the Fortran routine as in the documentation
 ------------------------------------------------------------ ---------
      SUBROUTINE SUM_ARRAY(argc, argv)  !Called by IDL
      INTEGER*4 argc, argv(*)           !Argc and Argv are integers

      j = LOC(argc)         !Obtains the number of arguments (argc)
                    !Because argc is passed by VALUE.

      CALL SUM_ARRAY1(%VAL(argv(1)), %VAL(argv(2)), %VAL(argv(3)))
      RETURN
      END

      SUBROUTINE SUM_ARRAY1(array, n, sum)
      INTEGER*4 n, test
      REAL*4 array(n), sum

      sum=0.0
      DO i=1,n
      sum = sum + array(i)
```

```
      ENDDO
      RETURN
      END
```

I compiled this with gcc as (to get the shared object dave.so)

gcc -w -shared -o dave.so dave.f

and then run the following idl program...

```
 ------------------------------------------------------------ ---------
test_call_external2.pro - idl calling routine
 ------------------------------------------------------------ ---------
pro test_call_external

 x = [1.0,2.0,3.0]
 sum = 0.0
  n = n_elements(x)
 s = call_external('/home/david/PhD/Fortran/dave.so', 'sum_array__', $
    x, n, sum)
 print, s
 help, s

end
```

the entry point has 2 trailing underscores due to the gcc compile
process i think, i used "nm dave.so" to find that out. After running
that idl program i get

```
 -1073746824
S               LONG      =  -1073746824
```

Which is almost identical to the c wrapper approach, anyone have an
idea as to my mistake?

The next thing i did was to try out the dlm approach described by
S.V.H. Haugun on http://www.astro.uio.no/~steinhh/idl/additions.html
using his cool perl script and ftnchek. I figured i test it out with
the example he suggested, ie this fortran routine...

```
 ------------------------------------------------------------ ---------
square.for - fortran rountine
 ------------------------------------------------------------ ---------
      DOUBLE PRECISION FUNCTION SQUARE(X)
      DOUBLE PRECISION X
      SQUARE = X * X
      END
```

Then, trying to follow his instructions for my machine i did:

ftnchek -nocheck -quiet -makedcl square.for
dlmform square.for
gcc -o square.fo -c square.f
gcc -c square.c
gcc -shared -o square.so square.o square.fo
mv square.so /usr/local/rsi/idl/bin/bin.linux.x86/square.so
mv square.dlm /usr/local/rsi/idl/bin/bin.linux.x86/square.dlm

restarted idl and did

dlm_load, 'square'
% Loaded DLM: SQUARE.

but then,
IDL> print, square(4)
% Variable is undefined: SQUARE.
% Execution halted at: $MAIN$
IDL>

i checked the loaded module with

% Execution halted at: $MAIN$
IDL> help, /dlm, name='square'
** SQUARE - Subroutines: SQUARE (loaded)
   Version: <unknown>, Build Date: 8 Oct 2003, Source: Perl script
written by S. V. H. Haugan
   Path: /usr/local/rsi/idl_6.0/bin/bin.linux.x86/square.so
IDL>

So i'm not sure why that is not working either :(

Anyhelp would be great, Thanx.