Subject: Re: Calling fortran under unix
Posted by dors on Wed, 02 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

You can also write your wrapper functions in FORTRAN if you are not into C. I can write FORTRAN and C but when taking a piece of existing FORTRAN code I find it much easier to stay in FORTRAN. And in this case FORTRAN has all the power you need (DEC FORTRAN does anyway). I politely disagree that writing in FORTRAN is a problem. Everyone has their own favorite language. They aren't the real limitation, any task can be done in any language, it is all a matter of efficiency.

Here are some test files for a very simple example, only passing longs and strings. Many more complicated examples could be created, but this should get the main point across. There is much documentation in the IDL manuals on all of this for many architectures. And I remember seeing some example files somewhere in the idl source tree.:

IDL:

```
PRO pass

a = long(10)
b = 'I can pass strings!!'
c = long(9)
print, 'calling FORTRAN'
result = CALL_EXTERNAL ('simple.so','simple_',a,b,c,VALUE=[0,1,0])
print, 'back in IDL'
print, 'result=', result

END
```

FORTRAN:

```
c      Interface routine
       function simple (argc, argv)
       INTEGER*8 argc, argv(*)
       INTEGER*4 temp1
       simple=sum (%VAL(argv(1)),%val(argv(2)),%val(argv(3)))
       return
       END

c      desired function
       function sum (a, b, c)
       integer*4 sum
       integer*4 a,c
       character*20 b

       print*, 'Executing FORTRAN'
```

```
      print*, a
      print*, b
      print*, c
      sum=a+c
      print*, 'Leaving FORTRAN'
      return
      end
```

Here are the statements I used to compile (I know I have some extra libraries here but it is a compile statement I coppied from another project and I don't feel like sorting through what is needed :^)) Also I assume you can do the same with the f90 compiler but I haven't used it yet.:

```
f77 -c -o simple.o simple.for
ld -o simple.so -shared simple.o -lUfor -lfor -lFutil -lm -lots -lc
```

Note 1: Make sure you think about the use of pass by value versus pass by reference when it comes to strings.

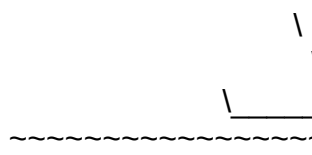
Note 2: Make sure you know the pointer size on your machine, on my DEC Alpha it is integer*8, if this is not true for your machine you will get a segmentation fault unless you make the appropriate changes.

Note 3: IDL will not pass integer*8's as data, there is no internal format for that, nor will it pass real*16's.

Good luck,

Eric

--



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

"if free particles were truly free..."
"they wouldn't be represented by bras, <k| "
---oh no, how did a physics joke make it in here? :^)

==========

Subject: Re: Calling fortran under unix
Posted by korpela on Wed, 02 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

In article <32523466.41C6@mod5.ag.rl.ac.uk>,
Simon Williams <williams@mod5.ag.rl.ac.uk> wrote:
> I gather that the CALL_EXTERNAL interface is rather different
> under Unix, and that I might have to write some sort of
> intermediate "wrapper" between the idl and the fortran.
>
> I would be very grateful if anyone could show me how to do
> it for a simple example:
>
>   subroutine simple(a,b,c)
>
>   integer a,b,c
>
>   c=a+b
>
>   end

You'll have to write a C wrapper that calls your fortran subroutine.
Here's an example that may work on your above example.

```
 ------------------------------------------------------------ --------------
/* here's the prototype for the fortran function */
void simple_(int *a, int *b, int *c);

int wrapper(int argc, void *argv[])
{
  int *a, *b, *c;

  /* check the number of arguments */
  if (argc == 3) {
    a=(int *)argv[0];
    b=(int *)argv[1];
    c=(int *)argv[2];
    simple_(a,b,c);
    return(0);
  } else {
    return(-1);
  }
}
 ------------------------------------------------------------ --------------
```

you'd hace to link the wrapper function and the fortran function into the

same shared library. Under SunOs the commands would be...

```
% gcc -Wall -fpic -c wrapper.c
% f77 -PIC -c simple.f
% ld -o simple.so -assert pure-text simple.o wrapper.o -lF77
```

Under your unix they may be different.

The call_external statement would be:

```
error_code=call_external('simple.so','wrapper',a,b,c)
```

The general rules for linking C and Fortran are

1) Most varieties of fortran append an underscore to function names.
    The fortran function hello() becomes the C function hello_()

2) Fortran subroutine and function parameters are passed as pointers.
    subroutine garbage(a,b,c) INTEGER*4 a, REAL b, INTEGER*2 c
    becomes void garbage_(long *a, float *b, short *c)

3) Watch your types! "integer" is not necessarily "int". In your fortran
    code use integer*2 and integer*4 in the C code use short and long.
    Use real and double precision in fortran. float and double in C.
    None of the above is guaranteed to work, but it's a starting point.

4) Fortran arrays are passed as a pointer to the first element of the array
    subroutine garbage(a,b,c) INTEGER*4 a(100), REAL B(100), INTEGER*2 c(100)
    becomes void garbage_(long *a, float *b, short *c)

5) To avoid all these problems, skip fortran and move straight to C.

Eric

--
Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.

Click here for more info.