Subject: Re: call_external fortran with OS X...what compiler for f95?
Posted by henrygroe on Tue, 04 Nov 2003 18:10:29 GMT
View Forum Message <> Reply to Message

Ok, I've answered my own question... hopefully this might be of use to someone else.

I've decided to use NAGWare F95, available at www.nag.com. They will allow you to download and get a trial license (good for a month or two) for free. After that an academic license is $240 + $15 shipping.

NAGWareF95 does not include %VAL(), so my above example with g77 wouldn't work, but, I've found two different ways of using call_external with NAGWareF95:

1. use a C wrapper instead of a fortran wrapper.
2. compile a f77 wrapper using g77.

A simple example of each of these follows (both do the same few simple operations; NOTE that the f95 subroutine names differ by an underscore):

1. Using a C wrapper:

```
------------------file osx_test6.f95--------------------------
SUBROUTINE osx_test6(n1,n2,nl1,nl2,f1,f2,d1,d2,b1,b2)
use f90_kind
IMPLICIT NONE
INTEGER(int16)::n1,n2
INTEGER(int32)::nl1,nl2
REAL(single)::f1(n1),f2(n1)
REAL(double)::d1(n2),d2(n2)
INTEGER(kind=int8)::b1,b2

!add one to each of b1,b2
b1 = b1 + 1
b2 = b2 + 1
!we will copy f1 to f2 and then place the sum of f2 in f1(1)
f2 = f1
f1(1) = SUM(f2)
!we will copy d1 to d2 and then place the sum of d2 in d1(1)
d2 = d1
d1(1) = SUM(d2)
!set nl2 = nl1
nl2 = nl1
return
end
------------------end of file osx_test6.f95--------------------
--------------------file call_osx_test6.c----------------------
#include <stdio.h>
```

```
void call_osx_test6(int argc, void *argv[])
{
  extern void osx_test6_();     /* FORTRAN routine */
  int *n1,*n2;  /* int is equivalent to IDL's standard integer */
  long int *nl1,*nl2;  /* long int is equivalent to IDL's long integer
*/
  float *f1,*f2;  /*float is equivalent to IDL's float */
  double *d1,*d2; /*double is equivalent to IDL's double */
  unsigned char *b1, *b2; /*unsigned char is equivalent to IDL's byte
*/

  n1 = (int *) argv[0];
  n2 = (int *) argv[1];
  nl1 = (long int *) argv[2];
  nl2 = (long int *) argv[3];
  f1 = (float *) argv[4];
  f2 = (float *) argv[5];
  d1 = (double *) argv[6];
  d2 = (double *) argv[7];
  b1 = (unsigned char *) argv[8];
  b2 = (unsigned char *) argv[9];

  osx_test6_(n1,n2,nl1,nl2,f1,f2,d1,d2,b1,b2);
}
----------end of file call_osx_test6.c----------------------
To compile:
        f95 -PIC -c call_osx_test6.c
        f95 -PIC -c osx_test6.f95
        f95 -bundle -flat_namespace -dynamic -lm -lc  \
                -o osx_test6.so call_osx_test6.o osx_test6.o
(there will be a few warnings)
To run an example in IDL:
        n1=5
        n2=3
        nl1=25
        nl2=33
        f1=findgen(n1)
        f2=fltarr(n1)
        d1=dindgen(n2)
        d2=dblarr(n2)
        b1 = 127B
        b2 = 255B
        print,n1,n2,nl1,nl2
        print,f1
        print,f2
        print,d1
        print,d2
```

```
        print,b1,b2
        t = CALL_EXTERNAL( 'osx_test6.so','call_osx_test6', &
           n1,n2,nl1,nl2,f1,f2,d1,d2,b1,b2)
        print,n1,n2,nl1,nl2
        print,f1
        print,f2
        print,d1
        print,d2
        print,b1,b2
---------END of C wrapper example

2. compile f77 wrapper with g77, f95 code with f95
-------------------file osx_test7.f95--------------------------
SUBROUTINE osx_test7_(n1,n2,nl1,nl2,f1,f2,d1,d2,b1,b2)
use f90_kind
IMPLICIT NONE
INTEGER(int16)::n1,n2
INTEGER(int32)::nl1,nl2
REAL(single)::f1(n1),f2(n1)
REAL(double)::d1(n2),d2(n2)
INTEGER(kind=int8)::b1,b2

!add one to each of b1,b2
b1 = b1 + 1
b2 = b2 + 1
!we will copy f1 to f2 and then place the sum of f2 in f1(1)
f2 = f1
f1(1) = SUM(f2)
!we will copy d1 to d2 and then place the sum of d2 in d1(1)
d2 = d1
d1(1) = SUM(d2)
!set nl2 = nl1
nl2 = nl1
return
end
------------end of file osx_test7.f95--------------------------
-------------------file call_osx_test6.f-----------------------
        FUNCTION call_osx_test7(ARGC, ARGV)
        IMPLICIT NONE

        INTEGER*4             ARGC    !Argument count
        INTEGER*4             ARGV(*) !Vector of pointers to
argments
        INTEGER               ARG_CNT
        INTEGER*4        call_osx_test7

C       The argument count is passed in by value. Get the location of
C       this value in memory (a pointer) and convert it into an
```

```
C       Fortran integer.

        ARG_CNT = LOC(ARGC)

C  Insure that we got the correct number of arguments
        IF(ARG_CNT .ne. 10)THEN
          WRITE(*,*)': Incorrect number of arguments'
           call_osx_test7 = -1
          RETURN
        ENDIF

C       To convert the pointers to the IDL variables contained in ARGV
C       we must use the Fortran function %VAL. This funcion is used
C       in the argument list of a Fortran sub-program. Call the
Fortran
C       subroutine that will actually perform the desired operations.
C       Set the return value to the value of this function.

        CALL osx_test7( %val(ARGV(1)), %val(ARGV(2)), %val(ARGV(3)),
     &   %val(ARGV(4)), %val(ARGV(5)), %val(ARGV(6)), %val(ARGV(7)),
     &   %val(ARGV(8)), %val(ARGV(9)), %val(ARGV(10)) )
C       Thats all, return to IDL.

        call_osx_test7 = 1
        RETURN
        END
--------------end of file call_osx_test6.f----------------------
to compile:
        g77 -fPIC -c call_osx_test7.f
        f95 -PIC -c osx_test7.f95
        g77 -bundle -flat_namespace -dynamic -lm -lc -undefined
suppress \
                        -o osx_test7.so osx_test7.o
call_osx_test7.o
To run an example in IDL:
        n1=5
        n2=3
        nl1=25
        nl2=33
        f1=findgen(n1)
        f2=fltarr(n1)
        d1=dindgen(n2)
        d2=dblarr(n2)
        b1 = 127B
        b2 = 255B
        print,n1,n2,nl1,nl2
        print,f1
        print,f2
```

```
        print,d1
        print,d2
        print,b1,b2
        t = CALL_EXTERNAL( 'osx_test6.so','call_osx_test6', &
           n1,n2,nl1,nl2,f1,f2,d1,d2,b1,b2)
        print,n1,n2,nl1,nl2
        print,f1
        print,f2
        print,d1
        print,d2
        print,b1,b2
---------END of C g77/f95 example
```