
Subject: Calling FORTRAN routines from IDL
Posted by [cruz](#) on Tue, 02 Feb 1993 00:33:22 GMT

[View Forum Message](#) <> [Reply to Message](#)

I'm a new user of IDL, so I'm soliciting HELP!!! I'm attempting to call FORTRAN routines from IDL using the 'call_external' function. In the call, I'm passing a file name to the routine. Basically I have a FORTRAN front-end to the FORTRAN routine. The FORTRAN front-end passes the parameters by 'value' not reference. The IDL commands follow. Note that 'hdr.txt' is my file name.

```
IDL> a=bytarr(1)
IDL> a(0) = 1
IDL> tmp=call_external('readhdr.so', '_readhdr_', 'hdr.txt', value=a)
....then I receive garbage.
```

I've also tried sending the file name by reference and received the same type of results.

Has anyone experienced this type of problem?? Is there a solution or easier method for accessing FORTRAN routines from IDL?

Thanks for all responses,
Jennifer Cruz
Jet Propulsion Labs
Pasadena, CA
(818)354-0387
cruz@ipanema.jpl.nasa.gov

Subject: Re: Calling Fortran routines from IDL
Posted by [steinhh](#) on Sat, 24 Apr 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Sang-Heon Shim wrote:

> I heard there is a method to call fortran subroutines (f77,
> 90, or both?) from IDL professional version. If that is
> true, it will enable me to link my fortran routines and
> other great fortran routines out there to IDL, which is
> great to me!

Yes, it's true (for most platforms). A "wrapper" routine written in C is necessary -- but anything that can be called from C can be called from IDL, in principle.

For f77 programs, there's now a pretty good, semi-automatic way of generating the C wrapper function: Look at

<http://www.uio.no/~steinhh/idl/dlmform.html>

One big area of trouble is *strings* - I don't know enough on how Fortran XX handles strings, and it's a troublesome area in general, so beware. But who writes Fortran subroutines to manipulate strings...???

Also see <http://www.uio.no/~steinhh/idl/additions.html> for a brief discussion on Dynamically Loadable Modules in general, and some C examples.

Regards,

Stein Vidar

Subject: Re: Calling Fortran routines from IDL
Posted by [Ole Bossing Christens](#) on Tue, 27 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <7fspsb\$30t\$1@readme.uio.no>,
steinhh@ulrik.uio.no (Stein Vidar Hagfors Haugan) wrote:

>
> Sang-Heon Shim wrote:
>
>> I heard there is a method to call fortran subroutines (77,
>> 90, or both?) from IDL professional version.

> Yes, it's true (for most platforms). A "wrapper" routine
> written in C is necessary --

Hi everybody,

Actually, it is not necessary with a C-routine. I attach two files, an IDL function `read_grib.pro` and a corresponding fortran file `read_grib.f`. This interface has been written by Uffe Andersen here at DMI, inspired by an old example from IDL 4 called something like `idl_4/external/examples/sharelib/ftn_only_sun.f` (so, I guess we cannot be the only ones around doing it this way).

We are running IDL 5.2 on SGI IRIX 6.3, and the routines that are part of our GRIB read setup work fine.

Greetings, Ole BC & Uffe Andersen

--

Climate Research Div. Phone : (+45) 39 15 74 26

```
*****
*
* PURPOSE:    Reading grib files directly from IDL
*
*****
*
* LANGUAGE:   Fortran 77
*
*****
*
* PROGRAMMED BY: Uffe Andersen (ua@dmi.dk), 1/4-1998.
* MODIFICATIONS BY: Ole B Christensen (bossing@dmi.dk) 5/10-1998
*
* Danish Meteorological Institute
*
*****
*
* INTERFACE:
*
*   Compiled by:
*       f77 -o32 -c read_grib.f
*
*   Linked by:
*       ld -o32 -shared read_grib.o -L /home/climate/libo32 -lemos -o
libread_grib.so
*
*   Called from IDL by:
*
*       output_arr = FLTARR( nmb_of_data_per_record , nmb_of_records )
*
*       tmp = CALL_EXTERNAL('read_grib.so',      $
*           'read_grib_' ,      $
*           file_name,          $
*           grib_code,          $
*           level,              $
*           lvl_type,           $
*           year,               $
*           month,              $
*           day,                $
*           time,               $
*           grid_par,           $
*           grid_def,           $
*           err_msg,            $
*           nrec,               $
*           ndata,              $
```

```

*           output_arr      )
*
*
*
*
* INPUT:  file_name      : name of the GRIB file
*         grib_code      : the GRIB code of the parameter
*                        to read. If -1 read all parameters
*         level          : the level (pressure or model) to
*                        read. If -1 read first occurrence
*         lvl_type       : the level type (pressure or model) to
*                        read. If -1 read first occurrence
*         year           : the year to read
*         month          : the month to read
*         day            : the day to read
*         time           : the time of day to read
*         nrec           : numbers of records to be read
*
*
* OUTPUT: grid_par       : parameters describin the grid,
*                        identical to the KSEC2 variable
*                        in the GRIB decoding.
*         grid_def       : parameters defining the grid,
*                        identical to thePKSEC2 variable
*                        in the GRIB decoding.
*         err_msg        : error message.
*                        -1 if no record matching the
*                        specifications are found
*                        -3 if the data array is too
*                        small to contains the data
*         nrec           : numbers of records read
*         ndata          : numbers of data per record
*         output_arr     : the data array
*
*
*
*****
*
* REFERENCE:
*
* The IDL-fortran interface program is mainly taken from:
*
* pack/idl_4/external/examples/sharelib/ftn_only_sun.f
*
*****

```

C This is the interface subroutine, called by IDL
C -----

```
SUBROUTINE read_grib(argc, argv)
INTEGER*4 argc, argv(*)
```

C To convert the pointers to the IDL variables contained in ARGV
C we must use the Fortran function %VAL. This function 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.

```
j= LOC(argc)
```

```
CALL read_grib1 ( %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)),
&                %VAL(argv(11)),
&                %VAL(argv(12)),
&                %VAL(argv(13)),
&                %VAL(argv(14)) )
```

```
RETURN
END
```

C This is the fortran subroutine
C -----

```
SUBROUTINE read_grib1 ( idl_str,
&                      code,
&                      level,
&                      lvl_type,
&                      year,
&                      month,
&                      day,
&                      time,
```

```

&          isec2,
&          zsec2,
&          err_msg,
&          nrec,
&          ndata,
&          read_arr )

```

C ----- Declarations -----

```

INTEGER    char_size
PARAMETER  ( char_size = 100 )

```

```

CHARACTER*(char_size) filename

```

```

INTEGER*4  code
INTEGER*4  level
INTEGER*4  lvl_type
INTEGER*4  year
INTEGER*4  month
INTEGER*4  day
INTEGER*4  time
INTEGER*4  err_msg
INTEGER*4  nrec
INTEGER*4  ndata
INTEGER*4  i, nr

```

```

DIMENSION  grid(384)
INTEGER*4  grid

```

```

PARAMETER  (jpack=100000)
DIMENSION  read_arr( jpack * 200 )
DIMENSION  isec0(2)
DIMENSION  isec1(60)
DIMENSION  isec2(384)
DIMENSION  isec3(2)
DIMENSION  isec4(42)
DIMENSION  zsec2(96)
DIMENSION  zsec3(2)
DIMENSION  zsec4(jpack*4)
DIMENSION  inbuff(jpack)
CHARACTER*1 yoper
INTEGER    file, jcount

```

C Declare an IDL string structure

```
STRUCTURE /STRING/  
  INTEGER*2 slen  
  INTEGER*2 stype  
  INTEGER  s  
END STRUCTURE
```

```
RECORD /STRING/ idl_str
```

C ----- Convert IDL-string to FORTRAN-string -----

```
CALL IDL_2_FORT(%VAL(idl_str.s), idl_str.slen, filename, char_size)
```

```
WRITE(6,*) '**'  
  WRITE(6,*) 'Reading Data from : ',filename(1:idl_str.slen)
```

```
  err_msg = 0  
  NUMERR = 0  
  ILENB = jpack  
  IPUNP = jpack * 4
```

C ----- Open file -----

```
CALL PBOPEN( file, filename(1:idl_str.slen), 'R', kret)
```

```
IF ( KRET .NE. 0 ) THEN  
  WRITE(6, *) ' Return code from PBOPEN = ',kret  
  CALL PBCLOSE(FILE, kret)  
  STOP 'Fault in PBOPEN'  
ENDIF  
ILENB = ILENB*4
```

C ----- Header for screen output -----

```
WRITE(6,*)  
& ' Rec no GRIB code  Level',  
& ' Type  Year  Month  Day  Time'
```

```

WRITE(6,*)
& '===== ='

C ----- Record reading loop -----

      nr = 0
100  CONTINUE

C ----- Read data from GRIB file -----

      CALL PBGRIB( file, inbuff, ilenb, lenout, kret )

      IF ( KRET .LT. 0) GOTO 200

C ----- Decode read values -----

      yoper = 'D'
      ierr = 1
      CALL GRIBEX (isec0,isec1,isec2,zsec2,isec3,zsec3,isec4,
&      zsec4,ipunp,inbuff,ilenb,iword,yoper,ierr)
      IF (ierr.NE. 0) then
WRITE(6,*) ' *** GRIBEX - ERR.'
CALL PBCLOSE(file, kret)
STOP
      ENDIF

      ndata = isec4(1)

C --- Check if specified parameters are correct ---
C --- Grib code, level, year, month, day, time ---

      IF ( (code .EQ. isec1(6) .OR. code .EQ. -1) .AND.
& (level .EQ. isec1(8) .OR. level .EQ. -1) .AND.
& (lvl_type .EQ. isec1(7) .OR. lvl_type .EQ. -1) .AND.
& (year .EQ. isec1(10) .OR. year .EQ. -1) .AND.
& (month .EQ. isec1(11) .OR. month .EQ. -1) .AND.
& (day .EQ. isec1(12) .OR. day .EQ. -1) .AND.
& (time .EQ. isec1(13) .OR. time .EQ. -1) ) THEN

```



```
nr = nr + 1
```

```
C    --- This is output to the screen. It was impossible to ---  
C    --- have a longint in a formatted write out (?). The ---  
C    --- present formatted output might behave odd.      ---
```

```
    inr = nr  
    WRITE(6, '(1x,i6,4x,i6,2x,i8,4x,i3,5x,4(i2,4x))' )  
& inr,  
& isec1(6),isec1(8), isec1(7), isec1(10),  
& isec1(11),isec1(12), isec1(13)
```

```
    DO i=1, ndata  
        read_arr(i + ndata*(nr-1)) = zsec4(i)  
    ENDDO
```

```
ENDIF
```

```
C    --- If nrec records are read, jump out of loop ---
```

```
    IF (nr .EQ. nrec .AND. nrec .NE. 0 ) GOTO 200
```

```
    GOTO 100
```

```
C    -- End of record reading loop
```

```
200 CONTINUE
```

```
    IF (nr .EQ. 0) err_msg = KRET
```

```
    nrec = nr
```

```
C    ----- Close file -----
```

```
    CALL PBCLOSE(file, kret)
```

```
RETURN
END
```

```
C=====
=====
```

```
C$Subroutine IDL_2_FORT
```

```
SUBROUTINE IDL_2_FORT(idlstr, strlen, fortstr, f_len)
```

```
C PURPOSE:
```

```
C    Copies an IDL string to a Fortran character string.
```

```
INTEGER*2    strlen
```

```
CHARACTER*(*) idlstr
```

```
CHARACTER*(*) fortstr
```

```
INTEGER      f_len
```

```
C    If the IDL string is smaller then copy the entire string into
C    the Fortran string, otherwise truncate it.
```

```
IF(strlen .LE. f_len) THEN
    fortstr(1:strlen)=idlstr(1:strlen)
ELSE
    fortstr(1:f_len)=idlstr(1:f_len)
ENDIF
```

```
RETURN
END
```

```
-----
;+
;=====
=====
;          READ_GRIB
;=====
=====
;
; PURPOSE :
; -----
;   Read data from a grib file
;
; CALLING SEQUENCE:
```

```

; -----
; field = read_grib(filename)
;
; INPUTS:
; -----
; filename : File name (including the path) for the file to be read
;
; OUTPUTS:
; -----
; Return value: Matrix containing the read values. The dimension
;               of the array is: ( ndata, nrec )
;
; KEYWORD PARAMETERS:
; -----
;
; INPUT:
;
; code      : The GRIB code (integer) [code tables are here :
;               http://intranet.dmi.min.dk/~climate/ERA/table2-v128.html
;               http://intranet.dmi.min.dk/~climate/ERA/table2-v160.html ]
; level     : Level of the data (integer)
; lvl_type  : level type (integer or string)
;               100 or 'pres' = pressure levels
;               105 or 'surf' = surface levels
;               109 or 'model' = model levels
; year      : year (integer)
; month     : month (integer)
; day       : day (integer)
; time      : time (integer)
; nrec      : number of records to be read. Default is 1.
;               Set to 0 if all records that meets the keyword
;               specifications should be read.
; diag      : Flag for diagnostic printout. If this is set various
;               information, incl. month, day and grib-code for the
;               read data, are written to screen.
;
;
; OUTPUT:
;
; grid_par  : Array of integers containing the parameters
;               that describes the grid. Identical to the KSEC2 integer
;               array in the GRIB decoding convention.
; grid_def  : Array of reals that defines the grid. Identical to the
;               PSEC2 array.
; nrec      : number of records read.
;
;
;

```

```

;
; NOTES:
; -----
; The reading is done in a fortran program (read_grib.f).
;
; 'path' should be included in the LD_LIBRARY_PATH in the
; .cshrc file.
;
;     The longitudes and latitudes can be constructed using the
;     routine MAKE_GRID.
;
;
; EXAMPLE:
; -----
;     To read MSLP for 4th of February from file "/mydir/TTfile", type:
;
;     pres = READ_GRIB('/mydir/TTfile', CODE=151, MONTH=2, $
;                   DAY=4, GRID_PAR=gg )
;
;     All records that match the specifications are read. The
;     parameters that describes the grid are returned in the
;     integer array "gg".
;
;
; AUTHOR:
; -----
;     Uffe Andersen,  Danish Meteorological Institute (ua@dmi.dk) 26/10 -
1998
;
;
; REVISIONS:
; -----
;     19/4-99 : Added grid_def, default is to return only one record.  UA
;
;=====
=====
;-

```

FUNCTION read_grib, filename, \$; Input

```

    code=code, $
    level=level, $
    lvl_type=lvl_type, $
    year=year, $
    month=month, $
    day=day, $
    time=time, $
    nrec=nrec, $
    diag=diag, $

```

```
grid_par=grid_par,    $ ; Output
grid_def=grid_def     ; Output
```

```
IF N_PARAMS() EQ 0 THEN MESSAGE, 'File name missing.'
```

```
path = '/home/climate/idl/fk/lib/' ; Path to the fortran program ; that
maintains the communication ; between fortran and idl
```

```
source_file = 'read_grib'           ; Fortran program name
```

```
shared_file = path + 'lib' + source_file + '.so' ; Name of shared object
entry_name =      source_file + '_' ; Entry name in the shared
object
```

```
      ; Set non used keywords to -1
      ; -----
```

```
IF N_ELEMENTS(code) EQ 0 THEN code = -1 IF N_ELEMENTS(level) EQ 0 THEN
level = -1 IF N_ELEMENTS(lvl_type) EQ 0 THEN lvl_type = -1 IF
N_ELEMENTS(year) EQ 0 THEN year = -1 IF N_ELEMENTS(month) EQ 0 THEN
month = -1 IF N_ELEMENTS(day) EQ 0 THEN day = -1 IF N_ELEMENTS(time) EQ
0 THEN time = -1 IF N_ELEMENTS(nrec) EQ 0 THEN nrec = 1 ; Default
value for nrec is 1
```

```
diag = KEYWORD_SET(diag)
```

```
      ; If string convert lvl_type to code
      ; -----
```

```
IF SIZE(lvl_type, /TYPE) EQ 7 THEN BEGIN
  lvl_type = STRLOWCASE(lvl_type)
```

```
  CASE lvl_type OF
    'pres' : type_code = 100
    'surf' : type_code = 105
    'model' : type_code = 109
    ELSE   : MESSAGE, 'Level type ' + lvl_type + ' not known.'
  ENDCASE
```

```
    lvl_type = type_code  
ENDIF
```

```
    ; Year should be the year of the century  
    ; -----
```

```
IF year GE 100 THEN year = year - FLOOR(year/100)*100
```

```
IF nrec LT 0 THEN MESSAGE, 'nrec should be positive or zero'
```

```
err_msg = 1
```

```
    ; In the communication with the FORTRAN program  
    ; it is essential that the integer parameters  
    ; in the IDL and FORTRAN programs are of same  
    ; type (INT/LONG - INTEGER*2/INTEGER*4).  
    ; To facilitate communication only LONG-INTEGER*4  
    ; are used.
```

```
code    = LONG(code)  
level   = LONG(level)  
lvl_type = LONG(lvl_type)  
year    = LONG(year)  
month   = LONG(month)  
day     = LONG(day)  
time    = LONG(time)  
ndata   = LONG(0)  
nrec    = LONG(nrec)  
grid_par = LONARR(384)  
grid_def = FLTARR(96)
```

```
file = FINDFILE(filename, COUNT=cnt) ; Check if the data file exist IF cnt  
NE 1 THEN $ ; If not then stop MESSAGE, 'Could not find ' + filename
```

```
    ; Look for shared object. If it doesn't exist  
    ; compile the fortran program
```

```
file = FINDFILE(shared_file, COUNT=c_so) ; Find shared object file
```

```
IF (c_so EQ 1 ) AND diag THEN PRINT,'Shared object found. '
```

```
IF (c_so EQ 0 ) THEN $  
  MESSAGE, ' Shared object not found.'
```

```
read_arr = FLTARR( 51200L*120L )          ; Define read_arr
```

```
      ; Call the fortran subroutine in the shared object  
      ; -----
```

```
tmp = CALL_EXTERNAL( shared_file, $ ; INPUT: name of shared object  
entry_name, $ ; entry name in shared obj. filename, $ ;  
filename code, $ ; GRIB code level, $ ; level lvl_type, $ ; level  
type year, $ ; year month, $ ; month day, $ ; day time, $ ;  
time grid_par, $ ; OUTPUT: parameters descr. the grid grid_def, $ ;  
parameters defines the grid err_msg, $ ; error message from GRIB nrec,  
$ ; number of records ndata, $ ; numbers of data per record  
  read_arr ) ; output array
```

```
IF err_msg EQ -1 THEN BEGIN  
  MESSAGE, 'No record with the specifications : ', /CONTINUE  
  IF (code NE -1) THEN PRINT, ' Code   : ',code  
  IF (level NE -1) THEN PRINT, ' Level   : ',level  
  IF (lvl_type NE -1) THEN PRINT, ' Level type: ',lvl_type  
  IF (year NE -1) THEN PRINT, ' Year     : ',year  
  IF (month NE -1) THEN PRINT, ' Month    : ',month  
  IF (day NE -1) THEN PRINT, ' Day       : ',day  
  IF (time NE -1) THEN PRINT, ' Time      : ',time  
  PRINT, 'were found in the file : ' + filename  
  read_arr = -1  
ENDIF
```

```
IF err_msg EQ -3 THEN $  
  MESSAGE, ' Data array too small for the record '
```

```
IF diag THEN BEGIN  
  PRINT,'ndata ',ndata  
  PRINT,'nrec ',nrec  
  PRINT,'err_msg ',err_msg
```

ENDIF

```
    ; Find the number of elements in the array  
    ; -----
```

```
IF N_ELEMENTS(grid_par) GT 22 THEN BEGIN non_zero = WHERE(grid_par GT 0,  
cnt) ; addresses of positive numbers last_adr = MAX(non_zero) > 21 ; above  
21 grid_par = grid_par[0: last_adr] ; extract sub-arr ENDIF
```

```
grid_def = grid_def[0: 10+grid_par[11]]
```

```
read_arr = read_arr(0:ndata*nrec-1)
```

```
read_arr = REFORM( read_arr,ndata, nrec )
```

```
RETURN, read_arr
```

END

```
-----== Posted via Deja News, The Discussion Network ==-----  
http://www.dejanews.com/ Search, Read, Discuss, or Start Your Own
```
