Subject: Re: CALL_EXTERNAL
Posted by jacobsen on Fri, 20 Aug 1993 13:12:56 GMT
View Forum Message <> Reply to Message

Here's a document on how I write C code so that I can use the same
".pro" and ".c" files on VAX, AIX (and probably other Unix), and
MS-DOS/Windows 3.1 with Borland C++ 1.5:

This is idl_external.txt, August 20, 1993

Chris Jacobsen
Department of Physics
SUNY Stony Brook, Stony Brook NY 11794-3800
jacobsen@xray1.physics.sunysb.edu

xray1: /usr/local/x1_lib/notes
bnlls1: PUBLIC_DISK:[X1_LIB.IDL]

This is a cookbook example for making C functions and subroutines work
with IDL on AIX, VMS, and MS-DOS.

Note that on Unix and MS-DOS, CALL_EXTERNAL uses argc, argv lists, while on
VMS, it requires all the parameters to be spelled out. That's
the reason behind the cookbook approach described below.

Let's have a function which gets a 2D array and adds up the
contents of the array elements, and which also gets a string which
will simply be echoed. Here's the IDL procedure:

```
;+
; pro testext,array,total,the_string
;
; This just tests out IDL call_external stuff.  Puts the
; total of "array" into "total".  Takes float(array) inside,
; though it does not actually alter "array".  Also prints
; out "the_string". Note that on Windows, the printout will be
; on the screen background...
;-

pro testext,array,total,the_string

total=0.

; Check out the array, and set nx, ny
```

```
svec=size(array)
if (svec(0) eq 0) then begin
 ; If a scalar, make it a 1x1 array
 nx=long(1)
 ny=long(1)
 float_array=fltarr(1,1)+float(array)
endif else if (svec(0) eq 1) then begin
 ; Make it a Nx1 array
 nx=long(svec(1))
 ny=long(1)
 float_array=reform(float(array),nx,ny)
endif else if (svec(0) eq 2) then begin
 nx=long(svec(1))
 ny=long(svec(2))
 float_array=float(array)
endif else begin
 message,'array dimensions must be <= 2D'
 return
endelse

; Check out the string.  Want it to be a scalar string
svec=size(the_string)
if ( (svec(0) ne 0) or (svec(1) ne 7) ) then begin
 message,'argument "the_string" is not a scalar string'
 return
endif

; Now we're ready for CALL_EXTERNAL, because we know that the
; array is floating point 2D, and we have the array dimensions
; as long integers (32 bits), and we know we have a scalar string.
dummy_int=long(0)
progname='testext'
if (!version.os eq 'windows') then begin
; Windows 3.1
  object_file='\idl\lib\local\bin\'+strmid(progname,0,8)+'.dll '
 dummy_int=call_external(object_file,progname, $
  float_array,nx,ny,total,the_string )
endif else if (!version.os eq 'vms') then begin
; VAX/VMS
 object_file='cj$disk:[jacobsen.idl]'+progname+'.exe'
 dummy_int=call_external(progname,progname, $
  float_array,nx,ny,total,the_string, $
  default=object_file)
endif else if (!version.os eq 'AIX') then begin
; Unix
 object_file='/home/jacobsen/idl/'+progname+'.so'
 dummy_int=call_external(object_file,+progname, $
  float_array,nx,ny,total,the_string )
```

```
endif else begin
 print,'Unknown !version.os'
endelse

return
end

 ********************************************************** **********
 ********************************************************** **********
 ********************************************************** **********

/* This is testext.c, which tests out the call_external IDL stuff.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/********************************** First includes, defs ***************/
/************* Windows 3.1, Borland C 1.5 **********/
#if defined(__DLL__)
#include <windows.h>
#define INT32  long int
#define IDL_SHORT short int FAR
#define IDL_USHORT unsigned short int FAR
#define IDL_INT  long int FAR
#define IDL_FLOAT float FAR
/************* VAX/VMS, DEC C compiler ************/
#elif defined(VAXC)
#define INT32  int
#define IDL_SHORT short int
#define IDL_USHORT unsigned short int
#define IDL_INT  int
#define IDL_FLOAT float
/************* Probably most Unix compilers? Only tested on AIX *******/
#else
#define INT32  int
#define IDL_SHORT short int
#define IDL_USHORT unsigned short int
#define IDL_INT  int
#define IDL_FLOAT float
#endif

struct idl_string_struct {
  IDL_USHORT slen;
  IDL_USHORT stype;
  char *s;
```

```
 };

/************************** Then prototype the funcions  ***************/
/************* Windows 3.1, Borland C 1.5 **********/
#if defined(__DLL__)
int FAR PASCAL LibMain( HANDLE hInstance, WORD wDataSeg, WORD wHeapSize,
 LPSTR lpszCmdLine)
{
 if (wHeapSize > 0)
 {
  UnlockData(0);
 }
 return( 1 );
}
long FAR PASCAL _export  testext( IDL_INT argc, LPVOID *argv[])
/************* VAX/VMS, DEC C compiler ************/
#elif defined(VAXC)
int testext( IDL_FLOAT *array, IDL_INT *ptr_nx, IDL_INT *ptr_ny,
 IDL_FLOAT *ptr_total, struct idl_string_struct *ptr_idl_string )
/************* Probably most Unix compilers? Only tested on AIX *******/
#else
int testext( IDL_INT argc, void *argv[] )
#endif

#define MAX_STRING_LENGTH 1024
{
 int char_index, last_char_index;
 char string[MAX_STRING_LENGTH];
 int x_index, y_index;
 float this_value;

#ifndef VAXC
 IDL_FLOAT *array;
 IDL_INT *ptr_nx;
 IDL_INT *ptr_ny;
 IDL_FLOAT *ptr_total;
 struct idl_string_struct *ptr_idl_string;

 if (argc != 5) {
  fprintf( stderr, "testext: argc was %d rather than 5\n",
   argc );
 }

 array = (IDL_FLOAT *)argv[0];
 ptr_nx = (IDL_INT *)argv[1];
 ptr_ny = (IDL_INT *)argv[2];
 ptr_total = (IDL_FLOAT *)argv[3];
 ptr_idl_string = (struct idl_string_struct *)argv[4];
```

```
#endif

 /* First deal with the array */
 *ptr_total = 0.0;
 for ( y_index = 0; y_index < (*ptr_ny); y_index++ ) {
  for ( x_index = 0; x_index < (*ptr_nx); x_index++ ) {
   this_value = *(array + x_index + y_index * (*ptr_nx));
   *ptr_total = *ptr_total + this_value;
  }
 }

 /* Then deal with the string */
 last_char_index = ptr_idl_string->slen;
 if (last_char_index > (MAX_STRING_LENGTH - 2))
 {
  last_char_index = MAX_STRING_LENGTH - 2;
 }
 for ( char_index = 0; char_index < last_char_index; char_index++ )
 {
  string[char_index] = *(ptr_idl_string->s +
   char_index);
 }
 string[char_index] = '\000';

 /* Now we have a regular simple old C language string */
 fprintf( stderr, "String was \"%s\"\n", string );

 return(1);
}

 ********************************************************** **********
 ********************************************************** **********
 ********************************************************** **********

# Makefile for testext for AIX
# A limitation of AIX is that somehow IDL call_external finds only
# the first entry in a library, so need to make a separate .so for
# each program to be called from IDL. Ignore the messages:
# 0706-224 WARNING:  Duplicate symbol '.finite'.
# 0706-224 WARNING:  Duplicate symbol '.logb'.
# 0706-224 WARNING:  Duplicate symbol '.scalb'.

testext.o: testext.c
 cc -O -c testext.c

testext: testext.o testext.c
 ld -o testext.so testext.o -e testext \
  -bM:SRE -T512 -H512 -lc -lm
```

```
 ********************************************************** **********
 ********************************************************** **********
 ********************************************************** **********

$ ! This is make_testext.com for VAX/VMS
$ cc testext
$ link testext, sys$input/opt/share
universal = testext
$

 ********************************************************** **********
 ********************************************************** **********
 ********************************************************** **********

# This is a Makefile which will work for compiling
# code with Borland C 1.5 to work with IDL on Windows 3.1
# It puts all compiled stuff in a directory \idl\lib\local\bin

testext:    testext.c
 bcc -WD -ml! -3 -Lc:\borlandc\lib testext.c
 rc testext.dll
 copy testext.dll \idl\lib\local\bin

 ********************************************************** **********
 ********************************************************** **********
 ********************************************************** **********
```


--
****
Chris Jacobsen, Department of Physics, SUNY at Stony Brook
Phone (516) 632-8093, FAX -8101   Bitnet: cjacobsen@sbccmail
jacobsen@xray1.physics.sunysb.edu ALL-IN_ONE: CJACOBSEN
****