Subject: Updated C code to call IDL or PV-WAVE
Posted by grunes on Sat, 09 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Here is an updated attempt to post code to call IDL or
PV-WAVE from C in a simple minded fashion, formatted as
an attachment to avoid line wrapping:

Here is some C code to call IDL or PV-WAVE.

This is a portable interface between C and IDL (or PV-WAVE).
With it IDL or PV-WAVE can be easily used by C programs as a way to plot
data, show images, or do other types of data processing. Data transfer is
supported in both directions.

(If long line wraps garble the text, try turning off your news
reader's line wrapping. If it is still wrapped, e-mail me.)

See the comments in the source code, and the sample main program in that
source code for more information!

There are two files, idl_from_c.c (containing the source code) and
idl_from_c.h (containing the declarations).

Here is file idl_from_c.c: ----------CUT
HERE---------- /*
----------idl_from_c.c---------- ----------
This is a portable interface between C and IDL (or PV-WAVE). With it IDL or
PV-WAVE can be easily used by C programs as a way to plot data, show images,
or do other types of data processing. Data transfer is supported in both
directions.

Note: At this time, it only works on Unix platforms, because of the
handling and naming of environment variables and the names of the IDL or
PV-WAVE program. If someone wants to help me figure out how to make it work
under various Microsoft Windows environments, they are welcome to!

It is portable between many platforms because it works by writing files
which specify what IDL or PV-WAVE are to do, and which transfer data
to IDL or PV-WAVE through other files. It then spawns a child process to
run IDL or PV-WAVE, then reads back data files if required.

IDL is used if the Language #define below is set to 0, PV-WAVE if it is set
to 1.

If you want everything to work right, you must have a licensed version
of IDL or PV-WAVE running. This was done because I consider it unethical

to try to bypass software licensing, and for the very practical reason
that I wish to encourage RSI to continue to provide the free trial version
of IDL.

Specifically, that free trial version is able to open and read files, but can not write them (it also only runs for 7-10 minutes, and has some other restrictions). Accordingly, you can use this software to read in variables and commands into the trial version IDL environment, but can not to output results from the trial version IDL environment back into the C language caller--i.e., the following entry points work correctly with that free trial version: IDLopen IDLwindow IDLarray IDLplotI4 IDLplotR8 IDLWaitEnter IDLclose but the following do not: IDLgetarray IDLgetarray2

In detail the C program uses IDLopen and other calls to specify commands to be executed by the IDL or PV-WAVE program. It uses IDLarray and IDLgetarray to specify data that will be transferred. The user can insert additional (user) commands into the IDL/PV-WAVE procedure with fprintf(IDLfp,"<user command>\n");

The C program then uses IDLclose to start IDL (or PV-WAVE) with an
environment variable set to force execution of those commands.
IDL or PV-WAVE processes those commands, then terminates.

The C program resumes, using calls to IDLgetarray2 to grab back any data
that had been specified using IDLgetarray.

If the C program needs to run any more IDL or PV-WAVE commands at that point, it would need to start over with IDLopen again to specify a new session. It is thus not as "good" as using dynamic linking to call IDL or PV-WAVE, nor does it run as fast, but very few people seem able to figure out how to dynamically link to IDL or PV-WAVE. I haven't yet. Further, dynamic linking does not have a portable interface between computer platforms or revisions of operating systems, or between IDL and PV-WAVE.

Using this program is easier, at least for me!

Written by Mitchell R Grunes, for his own use.
E-Mail: grunesm@ieee.org */

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

#include "idl_from_c.h" /* Declarations for functions defined here.
                           The calling program should also include this. */

#define Language 0      /* 0 for IDL, 1 for PV-WAVE. */
```

```
 /*---------------------------------------------------------- ------------------*/

FILE *IDLfp;           /* File pointer for IDL procedure to run. */

 /*---------------------------------------------------------- ------------------*/

void IDLopen() {       /* Create idl_temp.sh file;
                       Open idl_temp.pro file. */
/* Written by Mitchell R Grunes. */

  IDLfp=fopen("idl_temp.sh","w");
  fprintf(IDLfp,"#!/bin/csh\n");
  #if Language == 0
    fprintf(IDLfp,"setenv IDL_STARTUP idl_temp1.pro\n");
    fprintf(IDLfp,"idl\n");
    fprintf(IDLfp,"unsetenv IDL_STARTUP\n");
  #else
    fprintf(IDLfp,"setenv WAVE_STARTUP idl_temp1.pro\n");
    fprintf(IDLfp,"wave\n");
    fprintf(IDLfp,"unsetenv WAVE_STARTUP\n");
  #endif
  fclose(IDLfp);

  IDLfp=fopen("idl_temp1.pro","w");
  fprintf(IDLfp,"idl_temp\n");
  fprintf(IDLfp,"exit\n");
  fprintf(IDLfp,"end\n");

  fclose(IDLfp);

  IDLfp=fopen("idl_temp.pro","w");

  fprintf(IDLfp,  "function wcolor,i  ; Return scaled color for current
window.\n"  "  ; i=value from 0 to 255.\n"  "  ; For 256 color windows, will
just return i.\n"  "  ; If the window has fewer than 256 colors,\n"  " ;
will return i scaled by the number of colors.\n"  "  ; For 24 bit/pixel
windows, will create\n"  "  ;  corresponding 24 bit code.\n"  "; Written by
Mitchell R Grunes.  Based partly on some ideas from\n" ";  David Fanning.\n"
 "j=byte(i)\n" "if !d.table_size lt 256 then
j=byte(i*float(!d.table_size-1)/255+.5)\n"  "if !d.n_colors le 256 then
begin\n"  "  return,j\n"  "endif else begin  ; Handle true color windows.\n"
"  tvlct,r,g,b,/get\n" "  return,r(j)+256L*g(j)+65536L*b(j)\n"  "endelse\n"
"end\n");

  fprintf(IDLfp,"pro idl_temp\n");
}
```

```
 /*-------------------------------------------------------- ------------------*/

void IDLwindow(         /* Open graphics window */
/* Written by Mitchell R Grunes. */
/* -------------------------Input Parameters---------------------------- */
    int iWin,           /* Window #.  Usually 0. */
    int xsize,          /* X-size in pixels.  0 indicates default. */
    int ysize,          /* Y-size in pixels.  0 indicates default. */
    long colors         /* Color mode.
                        0   to use default # of colors for 8 bit windows.
                        256 to use 256 colors in an 8 bit window.
                        16777216 to use true color windows.
                        Note that IDL does not yet you change color mode,
                        once set--so use same for all windows. */
) {
  fprintf(IDLfp,"colors=%d\n",colors);
  fprintf(IDLfp,
   "if colors le 256 then begin\n"
   "  if !d.name ne 'WIN' and !d.name ne 'WIN32' then device,pseudo=8\n"
   "  wait,.1\n"
   "endif else begin\n"
   "  colors=256L^3L\n"
   "  device,true=24\n"
   "endelse\n"
   "if strmid(!d.name,0,3) eq 'WIN' then window,0\n"
   "window,0,colors=256\n"
   "if strmid(!d.name,0,3) eq 'WIN' then begin\n"
   "  xsz=630\n"
   "  ysz=470\n"
   "endif else begin\n"
   "  xsz=700\n"
   "  ysz=512\n"
   "endelse\n");
  if (xsize > 0) fprintf(IDLfp,"xsz=%d\n",xsize);
  if (ysize > 0) fprintf(IDLfp,"ysz=%d\n",xsize);
  fprintf(IDLfp,"window,%d",iWin);
  if(colors >= 256) fprintf(IDLfp,",colors=%d",colors);
  fprintf(IDLfp,",retain=2,xsize=xsz,ysize=ysz\n"
   "if !prompt eq 'IDL> ' and colors le 256 and $\n"
   " !d.name ne 'WIN' and !d.name ne 'WIN32' then device,pseudo=8\n"
   "if !prompt eq 'IDL> ' and colors le 256 then wait,.1\n"
   "wshow,%d\n",iWin);
}

 /*-------------------------------------------------------- ------------------*/

                    /* Convert 2 byte string to integer */
static int s2i(char *a) {
```

```
  return( a[0]*256 + a[1] );
}

 /*---------------------------------------------------------- ----------------*/

void IDLarray(          /* Transfer array to IDL (or PV-WAVE) from C. */
/* Written by Mitchell R Grunes. */
/* -------------------------Input Parameters---------------------------- */
  char iType[2],        /* Type:
                        "I1"  =Signed 1 byte integer, converted to int.
                        "U1"  =Unsigned 1 byte integer.
                        "I2"  =Signed 2 byte integer.
                        "U2"  =Unsigned 2 byte integer, converted to long.
                        "I4"  =Signed 4 byte integer.
                        "R4"  =4 byte floating point.
                        "R8"  =8 byte floating point. */
  int nDim,             /* # of dimensions in array */
  long iDim1,           /* Size of first  C-style dimension. */
  long iDim2,           /* Size of second C-style dimension. */
  long iDim3,           /* Size of third  C-style dimension. */
  long iDim4,           /* Size of forth  C-style dimension. */
  void *Array,          /* Array to send */
  char *VarNam          /* Name of array in IDL (or PV-WAVE) */

   /********************************************************** *******/
  /* NOTE: IDL dimensions are Fortran-like and are therefore listed */
  /* in reverse order from C-like dimensions.  That is, if the C    */
  /* array is specified by                                          */
  /*    long a[5][10];                                              */
  /* You would use send it to IDL via                               */
  /*    IDLarray("I4",2,5,10,1,1,a,'a');                            */
  /* and the IDL array would  be of the same shape as               */
  /*    a=lonarr(10,5)                                              */
   /********************************************************** *******/

) {
/* -------------------------Local Parameters---------------------------- */
  char FilNam[80];
  FILE *FP;
  long nByte,nElem;
  static long ArrayNum=0; /* Current array file number. */

  sprintf(FilNam,"idlarray%d.sc",ArrayNum);
  printf("Writing %s to %s\n",VarNam,FilNam);
  ArrayNum++;

  printf(" VarNam=%s iType=%s nDim=%d iDim1=%d iDim2=%d iDim3=%d iDim4=%d\n",
   VarNam,iType,nDim,iDim1,iDim2,iDim3,iDim4);
```

```
if (s2i(iType) == s2i("I1") || s2i(iType) == s2i("U1")) {
  nByte=1;
  fprintf(IDLfp,"%s=bytarr(",VarNam);
} else if (s2i(iType) == s2i("I2") || s2i(iType) == s2i("U1")) {
  nByte=2;
  fprintf(IDLfp,"%s=intarr(",VarNam);
} else if (s2i(iType) == s2i("I4")) {
  nByte=4;
  fprintf(IDLfp,"%s=lonarr(",VarNam);
} else if (s2i(iType) == s2i("R4")) {
  nByte=4;
  fprintf(IDLfp,"%s=fltarr(",VarNam);
} else if (s2i(iType) == s2i("R8")) {
  nByte=8;
  fprintf(IDLfp,"%s=dblarr(",VarNam);
} else {
  fprintf(stderr,"Error: Invalid type %d\n",iType);
  exit(1);
}

if (nDim == 1) {
  nElem=iDim1;
  fprintf(IDLfp,"%d)\n",iDim1);
} else if (nDim == 2) {
  nElem=iDim1*iDim2;
  fprintf(IDLfp,"%d,%d)\n",iDim2,iDim1);
} else if (nDim == 3) {
  nElem=iDim1*iDim2*iDim3;
  fprintf(IDLfp,"%d,%d,%d)\n",iDim3,iDim2,iDim1);
} else if (nDim == 4) {
  nElem=iDim1*iDim2*iDim3*iDim4;
  fprintf(IDLfp,"%d,%d,%d,%d)\n",iDim4,iDim3,iDim2,iDim1);
}

FP=fopen(FilNam,"wb");

if (FP == NULL) {
  fprintf(stderr,"Error: Could not open %s\n",FilNam);
  perror("System message:");
  exit(1);
}

if (fwrite(Array,nByte,nElem,FP) != nElem) {
  fprintf(stderr,"Error: Could not write array %s\n",VarNam);
  perror("System message:");
  exit(1);
}
```

```
  fclose(FP);

  fprintf(IDLfp,"openr,10,'%s'\n",FilNam);
  fprintf(IDLfp,"readu,10,%s\n",VarNam);
  fprintf(IDLfp,"close,10\n");

  if (s2i(iType) == s2i("I1")) {
    fprintf(IDLfp,"%s=%s and 255\n",VarNam,VarNam);
  } else if (s2i(iType) == s2i("U2")) {
    fprintf(IDLfp,"%s=%s and 65535L\n",VarNam,VarNam);
  }

}

 /*------------------------------------------------------------ ------------------*/

void IDLgetarray(       /* Transfer array from IDL (or PV-WAVE) to C.
                        The transfer does not occur until you have
                        called IDLclose and IDLgetarray2. */
/* Written by Mitchell R Grunes. */

/* -------------------------Input Parameters---------------------------- */
  char *VarNam,         /* Name of array in IDL (or PV-WAVE) */

/* -------------------------Output Parameters--------------------------- */
  char FilNam[80]       /* File name in which data will be stored.
                        This is a TLA format file which can be read by
                        my read_tla.pro procedure, or Tom Ainsworth's
                        read_array.pro procedure. */
) {

  static long ArrayNum=0; /* Current array file number. */
  sprintf(FilNam,"idlwrarray%d.sc",ArrayNum);
  ArrayNum++;

  printf("Will read %s from %s\n",VarNam,FilNam);

  fprintf(IDLfp,"openw,10,'%s'\n",FilNam);
  fprintf(IDLfp,"writeu,10,size(%s)\n",VarNam);
  fprintf(IDLfp,"writeu,10,%s\n",VarNam);
  fprintf(IDLfp,"close,10\n");
}

 /*------------------------------------------------------------ ------------------*/

void *IDLgetarray2(     /* Transfer array from IDL (or PV-WAVE) to C.
                        You must have previously called IDLgetarray
```

```
                        and IDLclose. */

/* Written by Mitchell R Grunes. */
/* -------------------------Input Parameters--------------------------- */
  char FilNam[80],      /* File name in which data is stored.  This was
                        set by the call to IDLgetarray. */

/* -------------------------Output Parameters--------------------------- */
/* returns: */          /* Pointer to an array that will be allocated,
                        containing the data.  Don't forget to free this
                        space when you are done. */
  char iType[2],        /* Type:
                       "U1" = Unsigned 1 byte integer.
                       "I2" = Signed 2 byte integer.
                       "I4" = Signed 4 byte integer.
                       "R4" = 4 byte floating point.
                       "R8" = 8 byte floating point. */
  int *nDim,            /* # of dimensions in array */
  long *iDim1,          /* Size of first  C-style dimension. */
  long *iDim2,          /* Size of second C-style dimension. */
  long *iDim3,          /* Size of third  C-style dimension. */
  long *iDim4           /* Size of forth  C-style dimension. */

   /************************************************************ ************/
  /* NOTE: IDL dimensions are Fortran-like and are therefore listed      */
  /* in reverse order from C-like dimensions.  That is, if the IDL       */
  /* array were shaped by                                                */
  /*    a=lonarr(10,5)                                                   */
  /* You would specify sending it to C via                               */
  /*    long *a;                                                         */
  /*    char iType[2];                                                   */
  /*    int nDim;                                                        */
  /*    long iDim1,iDim2,iDim3,iDim4;                                    */
  /*     a=IDLgetarray2(FilNam,iType,&nDim,&iDim1,&iDim2, &iDim3,&iDim4);  */
  /* a would would have the same shape as though it had been defined by  */
  /*    long a[5][10];                                                   */
  /* The other variables would be set to "I4",2,5,10,1, and 1,           */
  /* respectively.                                                       */
   /************************************************************ ************/

) {


/* -------------------------Local Parameters---------------------------- */

  long temp,nElem,nByte;

  void *array;
```

```
FILE *FP=fopen(FilNam,"rb");
if (FP == NULL) {
  fprintf(stderr,"Error: Could not open %s\n",FilNam);
  perror("System message:");
  exit(1);
}

printf("Reading variable from %s\n",FilNam);

if (fread(&temp,4,1,FP) != 1) {
  fprintf(stderr,"Error: Could not read shape\n");
  perror("System message:");
  exit(1);
}
*nDim=temp;

if (*nDim >= 1) fread(iDim1,4,1,FP);
if (*nDim >= 2) fread(iDim2,4,1,FP);
if (*nDim >= 3) fread(iDim3,4,1,FP);
if (*nDim >= 4) fread(iDim4,4,1,FP);

fread(&temp,4,1,FP);

if (fread(&nElem,4,1,FP) != 1 || temp <= 0 || *nDim < 0 || *nDim > 4) {
  fprintf(stderr,"Error: Could not read shape\n");
  perror("System message:");
  exit(1);
}

if (temp == 1) {
  strcpy(iType,"U1");
  nByte=1;
} else if (temp == 2) {
  strcpy(iType,"I2");
  nByte=2;
} else if (temp == 3) {
  strcpy(iType,"I4");
  nByte=4;
} else if (temp == 4) {
  strcpy(iType,"R4");
  nByte=4;
} else if (temp == 5) {
  strcpy(iType,"R8");
  nByte=8;
} else {
  fprintf(stderr,"Error: Incorrect type\n");
  exit(1);
```

```
  }

  array = malloc(nElem*nByte);
  if (array == NULL) {
    fprintf(stderr,"Error: Could not allocate memory for read\n");
    exit(1);
  }

  if (fread(array,nByte,nElem,FP) != nElem) {
    fprintf(stderr,"Error: Could not read array\n");
    perror("System message:");
    exit(1);
  }

  fclose(FP);

  return(array);
}

 /*-------------------------------------------------------- ------------------*/

void IDLplotI4(         /* Plot long (assumed 4 byte) arrays */
/* Written by Mitchell R Grunes. */
/* -------------------------Input Parameters---------------------------- */
  long N,               /* # of points--make negative if X is undefined */
  long X[],             /* Array of X coordinates */
  long Y[],             /* Array of Y coordinates */
  char title[],         /* Title of plot */
  int color             /* color # to use. */
) {
/* -------------------------Local Parameters---------------------------- */
  long NN;
  NN=N < 0 ? -N : N;
  if (N >= 0) IDLarray("I4",1,NN,1,1,1,X,"X");
              IDLarray("I4",1,NN,1,1,1,Y,"Y");
  if (N >= 0) {
    fprintf(IDLfp,"plot,x,y,title='%s',color=wcolor(%d)\n",title,color);
  } else {
    fprintf(IDLfp,"plot,y,title='%s',color=wcolor(%d)\n",title,color);
  }
}

 /*-------------------------------------------------------- ------------------*/

void IDLplotR8(         /* Plot double (assumed 8 byte) arrays */
/* Written by Mitchell R Grunes. */
/* -------------------------Input Parameters---------------------------- */
  long N,               /* # of points--make negative if X is undefined */
```

```
  double X[],           /* Array of X coordinates */
  double Y[],           /* Array of Y coordinates */
  char title[],       /* Title of plot */
  int color           /* color # to use. */
) {
/* -------------------------Local Parameters---------------------------- */
  long NN;
  NN=N < 0 ? -N : N;
  if (N >= 0) IDLarray("R8",1,NN,1,1,1,X,"X");
              IDLarray("R8",1,NN,1,1,1,Y,"Y");
  if (N >= 0) {
    fprintf(IDLfp,"plot,x,y,title='%s',color=wcolor(%d)\n",title,color);
  } else {
    fprintf(IDLfp,"plot,y,title='%s',color=wcolor(%d)\n",title,color);
  }
}


 /*------------------------------------------------------------ ------------------*/

void IDLWaitEnter() {  /* "Hit enter to continue:" */ /* Written by Mitchell
R Grunes. */  fprintf(IDLfp,"print,"\n");  fprintf(IDLfp,"a="\n");
fprintf(IDLfp,"read,'Hit <Enter to continue>:',a\n"); }
 /*------------------------------------------------------------ ----------------
-*/

void IDLclose() {       /* Close idl_temp.pro file;
                        Use IDL or PV-WAVE to run it. */
/* Written by Mitchell R Grunes. */
  fprintf(IDLfp,"exit\n");
  fprintf(IDLfp,"end\n\n");
  fprintf(IDLfp,"idl_temp\n");
  fprintf(IDLfp,"exit\n");
  fprintf(IDLfp,"end\n");
  fclose(IDLfp);

  system("csh idl_temp.sh");
  system("rm idl_temp.sh idl_temp1.pro idl_temp.pro idlarray*.sc");
}

 /*------------------------------------------------------------ ------------------*/

/*  Sample calling program

#include "idl_from_c.h"
void main() {
  long Y[5]={4,2,1,3,6};
  char Im[2][3]={4,2,1,3,6,2};
```

```
  IDLopen();
  IDLwindow(0L,0,0,256);
  fprintf(IDLfp,"loadct,4\n");
  IDLplotI4(-5,Y,Y,"Test",200);
  IDLWaitEnter();

  IDLarray("U1",2,2,3,1,1,Im,"A");
  fprintf(IDLfp,"erase\n");
  fprintf(IDLfp,"tvscl,rebin(a,300,200,/sample)\n");
  IDLWaitEnter();

  {
    char FilNam[80],iType[2];
    int  nDim;
    long iDim1,iDim2,iDim3,iDim4;
    char *b;
    int i,j;

    fprintf(IDLfp,"b=byte(indgen(3,3))\n");
    IDLgetarray("b",FilNam);
    IDLclose();
    b=IDLgetarray2(FilNam,  iType,&nDim,&iDim1,&iDim2,&iDim3,&iDim4) ;
    printf("Type=%s nDim=%d iDim1=%d iDim2=%d\n",
     iType,nDim,iDim1,iDim2);
    for (i=0; i<iDim1; i++) {
      for (j=0; j<iDim1; j++) {
        printf("b[%d,%d]=%d\n",i,j,b[i*iDim2+j]);
      }
    }
    free(b);
  }
}

*/
--------------------------------CUT HERE-------------------------------


Here is file idl_from_c.h:
--------------------------------CUT HERE-------------------------------
#ifndef FILE
  #include <stdio.h>
#endif

extern FILE *IDLfp;     /* File pointer for IDL procedure to run. */

void IDLopen();
```

```
void IDLwindow(int iWin,int xsize,int ysize,long colors);

void IDLarray(char *iType,int nDim,long iDim1,long iDim2,
  long iDim3,long iDim4,void *Array,char *VarNam);

void *IDLgetarray2(char FilNam[80],char iType[2],int *nDim,long *iDim1,
  long *iDim2,long *iDim3,long *iDim4);

void IDLplotI4(long N,long X[],long Y[],char title[],int color);

void IDLplotR8(long N,double X[],double Y[],char title[],int color);

void IDLWaitEnter();

void IDLclose();
--------------------------------CUT HERE------------------------------

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