Subject: Re: Different Platforms
Posted by mgs on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3473138C.B39FFA08@ssec.wisc.edu>, Liam Gumley
<Liam.Gumley@ssec.wisc.edu> wrote:

> Neil Winrow wrote:
>
>> I have written a number of widget programs, which are visually correct
>> on my PC, however when they are run on the silicon graphics machines the
...
> (1) If your development platform is a PC, then make sure that once a day
> you test the program on a Unix box, and a Mac. Unless you do this, it is
> easy in your program design to go down a path from which there is no
> return.
>
> (2) As David Fanning suggested, having the commands
>
> Device, Set_Character_Size = [6, 9]
> Widget_Control, Default_Font = '7x13'
>
> in your IDL startup file will guarantee consistent graphics font and widget
> font sizes on all *Unix* platforms. However, these commands are ignored on
> PCs. You should try David's STR_SIZE program.
>
> (3) You may wish to look into using *hardware* fonts for graphics. I've
> been experimenting with them in Unix, and getting pretty good results. You
> can see the list of available hardware fonts in IDL by using the command
>
...
> The major drawback to using hardware fonts is that they do not rotate
> automatically, e.g.
>
> plot, indgen(10), xtitle = 'X AXIS', ytitle = 'Y AXIS'
>
> when using hardware fonts will give you the Y title plotted vertically, but
> not rotated. You can get around this by creating the Y axis title in a
> pixmap, reading an array of byte data from the pixmap, rotating the array
> using ROTATE, and TVing the rotated array next to your Y axis. This takes a
> bit of messing around, but the resulting graphics plots look *much* more
> professional than the default vector fonts.

And I thought I had a lot of patience for dealing with interface problems.

> (4) When creating widget programs, be very careful about using XSIZE and
> YSIZE keywords. I try to use them only for WIDGET_DRAW, WIDGET_LABEL, and
> WIDGET_BUTTON sizing.

>
> (5) Rely on the ROW, COLUMN, and alignment keywords when creating widget
> bases to position your widgets.

I tried staying away from [XY]Size for a long time. Now I find that it's
the only way to get a decent cross-platform look. I only rely on the
XSize, I don't think I've used YSize anywhere. Everything I use has Row
and Column keywords. As I mentioned earlier in this thread, I rely on the
Widget_Info(widget_id, /Geometry) call and my own widget size structure to
provide the info I need to get correctly sized widgets. The following URL
shows a very busy interface http://ww2.sd.cybernex.net/~mgs/IV_IAS_BB.html
with Mac and UNIX versions side by side. Well, it's the same except for an
overlapping hierarchy near the botton has been toggled. I haven't updated
the images in months and there has been a lot of development since then.
Maybe it's time to do that for both platforms and add in some additional
info about making the stuff work correctly.

> PS I've attached a Unix hardware font selection routine below - I'll be
> modifying it soon to work on PC and Mac.

Well, here's a version that runs on Mac and UNIX. It was tested a couple
years ago on a PC, but hasn't been run on a PC since. It could be extended
to include italics and additional sizes without too much of a headache.

Usage: Font_Struct = FontGen()

```
 ;############################################################## ################
; Author: Mike Schienle
; $Workfile:   fontgen.pro  $
; $Revision:   1.1  $
; Orig Date:  96-12-17
; $Modtime:   Wed Oct 01 10:26:18 1997  $
 ;############################################################## ################

FUNCTION FontGen, PROP=prop, MONO=mono, SYMBOL=symbol
   IF (N_Elements(prop) EQ 0L) THEN $
      prop = 'times'

   IF (N_Elements(mono) EQ 0L) THEN $
      mono = 'courier'

   IF (N_Elements(symbol) EQ 0L) THEN $
      symbol = 'symbol'

   ;   get Operating System info
   IF (!Version.OS_Family EQ 'unix') THEN BEGIN
      ;   We're using UNIX
      ;   specify the names of proportional and monospace fonts
```

```
   asFontName = ['-*-' + prop + '-', $
      '-*-' + mono + '-', $
      '-*-' + symbol + '-']
   ;  specify font weights
   asFontWeight = ['medium-', 'bold-']
   ;  specify "extras" - string completers
   asFontExtra = ['r-*-*-', '-*']
ENDIF ELSE BEGIN
   ;  Non-UNIX (Mac, Windows)
   ;  specify the names of proportional and monospace fonts
   asFontName = [prop + '*', mono + '*', symbol + '*']
   ;  specify font weights
   asFontWeight = ['', 'bold*']
   ;  specify "extras" - string completers
   asFontExtra = ['', '']
ENDELSE

;  font strings
;  UNIX style
;  -adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
;  Mac/PC Style
;  times*bold*18, times*18

;  font sizes
asFontSize = ['10', '12', '18', '24']
;  abbreviated font wieghts
asFontWAbbr = ['m', 'b']

;  create a structure of font names, proportional and monospace
sCmdFont = 'mFont = {'
FOR fs = 0, (n_elements(asFontSize) - 1) DO $
   FOR fw = 0, (n_elements(asFontWeight) - 1) DO $
      sCmdFont = sCmdFont + $
         'prop' + asFontSize(fs) + asFontWAbbr(fw) + ':"' + $
         asFontName(0) + asFontWeight(fw) + asFontExtra(0) + $
         asFontSize(fs) + asFontExtra(1) + '", ' + $
         'mono' + asFontSize(fs) + asFontWAbbr(fw) + ':"' + $
         asFontName(1) + asFontWeight(fw) + asFontExtra(0) + $
         asFontSize(fs) + asFontExtra(1) + '", ' + $
         'symbol' + asFontSize(fs) + asFontWAbbr(fw) + ':"' + $
         asFontName(2) + asFontWeight(fw) + asFontExtra(0) + $
         asFontSize(fs) + asFontExtra(1) + '", '

sCmdFont = StrMid(sCmdFont, 0, StrLen(sCmdFont) - 2) + '}'
;  example follows - Mac/PC version
;  mFont = {prop10m:"times*10", mono10m:"courier*10", $
;  prop10b:"times*bold*10", mono10b:"courier*bold*10", $
;  ...
```

```
   ;   prop24m:"times*24", mono24m:"courier*24", $
   ;   prop24b:"times*bold*24", mono24b:"courier*bold*24"}

   status = Execute(sCmdFont)

   ;   return the font structure
   Return, mFont
END
```

--
Mike Schienle Interactive Visuals
mgs@sd.cybernex.net http://ww2.sd.cybernex.net/~mgs/

Subject: Re: Different Platforms
Posted by mgs on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.edc959fe74be3f79896ac@news.frii.com>, davidf@dfanning.com (David Fanning) wrote:

> Neil Winrow (ncw@dl.ac.uk) writes:
>
>> I have written a number of widget programs, which are visually correct
>> on my PC, however when they are run on the silicon graphics machines the
>> layout starts to go terribly wrong. The character sizes are wrong, and
>> the labelling carried out using the 'XYOUTS' call is all wrong. The
>> whole window sizing falls down. The programs are going to be used on
>> PC's, silicon graphics, and MAC's. Could anyone offer me a few pointers
>> on how to correct these problems to run the programs on the different
>> platforms.
>
> There is nothing quite like trying to get IDL widget programs
> to run on PCs, SGIs and Macs to get you to question your
> sanity. And, of course, as soon as you have it figured out,
> someone buys a Sun and then you have to deal with *those*
> tinsy-tiny 12-point fonts. You soon realize that "font size"
> must be one of those words like "true love" that is open to
> all kinds of interpretation.
...

> Let us know what you come up with. I'll add it to the list.
> It will make interesting reading 100 years from now. :-)

I put together a couple functions to help out with this. One returns a structure based on !Version.OS_Family which contains offsets for widget parameters such as scroll bars, frames, and button types, with adjustments based on the window manager. The other creates a set of fontnames which

can be called generically cross-platform. Sizing of widgets also relies on calls to Widget_info with the /Geometry keyword set. It's ugly, but useful.

```
   mOsInfo = {$
      sDiskChar: '', $       ;   disk separator character
      sDirChar: '', $        ;   directory separator character
      sWinName: '', $        ;   Window manager name
      mBuffer: {$            ;   buffer offsets
         scroll: 0, $        ;   scroll bar
         frame: 0, $         ;   frame boundary
         exclusive: 0, $     ;   radio button (exclusive)
         nonexclusive: 0, $  ;   checkbox button (nonexclusive)
         button: 0}, $       ;   button (regular)
   ;   followed by settings for each platform - UNIX, Mac, Windows, VMS
```

FontGen returns a structure of fonts for proportional and monotype sizes of 10, 12, 18, and 24 at medium and bold weights. The default fonts are times and courier but can be whatever you specify. The referenced font names are based on the platform you are running IDL on: Mac/PC mono10b would be "courier*bold*10", UNIX mono10b would be "*-courier-bold-r-normal--10-*"

```
   mFont = FontGen()

   ;   declare some generic font names for common usage
   labelFont = mFont.prop12b ; proportional (times), 12 point, bold
   buttonFont = mFont.prop12m ; proportional, 12 point, medium
   fieldFont = mFont.mono10m ; monotype (courier), 10 point, medium

   ;    the mMisc structure contains mFont as well as mOsInfo
   wLabel = Widget_Label(wBase, Value='Output on Close: ', $
      Font=mMisc.labelFont)

   ;   get dimensions of base
   mGeoBase = Widget_Info(wBase, /Geometry)

   ;   get dimensions of wLabel
   mGeoLabel = Widget_Info(wLabel, /Geometry)

   asWriteText = ['Trend', 'Report', 'Residuals']

   ;   display the write buttons
   ;   CW_BGroup2 is a modification of CW_BGroup that allows
   ;   additional font, and sizing info to be set.
   wBGWrite = CW_BGroup2(wBaseOutput, asWriteText, /Row, /Frame, $
      /Align_Left, ButtonSize=((mGeoBase.XSize - mGeoLabel.XSize) / $
      N_Elements(asWriteText) - mMisc.mOsInfo.mBuffer.nonexclusive - $
```

```
      mMisc.mOsInfo.mBuffer.frame /  N_Elements(asWriteText)), $
      /NonExclusive, ButtonFont=mMisc.buttonFont)

---------------------- wBase ----------------------
         ---------------------------------------
Output OC: | [] Trend   []   Report [] Residuals  |
         ---------------------------------------
 wLabel                    wBGWrite
-------------------------------------------------
```

assuming:
wBase = 400 pixels (from Widget_Info(wBase, /Geometry))
wLabel = 130 pixels (from Widget_Info(wBase, /Geometry))
nonexclusive = 16 pixels (width of a non-exclusive button)
frame = 6 pixels (width of a frame - both sides)
N_Elements(asWriteText) = 3

buttonsize = ((wbase - wlabel) / 3) - nonexclusive - frame / 3
buttonsize = ((400 - 130) / 3) - 16 - 6 / 3 = 72 pixels
The framed group will fit in the base next to the label as shown above.

Since I have settings for the values of nonexclusive and frame in my
mOsInfo structure and the Widget_info(widget_id, /Geometry) call returns
the current size of the specified widgets, I have a platform-independent
sizing system.

--
Mike Schienle   Interactive Visuals
mgs@sd.cybernex.net   http://ww2.sd.cybernex.net/~mgs/

Subject: Re: Different Platforms
Posted by David Foster on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Neil Winrow wrote:
>
> Can anyone offer me any advice.
>
> I have written a number of widget programs, which are visually correct
> on my PC, however when they are run on the silicon graphics machines the
> layout starts to go terribly wrong. The character sizes are wrong, and
> the labelling carried out using the 'XYOUTS' call is all wrong. The
> whole window sizing falls down. The programs are going to be used on
> PC's, silicon graphics, and MAC's. Could anyone offer me a few pointers
> on how to correct these problems to run the programs on the different
> platforms.
>

> Many Thanks In Advance
>
> Neil.

When I needed to run IDL programs on an SGI that were
originally written on a Sun system, I had to make the following
modifications to the 'idl_startup' file on the SGI:

```
; Choose pseudo-color 8-bit visual
device, pseudo_color=8

; Select default backing-store method to be provided by IDL, as SGI
;  X server does not seem to provide it
device, retain=2

; Change size of font so programs fit on-screen (IRIX 4.0 or later)
; Reference: sgi.doc document in $IDL_DIR/notes
WIDGET_CONTROL, $
   DEFAULT_FONT="-adobe-helvetica-bold-r-normal-*-14-100-*-*-*-*-* "

; Set default plotting font to same hardware font above
;  (Create a pixmap window to avoid window creation upon
;   calling DEVICE, FONT= ; then delete window)
window, xsize=5,ysize=5,/free,/pixmap
!p.font = 0      ; Use hardware font
device, FONT="-adobe-helvetica-bold-r-normal-*-10-100-*-*-*-*-*"
wdelete          ; Delete window created
```

Of course this assumes that you have access to the IDL configuration
on the SGI. But this worked well for me.

Dave
--

~~~~~~ ~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~

Subject: Re: Different Platforms
Posted by Liam Gumley on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Neil Winrow wrote:

> I have written a number of widget programs, which are visually correct
> on my PC, however when they are run on the silicon graphics machines the
> layout starts to go terribly wrong. The character sizes are wrong, and
> the labelling carried out using the 'XYOUTS' call is all wrong. The
> whole window sizing falls down. The programs are going to be used on
> PC's, silicon graphics, and MAC's. Could anyone offer me a few pointers
> on how to correct these problems to run the programs on the different

(1) If your development platform is a PC, then make sure that once a day
you test the program on a Unix box, and a Mac. Unless you do this, it is
easy in your program design to go down a path from which there is no
return.

(2) As David Fanning suggested, having the commands

```
Device, Set_Character_Size = [ 6, 9 ]
Widget_Control, Default_Font = '7x13'
```

in your IDL startup file will guarantee consistent graphics font and widget
font sizes on all *Unix* platforms. However, these commands are ignored on
PCs. You should try David's STR_SIZE program.

(3) You may wish to look into using *hardware* fonts for graphics. I've
been experimenting with them in Unix, and getting pretty good results. You
can see the list of available hardware fonts in IDL by using the command

```
Device, Get_Fontnames = Fontnames
```

which returns a string array of all the font names defined on your system
(Unix or PC or Mac). You can then select a font that looks consistent on
all platforms (say a 14 point Helvetica font), and make it the default
graphics and widget font by the commands

```
device, font = name     ; set graphics font
!p.font = 0     ; use hardware fonts for graphics instead of vector fonts
widget_control, default_font = name     ; set the default widget font
```

The major drawback to using hardware fonts is that they do not rotate
automatically, e.g.

```
plot, indgen(10), xtitle = 'X AXIS', ytitle = 'Y AXIS'
```

when using hardware fonts will give you the Y title plotted vertically, but
not rotated. You can get around this by creating the Y axis title in a
pixmap, reading an array of byte data from the pixmap, rotating the array
using ROTATE, and TVing the rotated array next to your Y axis. This takes a
bit of messing around, but the resulting graphics plots look *much* more

professional than the default vector fonts.

(4) When creating widget programs, be very careful about using XSIZE and YSIZE keywords. I try to use them only for WIDGET_DRAW, WIDGET_LABEL, and WIDGET_BUTTON sizing.

(5) Rely on the ROW, COLUMN, and alignment keywords when creating widget bases to position your widgets.

Cheers,
Liam.
PS I've attached a Unix hardware font selection routine below - I'll be modifying it soon to work on PC and Mac.

```
PRO SELECT_FONT, HELVETICA = HELVETICA, TIMES = TIMES, $
  PALATINO = PALATINO, COURIER = COURIER, BOLD = BOLD, ITALIC = ITALIC, $
  SIZE = SIZE, NAME = NAME

;+
; Purpose:
;    Select a Unix hardware font for IDL graphics.
;
; Usage:
;    SELECT_FONT
;
; Input:
;    None required.
;
; Optional Keywords:
;    /HELVETICA    Select a Helvetica font (default)
;    /TIMES        Select a Times font
;    /PALATINO     Select a Palatino font
;    /COURIER      Select a Courier font
;    /BOLD         Select a bold font (default is no bold)
;    /ITALIC       Select and italic font (default is no italics)
;    SIZE          If set to a named variable, sets the font size
(default=12)
;    NAME          If set to a named variable, returns the font name
selected
;
; Revised:
;    17-OCT-1997 Liam Gumley, CIMSS/SSEC
;    Created
;
; Notes:
;    (1) This procedure currently only works on Unix IDL platforms.
```

```
;    (2) The NAME value returned by SELECT_FONT can be used to set the
;        default widget font by the command
WIDGET_CONTROL,DEFAULT_FONT=NAME
;
; Example:
; !P.MULTI=[0,1,2,0,0]
; PLOT,INDGEN(10)
; SELECT_FONT,/BOLD
; PLOT,INDGEN(10)
;-

;- this version is only for Unix at the moment

if !version.os_family ne 'unix' then begin
  message, /continue, 'Only works on Unix at the moment'
  return
endif

;- check keyword flags

if not keyword_set( helvetica ) then helvetica = 0
if not keyword_set( times ) then times = 0
if not keyword_set( palatino ) then palatino = 0
if not keyword_set( courier ) then courier = 0
if not keyword_set( bold ) then bold = 0
if not keyword_set( italic ) then italic = 0

;- check keyword values

if n_elements( size ) eq 0 then size = 12

;- set keyword return values

name = ''

;- create font search string

case 1 of
  helvetica : search = '*helvetica*'
  times     : search = '*times*'
  palatino  : search = '*palatino*'
  courier   : search = '*courier*'
  else      : search = '*helvetica*'
endcase

if bold then begin
  search = search + 'bold-'
endif else begin
```

```
  search = search + 'medium-'
endelse

if italic then begin
  search = search + 'o-normal*'
endif else begin
  search = search + 'r-normal*'
endelse

;- open a graphics window

window, /free, /pixmap

;- get list of font names matching search string

device, font = search, get_fontnames = fontnames

;- find a font size that matches

fontstring = '--' + strcompress( long( size > 8 ), /remove_all ) + '-'
index = strpos( fontnames, fontstring )
loc = where( index ne -1, count )

;- use font if it was found, or else set graphics font size

if count ge 1 then begin
  name = fontnames( loc(0) )
  device, font = name
  !p.font = 0
endif else begin
  message, /continue, 'Requested font was not found - using graphics font
instead'
endelse

;- close graphics window

wdelete, !d.window

end
```

Subject: Re: Different Platforms
Posted by davidf on Wed, 19 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Neil Winrow (ncw@dl.ac.uk) writes:

> I have written a number of widget programs, which are visually correct

> on my PC, however when they are run on the silicon graphics machines the
> layout starts to go terribly wrong. The character sizes are wrong, and
> the labelling carried out using the 'XYOUTS' call is all wrong. The
> whole window sizing falls down. The programs are going to be used on
> PC's, silicon graphics, and MAC's. Could anyone offer me a few pointers
> on how to correct these problems to run the programs on the different
> platforms.

There is nothing quite like trying to get IDL widget programs
to run on PCs, SGIs and Macs to get you to question your
sanity. And, of course, as soon as you have it figured out,
someone buys a Sun and then you have to deal with *those*
tinsy-tiny 12-point fonts. You soon realize that "font size"
must be one of those words like "true love" that is open to
all kinds of interpretation.

I once got so frustrated I took a survey of the 10 best
IDL widget programmers I knew. "How do you size strings
in your widget programs?", I asked. Would you believe I
got 10 completely different answers and that almost
every single answer had more to do with occult
science than with computer science? I wouldn't
be able to withstand the public ridicule if I
published the answers here.

SGIs seem to be the worst. I don't know why. They
have HUGE default fonts. Liam Gumley, who should know,
suggests you try something like this in your SGI start-up
file:

```
Device, Set_Character_Size = [ 6, 9 ]
Widget_Control, Default_Font = '7x13'
```

I have had fairly good success with a program named
STR_SIZE that you can get from my web page. It calculates
the proper character size to get a target string to be
be a particular size (in normalized coordinates)
in the output window. But this means setting the
CharSize keyword on all graphics output commands.
(Not a bad idea when you are writing what you hope will
be portable IDL applications, by the way. Then, at least,
if your programs look lousy on some person's system they
can change the CharSize parameter, which I usually have
as a keyword to the main program.)

I've had pretty good luck with something like this:

```
rightSize = Str_Size('This is a default string', 0.25)
```

Plot, data, CharSize=rightSize

Let us know what you come up with. I'll add it to the list.
It will make interesting reading 100 years from now. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Different Platforms
Posted by Karsten Rodenacker on Thu, 20 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Following the track of arguments I am surprised not to see any hint
concerning resources of the windowing system.

I experienced the bizar behaviour too, even on ONE unix platform using
either idl in command mode or by idlde. If we could learn a bit more
about ajusting the Idl and Idlde window system resources I think most of
the problems mentioned will disappear.

Regards
Karsten

Subject: Re: Different Platforms
Posted by rivers on Sat, 22 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3473138C.B39FFA08@ssec.wisc.edu>, Liam Gumley <Liam.Gumley@ssec.wisc.edu>
writes:
> Neil Winrow wrote:
>
>> I have written a number of widget programs, which are visually correct
>> on my PC, however when they are run on the silicon graphics machines the
>> layout starts to go terribly wrong. The character sizes are wrong, and
>> the labelling carried out using the 'XYOUTS' call is all wrong. The
>> whole window sizing falls down. The programs are going to be used on
>> PC's, silicon graphics, and MAC's. Could anyone offer me a few pointers

>> on how to correct these problems to run the programs on the different

For widgets I use the following methods:

- Use a function which returns font names in a system-dependent manner.
    default_font = get_font_name(/SMALL, /HELVETICA)
    button_font = get_font_name(/LARGE, /HELVETICA, /BOLD)
    label_font = get_font_name(/MEDIUM, /HELVETICA)

   widget_control, default_font=default_font
   Use button_font and label_font when creating buttons and labels, etc.

   My get_font_name.pro is attached. It works on Unix, VMS and PC. Have not
   had a Mac to try it on.

- When laying out widgets of different types (e.g. buttons, labels and
  droplists) which need to line up in columns or rows do the following:

    1) Create dummy widgets with the same commands you will be using for your
       real widgets before you call WIDGET_CONTROL, /REALIZE (that way you won't
       see them).
    2) Use geometry = widget_control(id, /geometry) to determine the size of
       each widget in pixels (i.e. geometry.scr_xsize and geometry.scr_ysize)
    3) Step 2 will tell you which widget is the biggest.
    4) Now create your real widgets, using the SCR_XSIZE and SCR_YSIZE keywords
       to explicitly set the size of all of the widgets to the size of the
       biggest one.

This has worked fine for me on PCs, VMS and Unix.

Here is an example from one program:

```
;************************************************************ *
   row = widget_base(base, /row, /frame)
   ; Determine height for all of the widgets in this row
   dummy = widget_text(base, xsize=6)
   geometry = widget_info(dummy, /geometry)
   widget_control, dummy, /destroy
   scr_ysize = geometry.scr_ysize
   scr_xsize = geometry.scr_xsize

   col = widget_base(row, /column)
   t = widget_label(col, value='ROI', font=self.fonts.label)
   for i=0, nrois-1 do begin
      t = widget_label(col, value=strtrim(string(i),2), scr_ysize=scr_ysize)
   endfor

   col = widget_base(row, /column)
   t = widget_label(col, value='Use?', font=self.fonts.label)
```

```
   for i=0, nrois-1 do begin
      cal.widgets.use_flag[i] = $
         widget_droplist(col, value=['No','Yes'], scr_ysize=scr_ysize)
      widget_control, cal.widgets.use_flag[i], $
               set_droplist_select=cal.roi[i].use
   endfor
;*********************************************************** *
```

Here is get_font_name.pro

```
function get_font_name, $
      helvetica=helvetica, times=times, courier=courier, $
      tiny=tiny, small=small, medium=medium, large=large, huge=huge, $
      size=size, $
      bold=bold, italic=italic, $
      dpi75=dpi75, dpi100=dpi100

; Returns the name of the font with the specified characteristics

if (!version.os_family eq 'Windows') then begin
font = ''
if keyword_set(helvetica) then font = font + 'Helvetica' else $
if keyword_set(times)     then font = font + 'Times' else $
if keyword_set(courier)   then font = font + 'Courier' else $
                               font = font + 'MS San Serif'

if keyword_set(bold) then font = font + '*Bold'
if keyword_set(italic) then font = font + '*Italic'
if keyword_set(tiny)   then size=0
if keyword_set(small)  then size=1
if keyword_set(medium) then size=2
if keyword_set(large)  then size=3
if keyword_set(huge)   then size=4
if (n_elements(size) eq 0) then size=2
font_size_strings = ['12', '14', '16', '18', '20']
size = (size > 0) < (n_elements(font_size_strings)-1)
font = font + '*' + font_size_strings(size)
return, font

endif else if (!version.os_family eq 'Mac') then begin
  font='Helvetica'
  return, font
endif else begin

; VMS and Unix
font = '-adobe-'
if keyword_set(helvetica) then font = font + 'helvetica-' else $
if keyword_set(times)     then font = font + 'times-' else $
```

```
if keyword_set(courier)   then font = font + 'courier-' else $
                          font = font + 'helvetica-'

if keyword_set(bold) then font = font + 'bold-' else font = font + 'medium-'
if keyword_set(italic) then font = font + 'o-' else font = font + 'r-'
font = font + 'normal--*-'

if keyword_set(tiny)   then size=0
if keyword_set(small)  then size=1
if keyword_set(medium) then size=2
if keyword_set(large)  then size=3
if keyword_set(huge)   then size=4
if (n_elements(size) eq 0) then size=2
font_size_strings = ['80-', '100-', '120-', '140-', '180-']
size = (size > 0) < (n_elements(font_size_strings)-1)
font = font + font_size_strings(size)

if keyword_set(dpi100) then font = font + '100-100-' else $
if keyword_set(dpi75) then font = font + '75-75-' else $
                           font = font + '*-*-'
font = font + '*-*-iso8859-1'
return, font
endelse

end
```