Subject: Re: Porting IDL
Posted by Liam Gumley on Wed, 12 Aug 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Bernard Puc wrote:
> Thanks for the replies concerning software version control. Here's
> another query for the collective experience of the group: We are
> considering porting our IDL application to a PC platform (written
> for SGI workstation). Does anybody know if there would be
> significant advantages in time and effort in coding if we ported to
> a Linux platform in contrast to Win95?

Here's a few thoughts.

Byte-swapping

If your application reads any kind of 'binary' data that contains integer or floating point values, you need to be aware of byte-swapping (unless the data is in a portable format like ASCII/XDR/HDF/netCDF/CDF). I typically design the read routine to handle byte-swapping transparently, so that I don't have to worry about whether I'm on an SGI or a PC. My usual procedure for reading a binary file is

(1) Open the file
(2) Read a single 'record' into an IDL structure
(3) Decode part of the structure (say date and time) to see if the values are sensible
(4) If the values are sensible, then proceed with read as normal
(5) If the values are not sensible, byte-swap the entire structure using SWAP_ENDIAN
(6) Decode the same values as in (3) to see if the values are sensible
(7) If the values are sensible, set byte-swap flag ON. If values are not sensible, tell the user the file format was not recognized and quit.
(8) Read subsequent records, and byte-swap if the byte-swap flag is set

This is usually only necessary where you had no direct control over the way the file was created to begin with. If you are responsible for creating the file, then using one of the portable formats mentioned previously will make you life *much* easier.

(There is no difference between Windows and Linux regarding byte-swapping).

Widgets

If your application uses widgets, then as long as you were careful to

allow IDL to lay out the widgets (i.e. you used XSIZE/XOFFSET/XPAD etc. keywords sparingly), you should be ok. The bottom line is that you have to suck it and see (try your application on the PC and see what it looks like). If you use the standard Motif window manager under Linux (*not* FVWM et al.), the chances are good that the widgets will look pretty much like they do on the SGI. If you use a standard font for widgets, e.g.

widget_control, default_font = '7x13'

then your chances are even better. Try Linux and Windows, and see which one looks better to you.

Direct graphics vector fonts

If you are creating direct graphics with vector fonts, then pick a standard size and stick to it, e.g.

device, set_character_size = [6, 9]

Otherwise the graphics you have carefully crafted on the SGI will look a lot different in Linux or Windows, or any other Unix platform for that matter.

Colors

Your safest bet (unless you're of David Fanning caliber!) is to stick with 8 bit mode. If you can figure out how to handle the various flavors of 8/16/24/32 bit graphics available on PCs transparently, then my hat is off to you. Put something like the following in your IDL startup file:

```
if !version.os_family eq 'unix' then device, pseudo = 8
device, retain = 2, decomposed = 0
window, /free, /pixmap, colors = -5
widget_control, default_font = '7x13'
device, set_character_size = [ 6, 9 ]
```

and make sure you use the same startup file on all platforms. These commands must be executed right at startup, because once you create the first graphics window, the number of colors is set for the rest of that IDL session.

Save files

If you want to keep the SGI as your development platform, then you don't need to send all the application source code to the PC. Just do the following:

(1) Start a new IDL session on the SGI
(2) Do a .COMPILE on all the application routines (let's say the main program is myapp.pro)
(3) Save the compiled routines using
    SAVE,/ROUTINES,/XDR,FILE='myapp.sav'
(4) Send the binary file myapp.sav to the PC
(5) Start IDL on the PC (using the same startup file as the SGI)
(6) Type
    RESTORE,'myapp.sav'
(7) Type
    myapp

Cheers,
Liam.