
Subject: Re: set_plot, 'win/x'

Posted by [thompson](#) on Mon, 14 Jan 2002 15:53:14 GMT

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

Glenn Newnham <glenn.newnham@adelaide.edu.au> writes:

> I want an application to run on windows or UNIX/LINUX so I want to be
> able to automatically select between the commands

> set_plot, 'win'
> and
> set_plot, 'x'

> Is there a way detect what platform a program is running on?

> Thanks,
> Glenn

Here's something that does exactly what you want. I've also appended some general-purpose routines that it calls, most from the Astronomy User's Library.

Bill Thompson

```
;+
; Project   : SDAC
;
; Name      : SET_X
;
; Purpose   : Set the device to 'X' with vector fonts.
;
; Category  : graphics
;
; Explanation : Uses "set_plot"
;
; Use       : SET_X
;
; Inputs    : None
;
; Opt. Inputs : None
;
; Outputs   : None
;
; Opt. Outputs: None
;
; Keywords  : None
;
; Common    : None
```

```

;
; Restrictions:
;
; Side effects: !p.font set to -1
;
; Prev. Hist :
;
; Mod.      :
;   version 1, richard.schwartz@gsfc.nasa.gov 2-Mar-1996
; version 2, amy@aloha.nascom.nasa.gov 5-Mar-1998
; changed setplot,'x' to set_plot,'x'
;-
;=====
pro set_x

on_error,2

if not have_windows() then case os_family() of
'Windows': set_plot,'win'
'MacOS':   set_plot,'mac'
else: set_plot,'x'
endcase

!p.font=-1

end

FUNCTION HAVE_WINDOWS
;+
; Project   : SOHO - CDS
;
; Name      :
; HAVE_WINDOWS
; Purpose   :
; Tests whether current graphics device supports windows.
; Explanation :
; The system variable !D.FLAGS is examined to see if the current graphics
; device supports windows.
; Use       :
; Result = HAVE_WINDOWS()
;
; IF HAVE_WINDOWS() THEN ...
;
; Inputs    :
; None.
; Opt. Inputs :

```

```

; None.
; Outputs   :
; The result of the function is either 0 (false) or 1 (true) depending on
; whether or not the current graphics device supports windows.
; Opt. Outputs:
; None.
; Keywords   :
; None.
; Calls      :
; None.
; Common     :
; None.
; Restrictions:
; In general, the SERTS graphics devices routines use the special system
; variables !BCOLOR and !ASPECT. These system variables are defined in
; the procedure DEVICELIB. It is suggested that the command DEVICELIB be
; placed in the user's IDL_STARTUP file.
;
; Side effects:
; None.
; Category   :
; Utilities, Devices.
; Prev. Hist. :
; William Thompson, April 1992.
; Written    :
; William Thompson, GSFC, April 1992.
; Modified   :
; Version 1, William Thompson, GSFC, 27 April 1993.
; Incorporated into CDS library.
; Version    :
; Version 1, 27 April 1993.
;-
;
RETURN,(!D.FLAGS AND 256) NE 0
END

```

```

function OS_FAMILY, LOWER=LOWER
;+
; NAME:
; OS_FAMILY
; PURPOSE:
; Return the current operating system as in !VERSION.OS_FAMILY
;
; CALLING SEQUENCE
; result = OS_FAMILY( [ /LOWER] )
; INPUTS:

```

```

; None
; OUTPUTS:
; result - scalar string containing one of the four values
; 'Windows','MacOS','vms' or 'unix'
; OPTIONAL INPUT KEYWORD:
;   /LOWER - If set then returns lower-case strings
; NOTES:
; OS_FAMILY is assumed to be 'unix' if !VERSION.OS is not 'windows',
; 'MacOS' or 'vms'
;
; To make procedures from IDL V4.0 and later compatible with earlier
; versions of IDL, replace calls to !VERSION.OS_FAMILY with OS_FAMILY().
;
; PROCEDURES CALLED
; function TAG_EXISTS()
; REVISION HISTORY:
; Written, W. Landsman
;   Version 2, 15 May 2000, Zarro (SM&A/GSFC) - added /LOWER
;-

```

```

if tag_exist(!VERSION, 'OS_FAMILY') then begin
  os=!VERSION.OS_FAMILY
  if keyword_set(lower) then os=strlowcase(os)
  return,os
endif

```

```

case !VERSION.OS of

```

```

'windows': os= 'Windows'
'MacOS': os= 'MacOS'
'vms': os= 'vms'
else: os= 'unix'
endcase

```

```

if keyword_set(lower) then os=strlowcase(os)
return,os

```

```

end

```

```

;+
; Project   : SOHO - CDS
;
; Name      : TAG_EXIST()
;
; Purpose   : To test whether a tag name exists in a structure.

```

```

;
; Explanation : Routine obtains a list of tagnames and tests whether the
;               requested one exists or not. The search is recursive so
;               if any tag names in the structure are themselves structures
;               the search drops down to that level. (However, see the keyword
; TOP_LEVEL).
;
; Use          : IDL> status = tag_exist(str, tag)
;
; Inputs       : str - structure variable to search
;               tag - tag name to search for
;
; Opt. Inputs : None
;
; Outputs      : Function returns 1 if tag name exists or 0 if it does not.
;
; Opt. Outputs: None
;
; Keywords     : INDEX = Index of matching tag
;
; TOP_LEVEL = If set, then only the top level of the structure is
;               searched.
;
; Category     : Util, structure
;
; Written      : C D Pike, RAL, 18-May-94
;
; Modified     : Version 1.1, D Zarro, ARC/GSFC, 27-Jan-95
;               Passed out index of matching tag
; Version 2, William Thompson, GSFC, 6 March 1996
; Added keyword TOP_LEVEL
; Version 2.1, Zarro, GSFC, 1 August 1996
; Added call to help
; Version 3, Zarro, EIT/GSFC, 3 June 2000
; added check for input array structure
; Version 4, Zarro, EIT/GSFC, 23 Aug 2000
; removed calls to DATATYPE
; Version 5, Zarro, EIT/GSFC, 29 Sept 2000
; added /quiet
;-

function tag_exist, str, tag, index=index, top_level=top_level, quiet=quiet

loud=1-keyword_set(quiet)

;print,'new version'
;
; check quantity of input

```

```

;
;
if n_params() lt 2 then begin
    if loud then print,'Use: status = tag_exist(structure, tag_name)'
    return,0b
endif

;
; check quality of input
;

sz=size(str)
stype=sz(n_elements(sz)-2)
sz=size(tag)
dtype=sz(n_elements(sz)-2)
if (stype ne 8) or (dtype ne 7) then begin
    if loud then begin
        if datatype(str) ne 'STC' then help,str
        if datatype(tag) ne 'STR' then help,tag
        print,'Use: status = tag_exist(str, tag)'
        print,'str = structure variable'
        print,'tag = string variable'
    endif
    return,0b
endif

i=-1
tn = tag_names(str)
nt = where(tn eq strupcase(tag)) & index=nt(0)
if nt(0) eq -1 then begin
    status = 0b
    if not keyword_set(top_level) then begin
        for i=0,n_elements(tn)-1 do begin
            sz=size(str(0).(i))
            dtype=sz(n_elements(sz)-2)
            if dtype eq 8 then $
            status=tag_exist(str(0).(i),tag,index=index)
            if status eq 1b then return,status
        endfor
    endif
    return,0b
endif else begin
    return,1b
endelse
end

```

function datatype,var, flag0, descriptor=desc, help=hlp, Tname = tname

```

;+
; NAME:
;   DATATYPE()
;
; PURPOSE:
;   Returns the data type of a variable.
;
; EXPLANATION:
;   This routine returns the data type of a variable in a format specified
;   by the optional flag parameter.  Can also be used to emulate, in
;   earlier versions of IDL, the SIZE(/TNAME) option introduced in V5.2.
;
; CALLING SEQUENCE      :
;   Result = DATATYPE( VAR [, FLAG , /TNAME, /DESC ] )
;
; INPUTS:
;   VAR    = Variable to examine, no restrictions
;
; OPTIONAL INPUT PARAMETERS:
;   FLAG   = Integer between 0 and 3 giving the output format flag as
;           explained below.  The default is 0.
;   /DESC  = If set, then return a descriptor for the given variable.  If the
;           variable is a scalar the value is returned as a string.  If it is
;           an array a description is returned just like the HELP command
;           gives.  Ex:
;           IDL> print, datatype(fltarr(2,3,5),/desc) gives the string
;           'FLTARR(2,3,5)'
;   /TNAME = If set, then returns a identical result to the use of the /TNAME
;           keyword to the SIZE() function in IDL V5.2 and later.
;           Overrides the value of FLAG.
;   /HELP  = If set, then a short explanation is printed out.
;
; OUTPUT PARAMETERS:
;   The result of the function is the either a string or integer giving the
;   data type of VAR.  Depending on the value of FLAG or /TNAME, the result
;   will be one of the values from the following table:
;
;   FLAG = 0    FLAG = 1    FLAG = 2    FLAG = 3    /TNAME
;
;   UND        Undefined    0        UND        UNDEFINED
;   BYT        Byte         1        BYT        BYTE
;   INT        Integer      2        INT        INT
;   LON        Long         3        LON        LONG
;   FLO        Float        4        FLT        FLOAT
;   DOU        Double       5        DBL        DOUBLE
;   COM        Complex      6        COMPLEX    COMPLEX
;   STR        String       7        STR        STRING
;   STC        Structure    8        STC        STRUCT

```

```
; DCO      DComplex      9      DCOMPLEX  DCOMPLEX
; PTR      Pointer       10      PTR      POINTER
; OBJ      Object        11      OBJ      OBJREF
; UIN      UInt          12      UINT     UINT
; ULN      ULong         13      ULON     ULONG
; L64      Long64        14      LON64    LONG64
; U64      ULong64       15      ULON64   ULONG64
;
;
;
; REVISION HISTORY:
;   Original Version: R. Sterner, JHU/APL, 24 October 1985.
;   Major rewrite, add /TNAME keyword, unsigned and 64 bit datatypes
;   W. Landsman   August 1999
;   Zarro (SM&A/GSFC) - April 2000, replace error stops by continues
;-
;-----
```

```
if (N_params() lt 1) or keyword_set(hlp) then begin
  print, ' Datatype of variable as a string (3 char or spelled out).'
  print, ' typ = datatype(var, [flag])'
  print, '   var = variable to examine.      in'
  print, '   flag = output format flag (def=0). in'
  print, '   typ = datatype string or number.  out'
  print, '   flag=0  flag=1  flag=2  flag=3  /TNAME'
  print, '   UND      Undefined  0      UND      UNDEFINE'
  print, '   BYT      Byte       1      BYT      BYTE'
  print, '   INT      Integer    2      INT      INT'
  print, '   LON      Long       3      LON      LONG'
  print, '   FLO      Float      4      FLT      FLOAT'
  print, '   DOU      Double     5      DBL      DOUBLE'
  print, '   COM      Complex    6      COMPLEX  COMPLEX'
  print, '   STR      String     7      STR      STRING'
  print, '   STC      Structure  8      STC      STRUCT'
  print, '   DCO      DComplex   9      DCOMPLEX  DCOMPLEX'
  print, '   PTR      Pointer    10     PTR      POINTER'
  print, '   OBJ      Object     11     OBJ      OBJREF'
  print, '   UIN      UInt       12     UINT     UINT'
  print, '   ULO      ULong      13     ULON     ULONG'
  print, '   L64      Long64     14     LON64    LONG64'
  print, '   U64      ULong64    15     ULON64   ULONG64'
  print, ' Keywords:'
  print, ' /TNAME - Identical output to SIZE(/TNAME)
  print, ' /DESCRIPTOR returns a descriptor for the given variable.'
  print, '   If the variable is a scalar the value is returned as'
  print, '   a string. If it is an array a description is return'
  print, '   just like the HELP command gives. Ex:'
  print, '   datatype(flterr(2,3,5),/desc) gives'
```



```

    print,'    FLTARR(2,3,5) (flag always defaults to 3 for /DESC).'
    return, -1
endif

s_tname = ['UNDEFINE', 'BYTE','INT','LONG','FLOAT','DOUBLE','COMPLEX',$
          'STRING','STRUCT','DCOMPLEX','POINTER','OBJREF','UINT','ULON G', $
          'LONG64','ULONG64']

s_flag0 = ['UND','BYT','INT','LON','FLO','DOU','COM','STR','STC','DCO' ,'PTR',$
          'OBJ','UIN','ULO','L64','U64']

s_flag1 = ['Undefined','Byte','Integer','Long','Float','Double','Compl ex', $
          'String','Structure','DComplex','Pointer','Object','UInt','U Long',$
          'Long64','ULong64']

s_flag3 = [ 'UND','BYT','INT','LON','FLT','DBL','COMPLEX','STR','STC', $
          'DCOMPLEX','PTR','OBJ','UINT','ULON','LON64','ULON64']

s = size(var)
stype = s(s(0)+1)
if stype GT N_elements(s_tname) then begin
    message,'ERROR - Unrecognized IDL datatype',/cont
    stype=0
endif

if keyword_set(TNAME) then return, s_tname(stype)

if N_params() lt 2 then flag0 = 0 ; Default flag.
if keyword_set(desc) then flag0 = 3

case flag0 of

0: return, s_flag0(stype)
1: return, s_flag1(stype)
2: return, stype
3: typ = s_flag3(stype)
else: message,'ERROR - Flag parameter must be between 0 and 3'
endcase

if keyword_set(desc) then begin
    if stype EQ 0 then begin
        message,'ERROR - Input variable is undefined',/cont
        return,'Undefined'
    endif
    if s(0) eq 0 then return,strtrim(var,2) ; Return scalar desc.
    aa = typ+'ARR('
    for i = 1, s(0) do begin
        aa = aa + strtrim(s(i),2)

```

```
    if i lt s(0) then aa = aa + ','  
    endfor  
    aa = aa+','  
    return, aa  
endif else return,typ
```

```
end
```
