Subject: Re: Table widgets
Posted by Michael Asten on Tue, 12 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Nils Johnson wrote:

> : Don't know about the table widget. Don't use it. :-)
>
> So what do you use to get a bunch of strings on the screen so
> that it looks like columns of text? Is the table widget overkill
> for doing this if I just want to display data and not have it be
> modified by the user?
>

So why write a widget at all? Why not use a bit of systemware from our friendly sponsors at rsi?
The library routine XVAREDIT will display an array of strings (and anything else you can think of) in a Table widget, but the Table widget is horrible. The version of XVAREDIT from idl4.0 will do the job without the Table widget , more cleanly (as judged by one who only writes widgets as a last resort). If you dont have an old version of idl4.0, I append the version from rsi's distribution. I have renamed it XVAREDIT_idl4.pro to ensure it doesnt mix with the inferior (my view) idl 5.x version.
Copyright remains with RSI (apart from the 4 lines of main program at the end, which I doubt rsi would want to own :-)

Regards,
Michael Asten

==========

```
; $Id: xvaredit.pro,v 1.1 1993/04/02 19:54:08 idl Exp $

;
; Copyright (c) 1991-1993, Research Systems, Inc.  All rights reserved.
; Unauthorized reproduction prohibited.
;+
; NAME:
; XVAREDIT_idl4
; PURPOSE:
; This routine provides an editor for any IDL variable.
; CATEGORY:
; Widgets
; CALLING SEQUENCE:
```

```
; XVAREDIT, VAR
; INPUTS:
; VAR = The variable that is to be edited.
; KEYWORD PARAMETERS:
; NAME = The NAME of the variable.  This keyword is overwritten with the

;  structure name if the variable is a structure.
; GROUP = The widget ID of the widget that calls XVarEdit.  When this
;  ID is specified, a death of the caller results in a death of
;  XVarEdit.
; OUTPUTS:
; VAR= The variable that has been edited, or the original when the user
;  selects the "Cancel" button in the editor.
; COMMON BLOCKS:
; Xvarcom - stores the state of the variable that is being edited.
; SIDE EFFECTS:
; Initiates the XManager if it is not already running.
; RESTRICTIONS:
; If the variable is exceedingly large such as a giant structure or
; huge array, the editor will not fit on the screen and may not be able
; to create enough widget components to edit the whole variable.
; PROCEDURE:
; Create and register the widget and then exit.
; If the user selects "accept", the values in the editor are written
; to the variable passed in, otherwise, they are ignored.
; MODIFICATION HISTORY:
; Written by: Steve Richards, February, 1991
;-


 ;------------------------------------------------------------ ------------------

; procedure XVarEdit_ev
 ;------------------------------------------------------------ ------------------

; This procedure processes the events being sent by the XManager.
 ;------------------------------------------------------------ ------------------

PRO XVarEdit_ev, event

COMMON Xvarcom, thevar, initialvar, entries

WIDGET_CONTROL, event.id, GET_UVALUE = eventval  ;find the user value
      ;of the widget where
      ;the event occured
CASE eventval OF

  "DONT": BEGIN      ;the user chose the
```

```
    thevar = initialvar    ;cancel button so just
    WIDGET_CONTROL, event.top, /DESTROY  ;return the initial
   END      ;variable

  "DO": BEGIN      ;the user chose accept
         i = 0      ;so go ahead and modify
    WHILE(i LT N_ELEMENTS(entries))DO BEGIN ;the user's variable to
     IF(entries(i).type NE 6)THEN BEGIN  ;reflect his or her
       WIDGET_CONTROL, entries(i).widid, $ ;choice
   GET_VALUE = newval
       error = EXECUTE(entries(i).name  + $
   "= newval(0)")
     ENDIF ELSE BEGIN    ;when the user's
       WIDGET_CONTROL, entries(i).widid, $ ;variable has a complex
   GET_VALUE = realval  ;value, the real and
       i = i + 1     ;imaginary components
       WIDGET_CONTROL, entries(i).widid, $ ;must be reassembled
   GET_VALUE = imagval  ;from its respective
       error = EXECUTE(entries(i).name  + $ ;editable widget
    "= complex(" + $ ;components
    string(realval(0)) + $
    "," + $
    string(imagval(0)) + $
    ")")
     ENDELSE
     i = i + 1
   ENDWHILE
   WIDGET_CONTROL, event.top, /DESTROY  ;once the variables
  END      ;have been retrieved,
       ;the widget heiarchy
   ELSE:       ;can be destroyed

ENDCASE

END ;============== end of XVarEdit event handling routine task
=============



 ;-------------------------------------------------------------- -------------------

; procedure AddEditEntry
 ;-------------------------------------------------------------- -------------------

; This procedure adds an entry to the list that contains the variables
names
; and the widget id for the edit field corresponding to the variable
name.
 ;-------------------------------------------------------------- -------------------
```

```
PRO AddEditEntry, thename, thetype, thewidid

COMMON Xvarcom, thevar, initialvar, entries

IF N_ELEMENTS(thewidid) EQ 0 THEN thewidid = 0L
newelt = {entstr, name:thename, $   ;first create a record
    widid:thewidid, $   ;and then
    type:thetype}    ;just create a list
numents = N_ELEMENTS(entries)    ;with one more element
IF(NOT(KEYWORD_SET(entries)))THEN ENTRIES = newelt $ ;and replace the
old
ELSE BEGIN      ;one
  newentries = REPLICATE(newelt, numents + 1)
  newentries(0:numents - 1) = entries
  newentries(numents) = newelt
  entries = newentries
ENDELSE
END ;============== end of XVarEdit event handling routine task
==============


 ;------------------------------------------------------------ -------------------

; procedure XvarEditField
 ;------------------------------------------------------------ -------------------

;  This routine is used to create the widget or widgets needed for a
given
;  variable type.  It could call itself recursively if the variable was
itself
;  a structure comprised of other IDL variables.
 ;------------------------------------------------------------ -------------------

FUNCTION XvarEditField, base, type, val, NAME = NAME ;this is a dummy
RETURN,0      ;declaration so that
END       ;this routine can call
       ;itself recursively
FUNCTION XvarEditField, base, val, NAME = NAME, $
  RECNAME = RECNAME

dimarr = [18, 4, 7, 10, 12, 16, 12, 20]   ;an array of lengths of
       ;each type

typarr = ["Undefined", "Byte", "Integer", $  ;an array of names of
   "Longword Integer", "Floating Point", $ ;each type
   "Double Precision Floating", $
   "Complex Floating Point", $
```

```
   "String", "Structure"]

varsize = size(val)     ;determine the size and
vardims = N_ELEMENTS(varsize) - 2   ;type of the variable
type = varsize(vardims)
numelements = varsize(vardims + 1)

IF(numelements GE 5) THEN $    ;if the array is larger
  scrollval = 1 $     ;than 5 elements, use
ELSE scrollval = 0     ;a scrolling base to
       ;conserve screen space

abase = WIDGET_BASE(base, $    ;create a base for the
  /FRAME, $    ;variable to live in
  /COLUMN, $
  XPAD = 8, $
  YPAD = 8, $
  SCROLL = scrollval)

IF(numelements GT 1) THEN BEGIN    ;if the variable is an
  suffix = " Array("     ;array, then say so and
  FOR j = 1, varsize(0) DO BEGIN   ;show the array
    suffix = suffix + strtrim(varsize(j), 2)  ;dimensions.
    IF j NE varsize(0) THEN suffix = suffix + ", "
  ENDFOR
  suffix = suffix + ")"
ENDIF ELSE suffix = ""


IF(type EQ 8) THEN NAME = TAG_NAMES(val, /STRUCTURE) ;if the variable is
a
       ;structure, use its
       ;name

IF(KEYWORD_SET(NAME)) THEN $    ;build up the name of
  lbl = WIDGET_LABEL(abase, $    ;variable with the
  VALUE = NAME + $   ;type in parenthesese
   " (" + $
   typarr(type) + $
   suffix + $
   ")") $
ELSE lbl = WIDGET_LABEL(abase, $
  value = typarr(type) + suffix)


IF(NOT(KEYWORD_SET(RECNAME))) THEN RECNAME = "thevar" ;establish the
name
       ;if not being called
```

```
      ;recursively

IF(varsize(0) GT 1) THEN BEGIN
  moduli = LONARR(varsize(0)-1) + 1
  FOR i = varsize(0), 2,-1 DO BEGIN
    FOR j = 1,i-1 DO $
      moduli(i - 2) = moduli(i - 2) * varsize(j)
  ENDFOR
ENDIF

FOR element = 0, numelements - 1 DO BEGIN  ;for each array element

  IF(numelements NE 1) THEN BEGIN   ;use array subscripting
    indexname = "("     ;if variable is an
    indexname = indexname + $
  strtrim(element mod varsize(1),2)
    IF(varsize(0) GT 1) THEN BEGIN
      indexarr = lonarr(varsize(0) - 1)
      flatindex = element
      FOR i = varsize(0) - 2, 0, -1 DO BEGIN
 indexarr(i) = flatindex / moduli(i)
 flatindex = flatindex mod moduli(i)
      ENDFOR
      FOR i = 0, varsize(0) - 2 DO $
 indexname = indexname + ", " + $
  strtrim(indexarr(i), 2)
    ENDIF
    indexname = indexname + ")"
    thename = RECNAME + indexname   ;build up the name from
    thebase = WIDGET_BASE(abase, $   ;any previous recursive
  /FRAME, $    ;names
  /ROW)
    alabel = WIDGET_LABEL(thebase, $
  VALUE = indexname + ": ")
    FRAMESETTING = 0
  ENDIF ELSE BEGIN
    FRAMESETTING = 1
    thename = RECNAME
    thebase = abase
  ENDELSE

  CASE type OF      ;depending on the type,
      ;build a string
      ;variable with proper
      ;formatting

    0: thevalue = "Undefined Variable"   ;Undefined
```

```
   1: thevalue = string(val(element), $  ;Byte
  FORMAT = '(I3)')

   6: BEGIN      ;Complex Float
  rowbase = WIDGET_BASE(thebase, $  ;here the variable must
  /ROW)     ;be displayed in two
  lable = WIDGET_LABEL(rowbase, $  ;separate widgets for
  VALUE = "Real: ")   ;its real and imaginary
  id = WIDGET_TEXT(rowbase, $   ;components
  VALUE = STRING(FLOAT(val(element))), $
  FRAME = FRAMESETTING, $
  YSIZE = 1, $
  XSIZE = dimarr(type), $
  /EDITABLE, $
  UVALUE = ' ')
  AddEditEntry, thename, type, id
  lable = WIDGET_LABEL(rowbase, $
  VALUE = "Imaginary: ")
  id = WIDGET_TEXT(rowbase, $
  VALUE = STRING(IMAGINARY(val(element))), $
  FRAME = FRAMESETTING, $
  YSIZE = 1, $
  XSIZE = dimarr(type), $
  /EDITABLE, $
  UVALUE = ' ')
 AddEditEntry, thename, type, id
     END

   7: thevalue = val(element)    ;String

   8: BEGIN      ;Structure
  tags = TAG_NAMES(val(element))
  FOR i = 0, N_ELEMENTS(tags) - 1 DO BEGIN
    error = EXECUTE("fieldvalue = val(element)." + tags(i))
    fldsize = size(fieldvalue)
    flddims = N_ELEMENTS(fldsize) - 2
    id = XvarEditField(thebase, $
  fieldvalue, $
  NAME = tags(i), $
  RECNAME = thename + "." + tags(i))
  ENDFOR
 END

   ELSE: thevalue = strtrim(val(element), 2)
  ENDCASE

  IF((type NE 6) AND (type NE 8)) THEN BEGIN  ;here the actual widget
    id = WIDGET_TEXT(thebase, $    ;is created if it was
```

```
  value = thevalue, $   ;neither a structure or
  FRAME = FRAMESETTING, $   ;a complex value
  YSIZE = 1, $
  XSIZE = dimarr(type), $
  /EDITABLE, $
  UVALUE = ' ')
    AddEditEntry, thename, type, id
  END

ENDFOR

return,id

END ;============== end of XVarEdit event handling routine task
=============


 ;-------------------------------------------------------- -------------------

; procedure XVarEdit
 ;-------------------------------------------------------- -------------------

; this is the actual routine that is called.  It builds up the variable
editing
; fields by calling other support routines and then registers the widget

; heiarchy with the XManager.  Notice that the widget is registered as a
MODAL
; widget so it will desensitize all other current widgets until it is
done.
 ;-------------------------------------------------------- -------------------

PRO XVarEdit_idl4, var, GROUP = GROUP, NAME = NAME

COMMON Xvarcom, thevar, initialvar, entries

if(n_params() ne 1) THEN $
  MESSAGE, "Must have one parameter"

IF(XRegistered("XVarEdit")) THEN RETURN   ;only one instance of
       ;the XVarEdit widget
       ;is allowed.  If it is
       ;already managed, do
       ;nothing and return

XVarEditbase = WIDGET_BASE(TITLE = "XVarEdit", $ ;create the main base
  /COLUMN)
```

```
XPdMenu, [ "'Cancel"  DONT', $ ;create the menu
   "'Accept"  DO' $ ;selections
 ],     $
  XVarEditbase

initialvar = var
thevar = var
varsize = size(var)
vardims = N_ELEMENTS(varsize) - 2

ids = XvarEditField(XVarEditbase, var, NAME = NAME)

WIDGET_CONTROL, XVarEditbase, /REALIZE   ;create the widgets
      ;that are defined

XManager, "XVarEdit", XVarEditbase, $   ;register the widgets
  EVENT_HANDLER = "XVarEdit_ev", $ ;with the XManager
  GROUP_LEADER = GROUP, $   ;and pass through the
  /MODAL     ;group leader if this
      ;routine is to be
      ;called from some group
      ;leader.

entries = 0
var = thevar

END ;================== end of XVarEdit main routine
======================

;============
;  MAIN HERE
;============
sS=strarr(3)
sS=['David     Apple','Nils      Orange','Tom       Plum','Michael
Pineapple']
str={t:sS}
xvaredit_idl4,sS
end
```