
Subject: Xcd: change directory via mouse
Posted by [paulcs](#) on Tue, 08 Jul 1997 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Greetings comp.lang.idl-pvwave readers,

Here is a utility that I wrote to make changing directories require little or no typing in IDL 5.0 on the Windows95 platform. I put the whole thing in a file called xcd.pro, but it could be broken out into separate files (e.g. "xcd__define.pro" etc.).

I wrote this code on Windows95, and I expect it should work on other Windows platforms too.

I put a button for it in my idlde.

Hope you like it.

--
Paul C. Sorenson
paulcs@netcom.com

```
;+
; NAME:
;   xcd
;
; PURPOSE:
;   Change current directory via mouse.
;
;   Two lists are displayed side by side. The one on the left shows
;   directories. Click on a directory to cd there. The list
;   on the right shows files to help you see where you are.
;   (The list on the right does not respond to mouse clicks.)
; CATEGORY:
;   Utility.
; CALLING SEQUENCE:
;   xcd
; INPUTS:
;   None.
; KEYWORD PARAMETERS:
;   None
; OUTPUTS:
;   None.
; SIDE EFFECTS:
;   Your current directory can be changed.
; RESTRICTIONS:
;   Windows platforms only. Written on windows95. Should work
;   on other Windows platforms, but I haven't tried it.
```

```

;
; With a little effort, one probably could port Xcd to other platforms
; (i.e. Unix or Mac).
;
; Note that drive names (e.g. "a:", "c:", etc.) are hardcoded in xcd::init.
; Change that line of code to show drive letters appropriate for your system.
;
; PROCEDURE:
; Xcd creates an object that has a reference to a DirListing, and widgets
; for displaying that DirListing. If the user clicks on a sub-directory
; (or "..\") in the xcd object, or droplist-selects a different drive via
; the xcd object, the xcd object changes IDL's current directory to that
; location, and refreshes with a new current-directory DirListing.
;
; MODIFICATION HISTORY:
; Paul C. Sorenson, July 1997. paulcs@netcom.com.
;   Written with IDL 5.0. The object-oriented design of Xcd is based
;   in part on an example authored by Mark Rivers.
;-
function dirlisting::init, location
;
;Function DirListing::INIT: construct listing of LOCATION's contents.
;INPUT:
; LOCATION (optional): string indicating the directory we want listing of.
;   default is current directory.
;
;
catch, error_stat
if error_stat ne 0 then begin
  print, !err_string
  return, 0
end
;
;Store name of location.
;
if n_elements(location) gt 0 then $
  pushd, location
cd, current=current
self.Drive = strmid(current, 0, 2)
self.Path = strmid(current, 2, strlen(current))
;
;Obtain listing of location's contents.
;
listing = findfile()
if n_elements(location) gt 0 then $
  popd
;
;Divide into direcorey-only & file-only listings.
;

```

```

flags = bytarr(n_elements(listing))
for i=0,n_elements(listing)-1 do begin
    if rstrpos(listing[i], '\') eq (strlen(listing[i]) - 1) then $
        flags[i] = 1b
    end

dirs_indx = where(flags, dir_count)
files_indx = where(flags eq 0b, file_count)

if dir_count gt 0 then begin
    dirs = listing[dirs_indx]
    dirs = dirs[where(dirs ne '\')]
    dirs = dirs[sort(strupcase(dirs))]
end $
else begin
    dirs = ""
end

if file_count gt 0 then begin
    files = listing[files_indx]
    files = files[sort(strupcase(files))]
end $
else begin
    files = ""
end
;
;Store pointers to resulting string arrays.
;
self.pSubdirNames = ptr_new(dirs, /no_copy)
self.pFileNames = ptr_new(files, /no_copy)
return, 1 ; Success.
end
;-----
pro dirlisting::cleanup
ptr_free, self.pSubdirNames
ptr_free, self.pFileNames
end
;-----
pro dirlisting__define
void = {dirlisting, $
    Drive: "", $ ; e.g. 'c:'
    Path: "", $ ; location. e.g. '\foo\bar'
    pSubdirNames: ptr_new(), $ ; string array of sub-directory names
    pFileNames: ptr_new() $ ; string array of file names
}
end
;-----
function dirlisting::pSubdirNames

```

```

return, self.pSubdirNames
end
;-----
function dirlisting::pFileNames
return, self.pFileNames
end
;-----
function dirlisting::Path
return, self.Path
end
;-----
function dirlisting::Drive
return, self.Drive
end
;-----
pro xcd::handle, event

catch, error_stat
if error_stat ne 0 then begin
    catch, /cancel
    void = widget_message(!err_string, /error)
    self->update ; Try again, this time without "cd".
    return
end

case event.id of
    self.wDirList: begin
        path = self.rDirListing->Path()
        if rstrpos(path, '\') ne (strlen(path) - 1) then $
            path = path + '\'
        cd, self.rDirListing->Drive() $
            + path                $
            + *(self.rDirListing->pSubdirNames()))[event.index]
        self->update
        widget_control, self.tlb, /update ; workaround. Resize base.
        end
    self.wDriveList: begin
        widget_control, /hourglass
        cd, (*self.pDriveNames)[event.index]
        self->update
        widget_control, self.tlb, /update ; workaround. Resize base.
        end
    else: begin
        end
    endcase

end
;-----

```

```

pro xcd_cleanup, tlb
widget_control, tlb, get_uvalue=rXcd ; get a reference to Xcd.
obj_destroy, rXcd
end
;-----
pro xcd::cleanup
obj_destroy, self.rDirListing
ptr_free, self.pDriveNames
cd, current=current & print, current
end
;-----
pro xcd_event, event
widget_control, event.top, get_uvalue=rXcd
rXcd->handle, event
end
;-----
pro xcd::update
;
;Procedure XCD::UPDATE: set self's widgets and state values to
; reflect the current directory.
;
widget_control, /hourglass

obj_destroy, self.rDirListing
self.rDirListing = obj_new('dirlisting')

rDirListing = self.rDirListing
indx = where(strupcase(*self.pDriveNames) eq $
            strupcase(rDirListing->Drive()))
widget_control, self.wDriveList, set_droplist_select=indx(0)
widget_control, self.wLabel,    set_value=rDirListing->Path()
widget_control, self.wDirList,  set_value=*(rDirListing->pSubdirNames())
widget_control, self.wFileList, set_value=*(rDirListing->pFileNames())

end
;-----
function xcd::init

catch, error_status
if error_status ne 0 then begin
    print, !err_string
    return, 0
end

;CHANGE THESE HARDCODED DRIVENAMES TO SUIT YOUR SYSTEM.
self.pDriveNames = ptr_new(['a:', 'c:', 'd:', 'e:', 'f:'])
;
;Create widgets.

```

```

;
;
tlb = widget_base(title='xcd', /column, tlb_frame_attr=1) ; top-level base

readout_base = widget_base(tlb, /row)
self.wDriveList = widget_droplist(readout_base, value=*self.pDriveNames)
self.wLabel = widget_label(readout_base)

list_base = widget_base(tlb, /row)
ysize = 20 ; Looks good on my monitor.
self.wDirList = widget_list(list_base, ysize=ysize)
self.wFileList = widget_list(list_base, ysize=ysize)
;
;
;Set values.
;
self->update
self.tlb = tlb
widget_control, tlb, set_uvalue=self
;
;Center and realize tlb.
;
device, get_screen_size=scrsz
widget_control, tlb, map=0
widget_control, tlb, /realize
tlb_geometry = widget_info(tlb, /geometry)
widget_control, tlb, $
    tlb_set_xoffset= 0 > (scrsz(0) - tlb_geometry.scr_xsize) / 2, $
    tlb_set_yoffset= 0 > (scrsz(1) - tlb_geometry.scr_ysize) / 2
widget_control, tlb, map=1
widget_control, tlb, /update ; workaround. Resize base.
;
xmanager, 'xcd', tlb, cleanup='xcd_cleanup', /just_reg, /no_block
return, 1 ; Success.
end
;-----
pro xcd__define
void = {xcd,      $
    tlb:      0L, $      ; top-level base
    wDriveList: 0L, $      ; droplist of available drives.
    wLabel:    0L, $      ; shows name of current directory
    wDirList:  0L, $      ; shows sub-directories in current directory
    wFileList: 0L, $      ; shows files in current directory
    pDriveNames: ptr_new(),$; String array. e.g. ['c:', 'd:', etc.]
    rDirListing: obj_new() $; listing of current directory
}
end
;-----
pro xcd
;

```

on_error, 2 ; Return to caller if error.

if obj_new('xcd') eq obj_new() then \$
 message, 'failed to create xcd object.'

xmanager

end

--

Paul C. Sorenson
paulcs@netcom.com
