
Subject: Re: pickfile(), but with multiple files?
Posted by [haimov](#) on Sat, 08 Mar 1997 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <332193E9.FF6@kfa-juelich.de>, r.bauer@kfa-juelich.de says...

>
> Slacker wrote:
>>
>> Does this software exist, where I can select more than
>> one file to be stored into an array?
>
> You should use xfiles written by Ray Sterner
> <http://fermi.jhuapl.edu/s1r/idl>
> This is a starting point to his great idl Routines.

While what R. Bauer says is absolutely true and Ray's library is an invaluable addition, I believe Peter did a good job with pickfiles. I think RSI should follow on his upgrade and incorporate it in the future.

I downloaded his hacked version and modified it slightly so that it would not interfere with the standard RSI pickfile. I also spent some time to almost fix the pickfile widget display problem when

one uses keyword filter, but there are no files that match the filter string. I said almost because, I believe, there are still some widget bugs that would not allow me to fix it completely.

Attached is the pickfiles modified version.

Cheers, Sam

Samuel Haimov
Atmos. Sci Dept.
University of Wyoming
haimov@uwyo.edu

```
;+
; NAME:
;   PICKFILES
;
; PURPOSE:
;   This function allows the user to interactively pick one or more
;   file(s). A file selection tool with a graphical user interface
;   is created. Files can be selected and deselected from the
;   current directory or other directories.
;
; CATEGORY:
```

```

; Widgets.
;
; CALLING SEQUENCE:
;   Result = PICKFILES()
;
; KEYWORD PARAMETERS:
;
;   FILE: (input) A string or string array for setting the initial
;         value of the selection. Useful if there is a default file
;
;   GROUP: (input) The widget ID of the widget that calls PICKFILES.
;         When this ID is specified, a death of the caller results
;         in the death of the PICKFILES widget application.
;
;   READ: (Flag) Set this keyword to make the title of the PICKFILE
;         window "Select File(s) to Read".
;
;   WRITE: (Flag) Set this keyword to make the title of the PICKFILE
;         window "Select File(s) to Write".
;
;   PATH: (input) The initial path to select files from. If this
;         keyword is not set, the current directory is used.
;
;   FILTER: (input) A string value for filtering the files in the
;         file list. This keyword is used to reduce the number of
;         files to choose from. The user can modify the filter
;         unless the FIX_FILTER keyword is set. Example filter
;         values might be "*.pro" or "*.dat".
;
;   FIX_FILTER: (Flag) When this keyword is set, only files that
;         satisfy the filter can be selected. The user has no
;         ability to modify the filter and the filter is not shown.
;
;   TITLE: (input) A scalar string to be used for the window title.
;         If it is not specified, the default title is "Select File"
;
;   MUST_EXIST: (Flag) When set, only files that actually exist can
;         be selected.
;
;   POSITION: (input) 2-el vector with position of upper left
;         corner of the widget base, counted from top left
;         screen corner. May be ignored depending on your window
;         manager.
;
; OUTPUTS:
;   PICKFILES returns a string or string array that contains the name
;   of the file(s) selected. If no file is selected, PICKFILE
;   returns a null string.

```

```

;
; COMMON BLOCKS:
;   NEWPICKERS: COMMON block that maintains state for the widget.
;
; SIDE EFFECTS:
;   This function initiates the XMANAGER if it is not already running.
;
; RESTRICTIONS:
;   This routine is known to work on Suns (OPEN LOOK), MIPS, RS/6000,
;   DEC Ultrix, HP/700, VAX/VMS, SGI and Linux machines.
;
;   Only one instance of the PICKFILE widget can be running at one time.
;
;   PICKFILES does not recognize symbolic links to other files in UNIX.
;
; PROCEDURE:
;   Create and register the widget and then exit, returning the
;   filename(s) that were picked.
;
; EXAMPLE:
;   Create a PICKFILE widget that lets users select only files with
;   the extensions 'pro' and 'dat'. Use the 'Select File to Read' title
;   and store the name of the selected file in the variable F. Enter:
;
;       F = PICKFILES(/READ, FILTER = '*.pro *.dat')
;
; MODIFICATION HISTORY:
;   Written by:   Steve Richards, April, 1991
;   July, 1991   Added a FILTER keyword to allow users
;               to select files with a given extension or
;               extensions.
;   August, 1991 Fixed bugs caused by differences between
;               spawned ls commands on different machines.
;   September, 1991 Made Myfindfile so only one pass was
;               necessary to find files and directories.
;   3/92 - ACY    Corrected initialization of dirsave, change spawn
;               command to "ls -l" and added case for links
;               add NOCONFIRM keyword for auto exiting on selection
;   8/92 - SMR    Rewrote pickfile as a compound widget.
;   10/92 - SMR   Fixed a bug where extremely large file names didn't
;               show up properly in the file list or as return
;               values.
;   12/92 - JWG   Add better machine dependency code
;   1/93 - JWG    Added FILE, GET_PATH keywords.
;   1/93 - TAC    Added Windows Common dialog pickfile code
;   2/93 - SMR    Fixed the documentation example for multiple extensions
;   1/94 - KDB    If directory had no execute permission on Unix
;               platforms, CD fails and causes error. Added check

```


return, (not fix(result(0)))) ;convert result to int and NOT it.

END

```
;-----  
;   procedure GETDIR  
;-----  
; This routine finds the files or directories at the current directory level.  
; It must be called with either files or directories as a keyword.  
;-----
```

function getdirs

WIDGET_CONTROL, /HOUR

```
retval = ['./']  
;added -a switch to get .* dirs  
;change to /noshell, send errors to /dev/null  
SPAWN, ['/bin/sh', '-c', 'ls -laL 2> /dev/null'], /NOSHELL, results  
numfound = N_ELEMENTS(results)  
IF(KEYWORD_SET(results)) THEN BEGIN ;extension of ".dir"  
  firsts = STRUPCASE(STRMID(results, 0, 1))  
  dirs = (where(firsts EQ "D", found))  
  IF (found GT 0) THEN BEGIN  
    results = results(dirs)  
    spaceinds = WHERE(BYTE(results(0)) EQ 32)  
    spaceindex = spaceinds(N_ELEMENTS(spaceinds)-1)  
    retval = [retval, STRMID(results, spaceindex + 1, 100)]  
    ;; get rid of "." and ".." that ls -laL picks up  
    retval = retval(WHERE( (retval ne '.')and(retval ne '..')) )  
  ENDIF  
ENDIF  
RETURN, retval  
END ; function getdirs
```

```
;-----  
FUNCTION getfiles, filter
```

WIDGET_CONTROL, /HOUR

```
SPAWN, ['/bin/sh', '-c', 'ls -laL ' + filter + $  
  " 2> /dev/null"], results, /NOSHELL ;added -a to get all files
```

```
IF(KEYWORD_SET(results)) THEN BEGIN  
  firsts = STRUPCASE(STRMID(results, 0, 1))  
  fileinds = (WHERE(((firsts EQ "F") OR (firsts EQ "-") OR $
```

```

        (firsts EQ "I"), found))
IF (found GT 0) THEN BEGIN
    results = results(fileinds)
    FOR i=0, N_ELEMENTS(results) - 1 DO BEGIN
        spaceinds = WHERE(BYTE(results(i)) EQ 32)
        spaceindex = spaceinds(N_ELEMENTS(spaceinds) - 1)
        results(i) = STRMID(results(i), spaceindex + 1, 100)
    ENDFOR
    RETURN, results
ENDIF
ENDIF
RETURN, ""
END

;-----
;   procedure Pickfiles_ev
;-----
; This procedure processes the events being sent by the XManager.
;-----
PRO Pickfiles_ev, event

COMMON newpickers, pathtxt, filtxt, dirlist, filelist, sel_list, $
    ok, cancel, help, here, thefile, separator, selall, swapbutt, swap_dat

WIDGET_CONTROL, filtxt, GET_VALUE = filt
filt = filt(0)

CASE event.id OF

cancel: BEGIN
    thefile = ""
    WIDGET_CONTROL, event.top, /DESTROY
END

filtxt: BEGIN
    files = getfiles(filt)
    WIDGET_CONTROL, filelist, SET_VALUE = files
    WIDGET_CONTROL, filelist, SET_UVALUE = files
END

selall: BEGIN
    WIDGET_CONTROL, filelist, GET_UVALUE = files
    IF (KEYWORD_SET(files)) THEN BEGIN
        thefile = here + files
        WIDGET_CONTROL, sel_list, GET_UVALUE=selectlist
        IF selectlist(0) EQ " THEN BEGIN
            selectlist = thefile
        ENDIF
    ENDIF

```

```

FOR i=0, n_elements(thefile)-1 DO BEGIN
    IF max((selpos=where(strpos(selectlist, thefile(i)) $
        GE 0))) LT 0 THEN $
        selectlist = [selectlist, thefile(i)]
    ENDFOR
    WIDGET_CONTROL, sel_list, SET_VALUE=selectlist
    WIDGET_CONTROL, sel_list, SET_UVALUE=selectlist
ENDIF
END

```

```

dirlist: BEGIN
    WIDGET_CONTROL, dirlist, GET_UVALUE = directories
    IF (event.index GT N_ELEMENTS(directories) - 1) THEN RETURN

```

; Check and see if the directory is valid

```

if(not valid_dir(directories(event.index)) ) then return

```

```

IF (!version.os EQ "vms") THEN BEGIN
; Fixed logic error. If the users selects [-], the strpos/mid
; combo would return a null string. Added a check for [-],index=0

```

```

    if(event.index eq 0)then $
        found = 3 $ ; len of [-]
    else $
found = STRPOS(directories(event.index), ".", 0)

```

```

    CD, STRMID(directories(event.index), 0, found)
    CD, CURRENT = here ;get pwd

```

```

ENDIF ELSE IF !version.os EQ 'Win32' THEN BEGIN
    message,"Unsupported on this platform"
ENDIF ELSE BEGIN
    CD, directories(event.index)
    CD, CURRENT = here
    here = here + separator
ENDELSE
    WIDGET_CONTROL, pathtxt, SET_VALUE = here
    directories = getdirs()
    files = getfiles(filt)
    WIDGET_CONTROL, filelist, SET_VALUE = files
    WIDGET_CONTROL, filelist, SET_UVALUE = files
    WIDGET_CONTROL, dirlist, SET_VALUE = directories
    WIDGET_CONTROL, dirlist, SET_UVALUE = directories
END

```

```

pathtxt: BEGIN

```

```

WIDGET_CONTROL, pathtxt, GET_VALUE = newpath
newpath = newpath(0)
len = STRLEN(newpath) - 1
IF STRPOS(newpath, '/', len) NE -1 THEN $
    newpath = STRMID(newpath, 0, len)
IF (valid_dir(newpath(0))) THEN BEGIN
    here = newpath(0) + separator
    CD, here
    directories = getdirs()
    files = getfiles(filt)
    WIDGET_CONTROL, filelist, SET_VALUE = files
    WIDGET_CONTROL, filelist, SET_UVALUE = files
    WIDGET_CONTROL, dirlist, SET_VALUE = directories
    WIDGET_CONTROL, dirlist, SET_UVALUE = directories
ENDIF ELSE $
    WIDGET_CONTROL, pathtxt, SET_VALUE = here
END

```

```

filelist: BEGIN
    WIDGET_CONTROL, filelist, GET_UVALUE = files
    IF (KEYWORD_SET(files)) THEN BEGIN
        thefile = here + files(event.index)
        WIDGET_CONTROL, sel_list, GET_UVALUE=selectlist
        IF selectlist(0) EQ " THEN BEGIN
            selectlist = thefile
            WIDGET_CONTROL, sel_list, SET_VALUE=selectlist
            WIDGET_CONTROL, sel_list, SET_UVALUE=selectlist
        ENDIF
        IF max((selpos=where(strpos(selectlist, thefile) GE 0))) LT 0 THEN BEGIN
            selectlist = [selectlist, thefile]
            WIDGET_CONTROL, sel_list, SET_VALUE=selectlist
            WIDGET_CONTROL, sel_list, SET_UVALUE=selectlist
        ENDIF ELSE BEGIN
            ;; check for substrings
            selpos = selpos(where(selpos GE 0))
            FOR i=0, n_elements(selpos)-1 DO $
                IF thefile EQ selectlist(selpos(i)) THEN GOTO, have_it
            selectlist = [selectlist, thefile]
            WIDGET_CONTROL, sel_list, SET_VALUE=selectlist
            WIDGET_CONTROL, sel_list, SET_UVALUE=selectlist
        Have_it:
        ENDELSE
        WIDGET_CONTROL, ok, GET_UVALUE=auto_exit
        IF (auto_exit) THEN GOTO, checkfile
    ENDIF
END

```

ok: GOTO, checkfile


```

Sel_list: BEGIN
  WIDGET_CONTROL, sel_list, GET_UVALUE=selectlist
  IF (KEYWORD_SET(selectlist)) THEN BEGIN
    thefile = selectlist(event.index)
    nf = n_elements(selectlist)
    CASE event.index OF
      0: BEGIN
        IF nf EQ 1 THEN $
          selectlist = " $
        ELSE $
          selectlist = selectlist(1:*)
        END
      nf-1: BEGIN
        IF nf EQ 1 THEN $
          selectlist = " $
        ELSE $
          selectlist = selectlist(0:nf-2)
        END
      Else: selectlist = [ selectlist(0:event.index-1), $
                          selectlist(event.index+1:*)]
    ENDCASE
    WIDGET_CONTROL, sel_list, SET_VALUE=selectlist
    WIDGET_CONTROL, sel_list, SET_UVALUE=selectlist
  ENDIF
END

```

```

Swapbutt: BEGIN
  widget_control, swapbutt, get_val=butt_stat
  swap_dat = where(['No', 'Yes'] EQ butt_stat)
  swap_dat = swap_dat XOR 1
  widget_control, swapbutt, set_val=(['No', 'Yes'])(swap_dat(0))
END

```

```

Help: XDISPLAYFILE, "", $
  GROUP=event.top, $
  TITLE="File Selection Help", $
  WIDTH=50, $
  HEIGHT=13, $
  TEXT=["  This file selection widget lets you pick one", $
        "or more files. The files are shown on the right.", $
        "You can select a file by clicking on it with the", $
        "mouse. The selections are shown in the lower", $
        "frame. You can also deselect them there via", $
        "mouseclick. Pressing the 'OK' button will accept", $
        "the choice and the Cancel button will not. To", $
        "move into a subdirectory, click on its name in", $
        "the directory list on the left. The path can", $

```

```

"also be modified to view files from a different", $
"directory. The list of files can be modified by", $
"typing in a filter.]"

```

```

ENDCASE
RETURN

```

checkfile:

```

WIDGET_CONTROL, sel_list, GET_UVALUE = temp
WIDGET_CONTROL, cancel, GET_UVALUE = existflag
widget_control, swapbutt, get_val=butt_stat
swap_dat = where(['No', 'Yes'] EQ butt_stat)
IF existflag THEN BEGIN
    ON_IOERROR, print_error
    FOR i=0, n_elements(temp) DO BEGIN
        OPENR, unit, temp(0), /GET_LUN
        FREE_LUN, unit
    ENDFOR
ENDIF
thefile = temp
WIDGET_CONTROL, event.top, /DESTROY
RETURN

```

print_error:

```

WIDGET_CONTROL, selecttxt, SET_VALUE = "!!! Invalid File Name !!!"
thefile = ""

```

```

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

```

```

;-----
;   function Pickfiles
;-----
; This is the actual routine that creates the widget and registers it with the
; Xmanager. It also determines the operating system and sets the specific
; file designators for that operating system.
;-----
FUNCTION Pickfiles, GROUP=GROUP, PATH=PATH, READ=READ, WRITE=WRITE, $
    FILTER=FILTER, TITLE=TITLE, MUST_EXIST=MUST_EXIST, $
    FIX_FILTER=FIX_FILTER, FILE=FILE, $
    SWAP_DATA=SWAP_DATA, POSITION=pos

COMMON newpickers, pathtxt, filtxt, dirlist, filelist, sel_list, $
    ok, cancel, help, here, thefile, separator, selall, swapbutt, swap_dat

IF(XRegistered("Pickfiles")) THEN RETURN, 0

```

```

thefile = ""
existflag = 0

separator = '/'

CD, CURRENT = dirsave

IF (N_ELEMENTS(PATH) EQ 0) THEN BEGIN
  PATH = dirsave + separator
  here = PATH
ENDIF ELSE BEGIN
  IF(STRPOS(PATH, separator, STRLEN(PATH)- 1) EQ -1)$
    AND (PATH NE separator)THEN $
    PATH = PATH + separator
  CD, PATH ;if the user selected
  here = PATH ;a path then use it
ENDELSE

IF (KEYWORD_SET(NOCONFIRM)) THEN auto_exit = 1 ELSE auto_exit = 0
IF (KEYWORD_SET(MUST_EXIST)) THEN existflag = 1 ELSE existflag = 0
IF (KEYWORD_SET(FIX_FILTER)) THEN mapfilter = 0 ELSE mapfilter = 1
IF keyword_set(swap_data) THEN map_swap = 1 ELSE BEGIN
  map_swap = 0
  swap_data = 0
ENDELSE
swap_dat = swap_data

IF (N_ELEMENTS(FILE) EQ 0) THEN FILE = ""

IF (NOT (KEYWORD_SET(TITLE))) THEN $ ;build up the title
  TITLE = "Please Select the File(s)" ;based on the keywords

IF (KEYWORD_SET(READ)) THEN TITLE = TITLE + " for Reading" $
ELSE IF (KEYWORD_SET(WRITE)) THEN TITLE = TITLE + " for Writing"

IF (KEYWORD_SET(FILTER)) THEN filt = FILTER ELSE filt = ""

directories = getdirs()
dirLen=max(strlen(directories))-7 ; -7 I (SH) have no idea what is going on
files = getfiles(filt) ; but it corrects to some extent
if files(0) eq " then files=' ' ; the problem with xsize in widget_list
filesLen=max(strlen(files))-7 ; -7 like above

version = WIDGET_INFO(/VERSION)
IF (version.style EQ 'Motif') THEN osfrm = 0 ELSE osfrm = 1

Pickfilebase = WIDGET_BASE(TITLE=TITLE, /COLUMN)

```

```

;widebase = widget_label(Pickfilebase, value=title) ; is this really needed
widebase = WIDGET_BASE(Pickfilebase, /ROW)
label = WIDGET_LABEL(widebase, VALUE="Path:")
pathtxt = WIDGET_TEXT(widebase, VAL=here, /EDIT, FR=osfrm, XS=50)
filtbase = WIDGET_BASE(Pickfilebase, /ROW, MAP=mapfilter)
filtlbl = WIDGET_LABEL(filtbase, VALUE="Filter:")
filttxt = WIDGET_TEXT(filtbase, VAL=filt, /EDIT, XS=10, FR=osfrm)

```

```

selections = WIDGET_BASE(Pickfilebase, /ROW, SPACE=30)
dirs = WIDGET_BASE(selections, /COLUMN, /FRAME)
lbl = WIDGET_LABEL(dirs, VALUE="Subdirectories ")
dirlist = WIDGET_LIST(dirs, VALUE=directories, YSIZE=8, $
    XSIZE=dirlen>12, UVALUE=directories)
fls = WIDGET_BASE(selections, /COLUMN, /FRAME)
lbl = WIDGET_LABEL(fls, VALUE="Files ")
filelist = WIDGET_LIST(fls, VALUE=files, YSIZE=8, $
    XSIZE=fileslen>12, UVALUE=files)

```

```

widebase = WIDGET_BASE(Pickfilebase, /COLUMN) ; , /frame)
label = WIDGET_LABEL(widebase, VALUE="Selection(s) ")
sel_list = WIDGET_LIST(widebase, value=file, uval=file, ys=20, $
    xs=(dirlen+fileslen)>30)
rowbase = widget_base(Pickfilebase, /row, map=map_swap)
label = widget_label(rowbase, value='Byteswap Image data ?')
swapbutt = widget_button(rowbase, val=(['No', 'Yes'])(swap_dat))

```

```

rowbase = WIDGET_BASE(Pickfilebase, SPACE=20, /ROW)
ok = WIDGET_BUTTON(rowbase, VALUE="Ok", xsize=90, $
    UVALUE=auto_exit)
selall = WIDGET_BUTTON(rowbase, VALUE="Select all", xsize=90)
cancel = WIDGET_BUTTON(rowbase, VALUE="Cancel", xsize=90, $
    UVALUE=existflag)
help = WIDGET_BUTTON(rowbase, VALUE="Help", xsize=90)

```

```

IF keyword_set(pos) THEN $
    widget_control, Pickfilebase, xoff=pos(0), yoff=pos(1)

```

```

WIDGET_CONTROL, Pickfilebase, /REALIZE

```

```

XManager, "Pickfiles", Pickfilebase, EVENT_HANDLER="Pickfiles_ev", $
    GROUP_LEADER=GROUP, /MODAL

```

```

CD, dirsave
filt = ""
swap_data = swap_dat(0)

```

```

RETURN, thefile

```

END ;===== end of Pickfile routine =====
