Subject: Re: reading a series of image files
Posted by Matthew Larkum on Thu, 01 Aug 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Salvador Borges wrote:
>
> I would like to read a series of images from a file folder, wth
> filenames that serially increment like a01,a02 etc or increment with
> odd/even numbers like a01, a03 etc. and do a stack average.
>
> Is there an easy and efficient way to do this?
>
> Thanks,
> Salvador Borges

I had to do a similar thing recently and ended up writing my own routines
to do it. It might look like overkill to you, but I've just put the
string handling routines below in case it helps.

Once you have these routines compiled you should be able to write
something like:

```
function get_average, firstnum, lastnum, increment, xsize, ysize
average = fltarr(xsize,ysize)   ;*** xsize, ysize = size of image
name = getfilename()    ;*** user clicks on one of the images
     ;*** from the file dialog.  This
     ;*** doesn't have to be the first one.
filestem = name.namestem
filesuffix = name.suffix
for i=firstnum,lastnum,increment do begin
 filename = filestem + strtrim(i,2) + filesuffix
 average = average + tiff_read(filename) ;*** depending on what sort of image you have
 endfor
return, byte(average/((lastnum-firstnum)/increment))
end
```

anyway, here are the helper routines that work for me (no guaranties):

Matthew.

```
---------------------------- cut here -----------------------------
;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: STRPOSC
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Saturday, 13th April, 1996
; PURPOSE: Like STRPOS but returns the positions of all characters in a string
```

```
; CATEGORY: String manipulation
; CALLING SEQUENCE: result = STRPOSC(string,char)
; INPUTS: string = string input
; char = character to find
; OPTIONAL INPUT PARAMETERS: first = only return position of first char
; last = only return position of last char
; KEYWORD PARAMETERS: None.
; OUTPUTS: Returns the positions of all matching characters
; OPTIONAL OUTPUT PARAMETERS: None.
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; PROCEDURE:
; MODIFICATION HISTORY:
;-

function strposc, instring, char, first=first, last=last

char = byte(char)
if keyword_set(first) then return, (where(byte(instring) eq char(0)))(0) $
else if keyword_set(last) then begin
 all = where(byte(instring) eq char(0))
 return, all(n_elements(all)-1)
 endif $
else return, where(byte(instring) eq char(0))
end

;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: ISDIGIT
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Friday, 2nd February, 1996.
; PURPOSE: Check if the string argument is a digit
; CATEGORY: Math routine
; CALLING SEQUENCE: result = ISNUM(n)
; INPUTS: n = string argument
; OPTIONAL INPUT PARAMETERS: None.
; KEYWORD PARAMETERS:
; ignore_whitespace = returns 1 for white space characters
;  This can be useful if you want to distinguish
;  between text and number input since the STRING
;  value of a number often has white space characters.
; OUTPUTS: 0 = argument is not a digit
; 1 = argument is a digit
; NB: the argument can be an array
; OPTIONAL OUTPUT PARAMETERS: None.
```

```
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; PROCEDURE:
; MODIFICATION HISTORY:
;-

function isdigit, n, ignore_whitespace=ignore_whitespace

nb = byte(n)
if keyword_set(ignore_whitespace) then $
 nb(where((nb eq 32) or (nb eq 9))) = 48

return, (nb ge 48) and (nb le 57)
end

;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: TYPE
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Thursday, 30th November, 1995.
; PURPOSE: Returns the type of a variable
; CATEGORY: General
; CALLING SEQUENCE: t = TYPE(invar)
; INPUTS: invar = any input variable
; OPTIONAL INPUT PARAMETERS: None.
; KEYWORD PARAMETERS: None.
; OUTPUTS: string describing the type of the variable
; OPTIONAL OUTPUT PARAMETERS: None.
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; PROCEDURE:
; MODIFICATION HISTORY:
;-

function type, invar

;*** get a description of the data
a = size(invar)
;*** get the 2nd last element which is the type as a number
t = (reverse(a))(1)
;*** return the appropriate string
case t of
 0: return, "undefined"
 1: return, "byte"
```

```
 2: return, "integer"
 3: return, "long"
 4: return, "floating"
 5: return, "double"
 7: return, "string"
 8: return, "structure"
 else: return, "don't know"
 endcase
end

;
; $Id: template,v 1.1 91/03/11 20:30:47 jeffry Exp $
;
;+
; NAME: GETFILENAME
; WRITTEN BY: Matthew Larkum, University of Bern, Switzerland.
; DATE: Wednesday, 10th April, 1996
; PURPOSE: Returns a structure containing the parts of the filename
; CATEGORY: File handling
; CALLING SEQUENCE: result = getfilename(filename)
; INPUTS: filename = string containing filename with path
; OPTIONAL INPUT PARAMETERS: None.
; KEYWORD PARAMETERS: None.
; OUTPUTS: Returns a structure containing:
; name = file name without path
; path = file path without name if it exists, otherwise empty string
; suffix = file suffix if it exists, otherwise empty string
; number = number extracted from file name if there is one
;  other wise number = -1
; version = version number (VMS thing) if it exists otherwise -1
; filestem = file name with path but without suffix, number or version
;  useful for incrementing the number of a file
; OPTIONAL OUTPUT PARAMETERS: None.
; COMMON BLOCKS: None.
; SIDE EFFECTS: None.
; RESTRICTIONS: None.
; PROCEDURE:
; name = "H:\WINSHARE\PAPER2\FIGURES\DRAFT\D0326-10.TIF"
; file = getfilename(name)
; help, file, /struct
; gives:
; ** Structure <9884bc>, 6 tags, length=38, refs=1:
;    NAME            STRING    'D0326-10.TIF'
;    PATH            STRING    'H:\WINSHARE\PAPER2\FIGURES\DRAFT'
;    NUMBER          LONG              10
;    VERSION         INT            -1
;    NAMESTEM        STRING    'H:\WINSHARE\PAPER2\FIGURES\DRAFT\D0326-'
;    SUFFIX          STRING    'TIF'
```

```
;
; NOTE:
; There exists a perfectly good function FILEPATH for creating
; operating system independent names.  Consider using the FILEPATH
; function wherever possible because this will give safer performance
; on all systems.  FILEPATH is just a bit limited for all occasions,
; though.
; MODIFICATION HISTORY:
;-

function get_suffix, name
;*** find the position of all the dots
dotpos = strposc(name,".")
;*** since VMS uses dots to separate directories we
;*** need to find the last one
dotpos = (reverse(dotpos))(0)
;*** set the version to -1 by default to indicate that the
;*** file doesn't have a version
version = -1
;*** set number to -1 by default for same reason
number = -1
;*** if it doesn't find any dots return an empty string
if dotpos(0) eq -1 then return, {name:name,suffix:"",version:version} $
else begin ;*** else return the suffix part
 suffix = strmid(name,dotpos+1,strlen(name)-dotpos)
 ;*** but first take off that nasty version number if
 ;*** using VMS
 semicolonpos = strpos(suffix,";")
 if semicolonpos ne -1 then begin
  version =  fix(strmid(suffix,semicolonpos+1,strlen(suffix)-semicolonpos ))
  suffix = strmid(suffix,0,semicolonpos)
  endif
 return, {name:strmid(name,0,dotpos),suffix:suffix,version:version}
 endelse
end

function get_number_str, name
;*** this function tries to figure out the number of this file

;*** first make an array containing the boundaries of all numbers in the string
;*** here 255 means the letter before the start of a number and 1 means
;*** the position of the last digit of a number
num_ends = isdigit(name) - shift(isdigit(name),-1)

;*** if there are no numbers return -1
if (where(num_ends eq 255))(0) eq -1 then return, {name:name,number:-1}

;*** the number we want is the last one so find the position after last 255
```

```
numstart = (reverse(where(num_ends eq 255)))(0) + 1
;*** the end of this number is the next 1
numend = (where((num_ends eq 1) and (num_ends ge numstart)))(0)

return,  {name:strmid(name,0,numstart),number:long(strmid(name,numsta rt,numstart-numend))}
end

function getfilename, filename, filenumber=filenumber

if !version.os eq 'windows' then delimiter = '\' $
else if !version.os eq 'Win32' then delimiter = '\' $
else if !version.os eq 'linux' then delimiter = '/' $
else if !version.os eq 'vms' then delimiter = ']' $
else message, "not implemented for this operating system"

if n_elements(filename) eq 0 then filename = pickfile()

if type(filename) ne "string" then return, -1
if filename eq '' then return, -1

s = get_suffix(filename)
n = get_number_str(s.name)
if strpos(filename,delimiter) eq -1 then $
 return, {name:filename,path:"",number:n.number,version:s.version,$
  namestem:n.name,suffix:s.suffix}
namestart = strlen(filename) - strpos(string(reverse(byte(filename))),delimiter)
if !version.os eq 'vms' then begin
 nameend = (reverse(strposc(filename,';')))(0)
 if nameend(0) eq -1 then nameend = strlen(filename)
 pathend = namestart
 endif $
else begin
 nameend = strlen(filename)
 pathend = namestart-1
 end
name = strmid(filename,namestart,nameend)
path = strmid(filename,0,pathend)
return, {name:name,path:path,number:n.number,version:s.version,$
 namestem:n.name,suffix:s.suffix}
end
```