
Subject: Re: analyze reader and writer in IDL?
Posted by [G Karas](#) on Wed, 03 Apr 2002 18:21:19 GMT
[View Forum Message](#) <> [Reply to Message](#)

ermmm, found some code somewhere on the web..
cleaned it up and fixed some bugs..
uses the excellent endian() routine by Stern :)
the guy who did the original code: please contact so
I can put you in the (c) :)

still uses point_lun, i am not good at assoc() yet :p

here it comes:

```
FUNCTION IMPORT_ANALYZE, volume_name

; $Id: import_analyze.pro,v 1.3 2001/12/22  gk Exp $
;
; Copyright (c) 2001-2002, GB Karas  All rights reserved.
;   Unauthorized reproduction prohibited.
;+
; NAME: import_analyze
;
; PURPOSE: This function should import an analyze volume
;   and return a structure with the image and all
;   the header details. The function returns a
;   structure with all the necessary details needed
;   to work with an ANALYZE volume
;
; MAJOR TOPICS: VBM
;
; CALLING SEQUENCE: import_analyze(filename)
;   * filename can be either *.img or *.hdr
;   procedure converts to correct extension
;
; PROCEDURE: import_analyze
;
; CALLED MAJOR FUNCTIONS and PROCEDURES: endian()
;
; EXPLANATION: just an input utility
;
; --> ANALYZE HEADER positions
;
; Description of an ANALYZE header file for reference
;
; this is how the header looks (note: int == long in IDL!)
```

```

; byte
; 0  int      sizeof_hdr; /* For ANALYZE compatibility only
*/
; 4  char      pad1[28];
; 32 int      extents; /* For ANALYZE compatibility only
*/
; 36 char      pad2[2];
; 38 char      regular; /* For ANALYZE compatibility only
*/
; 39 char      pad3;
; 40 short int  ndim; /* For ANALYZE compatibility only
*/
; 42 short int  dims[7]; /* AIR */
; 56 char      pad4[14];
; 70 short int  datatype; /* For ANALYZE compatibility only
*/
; 72 short int  bits; /* AIR */
; 74 char      pad5[6];
; 80 float      pixel_sizes[7]; /* AIR */
; 108 float     scale_offset; /* SPM */
; 112 float     scale_factor; /* SPM */
; 116 char      pad6[24];
; 140 int        glmax; /* AIR */
; 144 int        glmin; /* AIR */
; 148 char      descrip[80]; /* AIR (non-essential) */
; 228 char      pad7[??];
; 254 short int  origin[3]; /* SPM */
; 260 char      pad8[??];
; 348
;
;
;
; ANALYZE Structure FORMAT:

```

```

; analyze_vol = {
;     vol_name: vol_name, $
;     hdr_name: hdr_name, $
;     ndims : n, $
;     x_dim : x_dim, $
;     y_dim : y_dim, $
;     z_dim : z_dim, $
;     spmtype : spm_type, $
;     idl_type: type, $
;     x_vox : x_vox, $
;     y_vox : y_vox, $
;     z_vox : z_vox, $
;     m_sc : m, $
;     b_sc : b, $
;     sp_org : origin, $

```

```

;   data   : raw_volume $
; }
;
;
; Creation Date: 2001-08-31
; by Giorgos Karas (c)
;
;
; Modification history
;
;
; 2001-09-02: Added an endian check to load files with either big or
;             little endian byte-order. This is accomplished by checking
;             if number 348 is present in the first 4 bytes of the
;             header or not. (GBK)
;
;
; 2001-10-22: Fixed the routine to work cross platfrom. Trick was to check
the
;             header file for 3000L and see if it is more than that and then
;             compare that to os version. (GBK)
;
;
; 2001-11-12: Fixed the structure returned as anonymous structure. This way
;             conflicts were eliminated when opening multiple files. GBK
;
;
; 2001-12-22: Fixed the procedure to read in Analyze 32bit volumes. In the
case
;             statement changed the type 32->6 to 32->5. Read IDL Docus for
;             various datatypes. Short copy-pasting here:
;   Type Code Data Type
;   0 Undefined
;   1 Byte
;   2 Integer (16-bit)
;   3 Longword integer (32-bit)
;   4 Floating point
;   5 Double-precision floating
;   6 Complex floating
;   7 String
;   8 Structure
;   9 Double-precision complex floating
;  10 Pointer
;  11 Object reference
;  12 Unsigned integer (16-bit)
;  13 Unsigned longword integer (32-bit)
;  14 64-bit integer
;  15 Unsigned 64-bit integer
;
;
;   (GBK)
;
;
; 2002-02-27: Added abblity to read AIR compatible files. These files have
;             an integer on position 72 of the file. Bug fixed. (GBK)
;
;

```

```
; 2002-03-04: Used the endian() procedure ability to check for endian. 0 is
;             for little endian and 1 for big endian.(GBK)
;
;
; 2002-03-08: Improved the endian check. Now system reports system status
;             and file status (big or little endian). Added testing
;             for files not in .hdr + .img format (GBK)
;
;
; Bugs:
;
```

```
; check if a value has been passed to the function
```

```
if N_ELEMENTS(volume_name) EQ 0 then begin
    print, 'import_analyze.pro error: No volume name set. Please choose a
volume'
    return, 0
endif
```

```
; extract the volume name, and create the header and raw data names
```

```
fname_pos = STRPOS(volume_name, '.', /REVERSE_SEARCH)
raw_name = STRMID(volume_name, 0, fname_pos)
vol_name = raw_name + '.img'
hdr_name = raw_name + '.hdr'
```

```
; test if both files exist
if (file_test(vol_name) EQ 0) then begin
    print, 'img data file not found. Exiting...'
    return, -1
endif
```

```
if (file_test(hdr_name) EQ 0) then begin
    print, 'Header file not found. Exiting...'
    return, -1
endif
```

```
print, 'Importing.. ', raw_name
```

```
; Check if platform is little Endian and set an appropriate flag
```

```

;
;
if (endian() EQ 0) then begin
  os_flag = 'little'
  print, 'Platform is little endian'
endif else begin
  os_flag = 'big'
  print, 'Platform is big endian'
endelse

; Read in the header file. Also check if it is possible to read the header
file

openr, luna, hdr_name, /get_lun, err=err

if (err ne 0) then begin
  print, 'The .hdr file could not be read. Please check your data'
endif

; Check if it is a Big or Little Endian ANALYZE header file

endian_val=3000L
readu, luna, endian_val

if endian_val GT 3000L then begin
  free_lun, luna
  endian_flag = 'swaped'
  openr, luna, hdr_name, /get_lun, err=err, /SWAP_ENDIAN

  if (os_flag EQ 'big') then begin
    print, 'Header is little endian'
  endif else begin
    print, 'Header is big endian'
  endelse

  print, 'ENDIAN SWAPPED'

endif else begin
  endian_flag = 'native'

  if (os_flag EQ 'big') then begin
    print, 'Header is also big endian'
  endif else begin
    print, 'Header is also little endian'
  endelse
endif

```

```

endelse

print, 'ENDIAN KEPT'

endelse

; Recalculate the header size ; correct this time
point_lun, luna, 0
endian_val=348L
readu, luna, endian_val

; number of dimensions
n=1000
point_lun, luna, 40
readu, luna, n
;n = n - 1 ; 4 dims means 3 dims and the intensities (4 dims)

; Dimension x,y,z
dims = intarr(4)
readu,luna, dims

x_dim = dims(0)
y_dim = dims(1)
z_dim = dims(2)

; data type and convert it to IDL datatype!
; Fix it here to read 32bit images as well
; Also the little field next to it

type = 1000
point_lun,luna, 70
readu, luna, type
spm_type = type

case type of
2: type=1
4: type=2
8: type=3
16: type=4
32: type=5 ; was 6 - bug fixed
64: type=5

```

```
else: begin
  tmp=widget_message("Data type not recognized. Trying 'byte'.")
  type=1
end
endcase
```

```
; AIR.TYPE
airtype = 16
point_lun, luna, 72
readu, luna, airtype
```

```
; voxel sizes
voxelsize = fltarr(3)
point_lun, luna, 80
readu, luna, voxelsize
```

```
x_vox = voxelsize[0]
y_vox = voxelsize[1]
z_vox = voxelsize[2]
```

```
; scaling
; check this one again
m = 1.0
b = 0.0
point_lun, luna, 108
readu, luna, b
readu, luna, m
```

```
if (m eq 0) then m = 1
```

```
; spatial origin
; Needed by SPM
origin = fix([0,0,0])
point_lun,luna, 253
readu, luna, origin
```

```
free_lun, luna
```

```
; Read in volume - check for endian structure
; If the header is ok, then import as is
; if header not ok, and we are in little endian import a big endian
```

```
; and if we are in big endian import a little endian =)
```

```
if endian_flag EQ 'native' then begin
```

```
  raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,  
y_dim, z_dim], endian = 'native')
```

```
endif else begin
```

```
  if os_flag EQ 'little' then begin
```

```
    raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,  
y_dim, z_dim], endian = 'big')
```

```
  endif else begin
```

```
    raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,  
y_dim, z_dim], endian = 'little')
```

```
  endelse
```

```
endelse
```

```
; Put everything in a structure and return it to the main procedure
```

```
;analyze_vol = {VOL_INFO, $
```

```
analyze_vol = {
```

```
  vol_name: vol_name, $
```

```
  hdr_name: hdr_name, $
```

```
  ndims : n, $
```

```
  x_dim : x_dim, $
```

```
  y_dim : y_dim, $
```

```
  z_dim : z_dim, $
```

```
  spmtype : spm_type, $
```

```
  idl_type: type, $
```

```
  air_type: airtype, $
```

```
  x_vox : x_vox, $
```

```
  y_vox : y_vox, $
```

```
  z_vox : z_vox, $
```

```
  m_sc : m, $
```

```
  b_sc : b, $
```

```
  sp_org : origin, $
```

```
  data : raw_volume $
```

```
}
```

```
return, analyze_vol
```

```
END
```

```
;-----
```

```

;+
; NAME:
;   ENDIAN
; PURPOSE:
;   Function indicating which endian the current machine uses.
; CATEGORY:
; CALLING SEQUENCE:
;   f = endian()
; INPUTS:
; KEYWORD PARAMETERS:
;   Keywords:
;     /LIST means list result to screen.
;     /TEXT means return /LIST text.
; OUTPUTS:
;   f = 0 if little, 1 if big.    out
; COMMON BLOCKS:
; NOTES:
;   Note: this is the order the bytes are for multibyte
;   numeric values. Use the IDL procedure BYTEORDER
;   to switch endian (use /LSWAP for 4 byte integers,
;   /L64SWAP for 8 byte integers).
; MODIFICATION HISTORY:
;   R. Sterner, 1999 Dec 13
;   R. Sterner, 2000 Apr 11 --- Added /TEXT
;
; Copyright (C) 1999, Johns Hopkins University/Applied Physics Laboratory
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made. This
; routine is provided as is without any express or implied warranties
; whatsoever. Other limitations apply as described in the file
; disclaimer.txt.
;-
;----- --
function endian, list=list, text=text, help=help

if keyword_set(help) then begin
  print, ' Function indicating which endian the current machine uses.'
  print, ' f = endian()'
  print, '  No args.'
  print, '  f = 0 if little, 1 if big.    out'
  print, ' Keywords:'
  print, '  /LIST means list result to screen.'
  print, '  /TEXT means return /LIST text.'
  print, ' Note: this is the order the bytes are for multibyte'
  print, '  numeric values. Use the IDL procedure BYTEORDER'
  print, ' to switch endian (use /LSWAP for 4 byte integers,'
  print, ' /L64SWAP for 8 byte integers).'
  return,"

```

```

if fix([0B,1B],0) eq 1 then f=1 else f=0
if (not keyword_set(list)) and (not keyword_set(text)) then return, f

```

```
h = getenv('HOST')
txt = h+' is '+([ 'little', 'big' ])(f)+' endian'
if keyword_set(list) then begin
  print, ' '
  print, ' '+txt+'.'
endif
if keyword_set(text) then f=txt
return, f
```

end

"Michael A. Miller" <mmiller3@iupui.edu> wrote in message
news:87zo0kstnh.fsf_-_@lumen.indyrad.iupui.edu...
> Can anyone point me to IDL codes for reading and writing analyze
> format images as defined be Mayo's Analyze and discussed at
> <http://www.dclunie.com/medical-image-faq/html/part5.html> ?
>
> I've got some beginning of my own reader/writer. If there isn't
> anything else out there, I'll clean it up and eventually make it
> available.
>
> Mike
>
> P.S. Thanks for all the followups to my question about getting
> access to command line arguments.
>
> --
> Michael A. Miller mmiller3@iupui.edu
> Imaging Sciences, Department of Radiology, IU School of Medicine

Subject: Re: analyze reader and writer in IDL?
Posted by [G Karas](#) on Wed, 03 Apr 2002 19:00:34 GMT
[View Forum Message](#) <> [Reply to Message](#)

oh, i have a writer as well.. in beta stage
principles are the same like the reader..
will post it after i clean it up a little :)

njoy!

GB Karas, M.D.
Ph.D. Student

Department of Radiology
Vrije Universiteit Medical Center
Amsterdam
The Netherlands

jacobian_@_gmx.net (remove underscore)

"G Karas" <jacobianat@gmx.net> wrote in message
news:a8fh70\$226q\$1@scavenger.euro.net...

> ermmm, found some code somewhere on the web..
> cleaned it up and fixed some bugs..
> uses the excellent endian() routine by Stern :)
> the guy who did the original code: please contact so
> I can put you in the (c) :)

>
> still uses point_lun, i am not good at assoc() yet :p
>
> here it comes:

>
> FUNCTION IMPORT_ANALYZE, volume_name
>
>
>
> ; \$Id: import_analyze.pro,v 1.3 2001/12/22 gk Exp \$
> ;
> ; Copyright (c) 2001-2002, GB Karas All rights reserved.
> ; Unauthorized reproduction prohibited.
> ;+
> ; NAME: import_analyze
> ;
> ; PURPOSE: This function should import an analyze volume
> ; and return a structure with the image and all
> ; the header details. The function returns a
> ; structure with all the necessary details needed
> ; to work with an ANALYZE volume
> ;
> ; MAJOR TOPICS: VBM
> ;
> ; CALLING SEQUENCE: import_analyze(filename)
> ; * filename can be either *.img or *.hdr
> ; procedure converts to correct extension
> ;
> ; PROCEDURE: import_analyze
> ;
> ; CALLED MAJOR FUNCTIONS and PROCEDURES: endian()
> ;
> ; EXPLANATION: just an input utility
> ;
> ; --> ANALYZE HEADER positions
> ;

```

> ; Description of an ANALYZE header file for reference
> ;
> ; this is how the header looks (note: int == long in IDL!)
> ; byte
> ; 0   int      sizeof_hdr;   /* For ANALYZE compatibility only
> */
> ; 4   char      pad1[28];
> ; 32  int      extents;      /* For ANALYZE compatibility only
> */
> ; 36  char      pad2[2];
> ; 38  char      regular;      /* For ANALYZE compatibility only
> */
> ; 39  char      pad3;
> ; 40  short int  ndim;         /* For ANALYZE compatibility only
> */
> ; 42  short int  dims[7];      /* AIR */
> ; 56  char      pad4[14];
> ; 70  short int  datatype;     /* For ANALYZE compatibility only
> */
> ; 72  short int  bits;         /* AIR */
> ; 74  char      pad5[6];
> ; 80  float      pixel_sizes[7]; /* AIR */
> ; 108 float scale_offset; /* SPM */
> ; 112 float scale_factor; /* SPM */
> ; 116 char pad6[24];
> ; 140 int        glmax;        /* AIR */
> ; 144 int        glmin;       /* AIR */
> ; 148 char      descrip[80];   /* AIR (non-essential) */
> ; 228 char      pad7[??];
> ; 254 short int  origin[3]; /* SPM */
> ; 260 char pad8[??];
> ; 348
> ;
> ;
> ; ANALYZE Structure FORMAT:
>
> ; analyze_vol = {                $
> ;   vol_name: vol_name,  $
> ;   hdr_name: hdr_name,  $
> ;   ndims   : n,         $
> ;   x_dim   : x_dim,     $
> ;   y_dim   : y_dim,     $
> ;   z_dim   : z_dim,     $
> ;   spmtype : spm_type,  $
> ;   idl_type: type,      $
> ;   x_vox   : x_vox,     $
> ;   y_vox   : y_vox,     $
> ;   z_vox   : z_vox,     $

```

```

> ; m_sc : m, $
> ; b_sc : b, $
> ; sp_org : origin, $
> ; data : raw_volume $
> ; }
> ;
> ; Creation Date: 2001-08-31
> ; by Giorgos Karas (c)
> ;
> ; Modification history
> ;
> ; 2001-09-02: Added an endian check to load files with either big or
> ; little endian byte-order. This is accomplished by checking
> ; if number 348 is present in the first 4 bytes of the
> ; header or not. (GBK)
> ;
> ; 2001-10-22: Fixed the routine to work cross platform. Trick was to check
> the
> ; header file for 3000L and see if it is more than that and
then
> ; compare that to os version. (GBK)
> ;
> ; 2001-11-12: Fixed the structure returned as anonymous structure. This
way
> ; conflicts were eliminated when opening multiple files. GBK
> ;
> ; 2001-12-22: Fixed the procedure to read in Analyze 32bit volumes. In the
> case
> ; statement changed the type 32->6 to 32->5. Read IDL Docus
for
> ; various datatypes. Short copy-pasting here:
> ; Type Code Data Type
> ; 0 Undefined
> ; 1 Byte
> ; 2 Integer (16-bit)
> ; 3 Longword integer (32-bit)
> ; 4 Floating point
> ; 5 Double-precision floating
> ; 6 Complex floating
> ; 7 String
> ; 8 Structure
> ; 9 Double-precision complex floating
> ; 10 Pointer
> ; 11 Object reference
> ; 12 Unsigned integer (16-bit)
> ; 13 Unsigned longword integer (32-bit)
> ; 14 64-bit integer
> ; 15 Unsigned 64-bit integer

```

```

> ;
> ;      (GBK)
> ;
> ; 2002-02-27: Added ability to read AIR compatible files. These files
have
> ;      an integer on position 72 of the file. Bug fixed. (GBK)
> ;
> ; 2002-03-04: Used the endian() procedure ability to check for endian. 0
is
> ;      for little endian and 1 for big endian.(GBK)
> ;
> ; 2002-03-08: Improved the endian check. Now system reports system status
> ;      and file status (big or little endian). Added testing
> ;      for files not in .hdr + .img format (GBK)
> ;
> ; Bugs:
> ;
>
>
>
>
> ; check if a value has been passed to the function
>
> if N_ELEMENTS(volume_name) EQ 0 then begin
>   print, 'import_analyze.pro error: No volume name set. Please choose a
>   volume'
>   return, 0
> endif
>
>
>
> ; extract the volume name, and create the header and raw data names
>
> fname_pos = STRPOS(volume_name, '.', /REVERSE_SEARCH)
> raw_name = STRMID(volume_name, 0, fname_pos)
> vol_name = raw_name + '.img'
> hdr_name = raw_name + '.hdr'
>
> ; test if both files exist
> if (file_test(vol_name) EQ 0) then begin
>   print, 'img data file not found. Exiting...'
>   return, -1
> endif
>
> if (file_test(hdr_name) EQ 0) then begin
>   print, 'Header file not found. Exiting...'
>   return, -1
> endif

```

```

>
>
>
> print, 'Importing.. ',raw_name
>
>
>
> ; Check if platform is little Endian and set an appropriate flag
> ;
> if (endian() EQ 0) then begin
>   os_flag = 'little'
>   print, 'Platform is little endian'
> endif else begin
>   os_flag = 'big'
>   print, 'Platform is big endian'
> endelse
>
>
>
> ; Read in the header file. Also check if it is possible to read the header
> file
>
> openr, luna, hdr_name, /get_lun, err=err
>
> if (err ne 0) then begin
>   print, 'The .hdr file could not be read. Please check your data'
> endif
>
>
>
> ; Check if it is a Big or Little Endian ANALYZE header file
>
> endian_val=3000L
> readu, luna, endian_val
>
> if endian_val GT 3000L then begin
>   free_lun, luna
>   endian_flag = 'swaped'
>   openr, luna, hdr_name, /get_lun, err=err, /SWAP_ENDIAN
>
>
>   if (os_flag EQ 'big') then begin
>     print, 'Header is little endian'
>   endif else begin
>     print, 'Header is big endian'
>   endelse
>
>   print, 'ENDIAN SWAPPED'

```

```

>
> endif else begin
>   endian_flag = 'native'
>
>   if (os_flag EQ 'big') then begin
>     print, 'Header is also big endian'
>   endif else begin
>     print, 'Header is also little endian'
>   endelse
>
>   print, 'ENDIAN KEPT'
>
> endelse
>
>
>
> ; Recalculate the header size ; correct this time
> point_lun, luna, 0
> endian_val=348L
> readu, luna, endian_val
>
>
>
> ; number of dimensions
> n=1000
> point_lun, luna, 40
> readu, luna, n
> ;n = n - 1 ; 4 dims means 3 dims and the intensities (4 dims)
>
>
> ; Dimension x,y,z
> dims = intarr(4)
> readu,luna, dims
>
> x_dim = dims(0)
> y_dim = dims(1)
> z_dim = dims(2)
>
>
>
> ; data type and convert it to IDL datatype!
> ; Fix it here to read 32bit images as well
> ; Also the little field next to it
>
> type = 1000
> point_lun,luna, 70
> readu, luna, type
> spm_type = type

```

```

>
> case type of
> 2: type=1
> 4: type=2
> 8: type=3
> 16: type=4
> 32: type=5 ; was 6 - bug fixed
> 64: type=5
> else: begin
>   tmp=widget_message("Data type not recognized. Trying 'byte'.")
>   type=1
>   end
> endcase
>
>
> ; AIR.TYPE
> airtype = 16
> point_lun, luna, 72
> readu, luna, airtype
>
>
> ; voxel sizes
> voxelsize = fltarr(3)
> point_lun, luna, 80
> readu, luna, voxelsize
>
> x_vox = voxelsize[0]
> y_vox = voxelsize[1]
> z_vox = voxelsize[2]
>
>
> ; scaling
> ; check this one again
> m = 1.0
> b = 0.0
> point_lun, luna, 108
> readu, luna, b
> readu, luna, m
>
> if (m eq 0) then m = 1
>
>
>
> ; spatial origin
> ; Needed by SPM
> origin = fix([0,0,0])
> point_lun,luna, 253
> readu, luna, origin

```

```

>
> free_lun, luna
>
>
>
> ; Read in volume - check for endian structure
> ; If the header is ok, then import as is
> ; if header not ok, and we are in little endian import a big endian
> ; and if we are in big endian import a little endian =>)
>
> if endian_flag EQ 'native' then begin
>   raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,
>   y_dim, z_dim], endian = 'native')
> endif else begin
>
>   if os_flag EQ 'little' then begin
>     raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,
>     y_dim, z_dim], endian = 'big')
>   endif else begin
>     raw_volume = read_binary(vol_name, DATA_TYPE = type, DATA_DIMS=[x_dim,
>     y_dim, z_dim], endian = 'little')
>   endelse
>
> endelse
>
>
> ; Put everything in a structure and return it to the main procedure
> ;analyze_vol = {VOL_INFO, $
>
> analyze_vol = {
>   vol_name: vol_name,  $
>   hdr_name: hdr_name,  $
>   ndims   : n,         $
>   x_dim   : x_dim,     $
>   y_dim   : y_dim,     $
>   z_dim   : z_dim,     $
>   spmtype : spm_type,  $
>   idl_type: type,      $
>   air_type: airtype,   $
>   x_vox   : x_vox,     $
>   y_vox   : y_vox,     $
>   z_vox   : z_vox,     $
>   m_sc    : m,         $
>   b_sc    : b,         $
>   sp_org  : origin,    $
>   data    : raw_volume $
> }
>

```

```

>
>
> return, analyze_vol
>
> END
>
>
> ;----- --
> ;+
> ; NAME:
> ;   ENDIAN
> ; PURPOSE:
> ;   Function indicating which endian the current machine uses.
> ; CATEGORY:
> ; CALLING SEQUENCE:
> ;   f = endian()
> ; INPUTS:
> ; KEYWORD PARAMETERS:
> ;   Keywords:
> ;   /LIST means list result to screen.
> ;   /TEXT means return /LIST text.
> ; OUTPUTS:
> ;   f = 0 if little, 1 if big.   out
> ; COMMON BLOCKS:
> ; NOTES:
> ;   Note: this is the order the bytes are for multibyte
> ;   numeric values. Use the IDL procedure BYTEORDER
> ;   to switch endian (use /LSWAP for 4 byte integers,
> ;   /L64SWAP for 8 byte integers).
> ; MODIFICATION HISTORY:
> ;   R. Sterner, 1999 Dec 13
> ;   R. Sterner, 2000 Apr 11 --- Added /TEXT
> ;
> ; Copyright (C) 1999, Johns Hopkins University/Applied Physics Laboratory
> ; This software may be used, copied, or redistributed as long as it is not
> ; sold and this copyright notice is reproduced on each copy made. This
> ; routine is provided as is without any express or implied warranties
> ; whatsoever. Other limitations apply as described in the file
> disclaimer.txt.
> ;
> ;----- --
> function endian, list=list, text=text, help=help
>
> if keyword_set(help) then begin
>   print, ' Function indicating which endian the current machine uses.'
>   print, ' f = endian()'
>   print, '  No args.'
>   print, '   f = 0 if little, 1 if big.   out'

```

```

> print,' Keywords:'
> print,' /LIST means list result to screen.'
> print,' /TEXT means return /LIST text.'
> print,' Note: this is the order the bytes are for multibyte'
> print,' numeric values. Use the IDL procedure BYTEORDER'
> print,' to switch endian (use /LSWAP for 4 byte integers,'
> print,' /L64SWAP for 8 byte integers).'
> return,"
> endif
>
> if fix([0B,1B],0) eq 1 then f=1 else f=0
> if (not keyword_set(list)) and (not keyword_set(text)) then return, f
>
> h = getenv('HOST')
> txt = h+' is '+(['little','big'])(f)+' endian'
> if keyword_set(list) then begin
>   print,' '
>   print,' '+txt+'.'
> endif
> if keyword_set(text) then f=txt
> return,f
>
> end
>
> "Michael A. Miller" <mmiller3@iupui.edu> wrote in message
> news:87zo0kstnh.fsf_-_@lumen.indyrad.iupui.edu...
>> Can anyone point me to IDL codes for reading and writing analyze
>> format images as defined be Mayo's Analyze and discussed at
>> http://www.dclunie.com/medical-image-faq/html/part5.html ?
>>
>> I've got some beginning of my own reader/writer. If there isn't
>> anything else out there, I'll clean it up and eventually make it
>> available.
>>
>> Mike
>>
>> P.S. Thanks for all the followups to my question about getting
>> access to command line arguments.
>>
>> --
>> Michael A. Miller mmiller3@iupui.edu
>> Imaging Sciences, Department of Radiology, IU School of Medicine
>
>

```
