NAME

TIFFOpen, TIFFFdOpen, TIFFClientOpen − open a TIFF file for reading or writing

SYNOPSIS

#include <tiffio.h>

TIFF* TIFFOpen(const char * *filename*, **const char *** *mode*)
TIFF* TIFFFdOpen(const int *fd*, **const char *** *filename*, **const char *** *mode*)

typedef tsize_t (*TIFFReadWriteProc)(thandle_t, tdata_t, tsize_t);
typedef toff_t (*TIFFSeekProc)(thandle_t, toff_t, int);
typedef int (*TIFFCloseProc)(thandle_t);
typedef toff_t (*TIFFSizeProc)(thandle_t);
typedef int (*TIFFMapFileProc)(thandle_t, tdata_t*, toff_t*);
typedef void (*TIFFUnmapFileProc)(thandle_t, tdata_t, toff_t);

TIFF* TIFFClientOpen(const char * *filename*, **const char *** *mode*, **thandle_t** *clientdata*, **TIFFReadWriteProc** *readproc*, **TIFFReadWriteProc** *writeproc*, **TIFFSeekProc** *seekproc*, **TIFFCloseProc** *closeproc*, **TIFFSizeProc** *sizeproc*, **TIFFMapFileProc** *mapproc*, **TIFFUnmapFileProc** *unmapproc*)

DESCRIPTION

TIFFOpen opens a TIFF file whose name is *filename* and returns a handle to be used in subsequent calls to routines in *libtiff*. If the open operation fails, then zero is returned. The *mode* parameter specifies if the file is to be opened for reading (‘‘r’’), writing (‘‘w’’), or appending (‘‘a’’) and, optionally, whether to override certain default aspects of library operation (see below). When a file is opened for appending, existing data will not be touched; instead new data will be written as additional subfiles. If an existing file is opened for writing, all previous data is overwritten.

If a file is opened for reading, the first TIFF directory in the file is automatically read (also see *TIFFSetDirectory*(3TIFF) for reading directories other than the first). If a file is opened for writing or appending, a default directory is automatically created for writing subsequent data. This directory has all the default values specified in TIFF Revision 6.0: *BitsPerSample*=1, *ThreshHolding*=bilevel art scan, *FillOrder*=1 (most significant bit of each data byte is filled first), *Orientation*=1 (the 0th row represents the visual top of the image, and the 0th column represents the visual left hand side), *SamplesPerPixel*=1, *RowsPerStrip*=infinity, *ResolutionUnit*=2 (inches), and *Compression*=1 (no compression). To alter these values, or to define values for additional fields, *TIFFSetField*(3TIFF) must be used.

TIFFFdOpen is like *TIFFOpen* except that it opens a TIFF file given an open file descriptor *fd*. The file’s name and mode must reflect that of the open descriptor. The object associated with the file descriptor **must support random access**.

TIFFClientOpen is like *TIFFOpen* except that the caller supplies a collection of functions that the library will use to do UNIX-like I/O operations. The *readproc* and *writeproc* are called to read and write data at the current file position. *seekproc* is called to change the current file position a la *lseek*(2). *closeproc* is invoked to release any resources associated with an open file. *sizeproc* is invoked to obtain the size in bytes of a file. *mapproc* and *unmapproc* are called to map and unmap a file’s contents in memory; c.f. *mmap*(2) and *munmap*(2). The *clientdata* parameter is an opaque ‘‘handle’’ passed to the client-specified routines passed as parameters to *TIFFClientOpen*.

OPTIONS

The open mode parameter can include the following flags in addition to the ‘‘r’’, ‘‘w’’, and ‘‘a’’ flags. Note however that option flags must follow the read-write-append specification.

l When creating a new file force information be written with Little-Endian byte order (but see below). By default the library will create new files using the native CPU byte order.

b When creating a new file force information be written with Big-Endian byte order (but see below). By default the library will create new files using the native CPU byte order.

L Force image data that is read or written to be treated with bits filled from Least Significant Bit (LSB) to Most Significant Bit (MSB). Note that this is the opposite to the way the library has worked from its inception.

B Force image data that is read or written to be treated with bits filled from Most Significant Bit (MSB) to Least Significant Bit (LSB); this is the default.

H Force image data that is read or written to be treated with bits filled in the same order as the native CPU.

M Enable the use of memory-mapped files for images opened read-only. If the underlying system does not support memory-mapped files or if the specific image being opened cannot be memory-mapped then the library will fallback to using the normal system interface for reading information. By default the library will attempt to use memory-mapped files.

m Disable the use of memory-mapped files.

C Enable the use of ''strip chopping'' when reading images that are comprised of a single strip or tile of uncompressed data. Strip chopping is a mechanism by which the library will automatically convert the single-strip image to multiple strips, each of which has about 8 Kilobytes of data. This facility can be useful in reducing the amount of memory used to read an image because the library normally reads each strip in its entirety. Strip chopping does however alter the apparent contents of the image because when an image is divided into multiple strips it looks as though the underlying file contains multiple separate strips. Finally, note that default handling of strip chopping is a compile-time configuration parameter. The default behaviour, for backwards compatibility, is to enable strip chopping.

c Disable the use of strip chopping when reading images.

h Read TIFF header only, do not load the first image directory. That could be useful in case of the broken first directory. We can open the file and proceed to the other directories.

4 ClassicTIFF for creating a file (default)

8 BigTIFF for creating a file.

D Enable use of deferred strip/tile offset/bytecount array loading. They will be loaded the first time they are accessed to. This loading will be done in its entirety unless the O flag is also specified.

O On-demand loading of values of the strip/tile offset/bytecount arrays, limited to the requested strip/tile, instead of whole array loading (implies D)

BYTE ORDER

The TIFF specification (**all versions**) states that compliant readers *must be capable of reading images written in either byte order*. Nonetheless some software that claims to support the reading of TIFF images is incapable of reading images in anything but the native CPU byte order on which the software was written. (Especially notorious are applications written to run on Intel-based machines.) By default the library will create new files with the native byte-order of the CPU on which the application is run. This ensures optimal performance and is portable to any application that conforms to the TIFF specification. To force the library to use a specific byte-order when creating a new file the ''b'' and ''l'' option flags may be included in the call to open a file; for example, ''wb'' or ''wl''.

RETURN VALUES

Upon successful completion *TIFFOpen*, *TIFFFdOpen*, and *TIFFClientOpen* return a TIFF pointer. Otherwise, NULL is returned.

DIAGNOSTICS

All error messages are directed to the *TIFFError*(3TIFF) routine. Likewise, warning messages are directed to the *TIFFWarning*(3TIFF) routine.

"%s": Bad mode. The specified *mode* parameter was not one of ''r'' (read), ''w'' (write), or ''a'' (append).

%s: Cannot open. *TIFFOpen*() was unable to open the specified filename for read/writing.

Cannot read TIFF header. An error occurred while attempting to read the header information.

Error writing TIFF header. An error occurred while writing the default header information for a new file.

Not a TIFF file, bad magic number %d (0x%x). The magic number in the header was not (hex) 0x4d4d or (hex) 0x4949.

Not a TIFF file, bad version number %d (0x%x). The version field in the header was not 42 (decimal).

Cannot append to file that has opposite byte ordering. A file with a byte ordering opposite to the native byte ordering of the current machine was opened for appending (“a”). This is a limitation of the library.

SEE ALSO

libtiff (3TIFF), *TIFFClose*(3TIFF)