PA-193: SECURE CODING PRINCIPLES AND PRACTICES

TIFF IMAGE PARSER

Ashwin Arvind Yakkundi, Rajesh Kumar, Ananya Chatterjee

Faculty of Informatics

Masaryk University

1. Tagged Image File Format, abbreviated TIFF or TIF, is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry,[1] and photographers. The TIFF format is widely supported by image-manipulation applications, by publishing and page layout applications, and by scanning, faxing, word processing, optical character recognition and other applications.

2. TIFF is a flexible, adaptable file format for handling images and data within a single file, by including the header tags (size, definition, image-data arrangement, applied image compression) defining the image's geometry. A TIFF file, for example, can be a container holding JPEG (lossy) and PackBits (lossless) compressed images. A TIFF file also can include a vector-based clipping path (outlines, croppings, image frames). The ability to store image data in a lossless format makes a TIFF file a useful image archive, because, unlike standard JPEG files, a TIFF file using lossless compression (or none) may be edited and re-saved without losing image quality. This is not the case when using the TIFF as a container holding compressed JPEG. Other TIFF options are layers and pages.

3. TIFF is an image file format. In this document, a file is defined to be a sequence of 8-bit bytes, where the bytes are numbered from 0 to N. The largest possible TIFF file is 2**32 bytes in length. A TIFF file begins with an 8-byte image file header that points to an image file directory (IFD). An image file directory contains information about the image, as well as pointers to the actual image data.

TIFF Structure

4. **Bytes 0-1:** A TIFF file begins with an 8-byte image file header, containing the following information:

The byte order used within the file. Legal values are:

"II" (4949.H)

"MM" (4D4D.H)

5. In the "II" format, byte order is always from the least significant byte to the most significant byte, for both 16-bit and 32-bit integers This is called little-endian byte order.

6. In the "MM" format, byte order is always from most significant to least significant, for both 16-bit and 32-bit integers. This is called big-endian byte order.

7. **Bytes 2-3.** An arbitrary but carefully chosen number (42) that further identifies the file as a TIFF file. The byte order depends on the value of Bytes 0-1.

8. **Bytes 4-7.** The offset (in bytes) of the first IFD. The directory may be at any location in the file after the header but must begin on a word boundary. In particular, an Image File Directory may follow the image data it describes. The term byte offset is always used in this document to refer to a location with respect to the beginning of the TIFF file. The first byte of the file has an offset of 0.

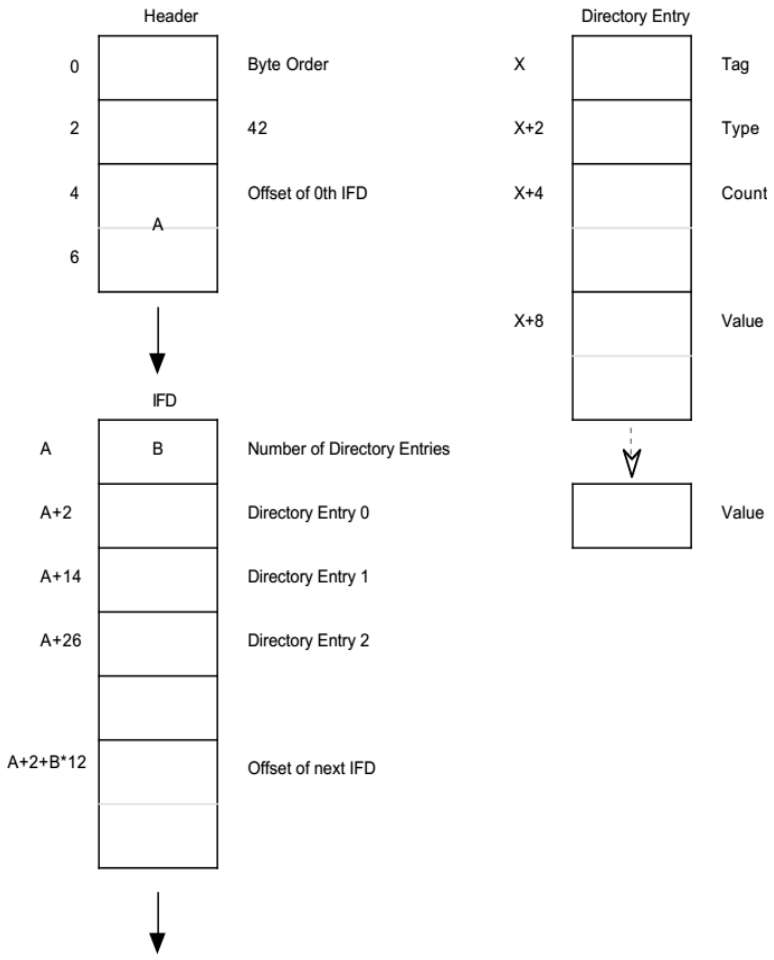

Figure 1

9. **IFD Entry.** An Image File Directory (IFD) consists of a 2-byte count of the number of directory entries (i.e., the number of fields), followed by a sequence of 12-byte field entries, followed by a 4-byte offset of the next IFD (or 0 if none). (Do not forget to write the 4 bytes of 0 after the last IFD.) There must be at least 1 IFD in a TIFF file and each IFD must have at least one entry.

Each 12-byte IFD entry has the following format:

Bytes 0-1: The Tag that identifies the field.

Bytes 2-3: The field Type.

Bytes 4-7: The number of values, Count of the indicated Type

Bytes 8-11: The Value Offset, the file offset (in bytes) of the Value for the field.

10. The Value is expected to begin on a word boundary; the corresponding Value Offset will thus be an even number. This file offset may point anywhere in the file, even after the image data.

Example Image: Bilevel Image.

11. A bilevel image contains two colors—black and white. TIFF allows an application to write out bilevel data in either a white-is-zero or black-is-zero format. The field that records this information is called PhotometricInterpretation.

PhotometricInterpretation

Tag = 262 (106.H)

Type = SHORT

Values:

0 = WhiteIsZero. For bilevel and grayscale images: 0 is imaged as white. The maximum value is imaged as black. This is the normal value for Compression=2.

1 = BlackIsZero. For bilevel and grayscale images: 0 is imaged as black. The maximum value is imaged as white. If this value is specified for Compression=2, the image should display and print reversed.

12. Data can be stored either compressed or uncompressed.

Compression

Tag = 259 (103.H)

Type = SHORT

Values:

1 = No compression, but pack data into bytes as tightly as possible, leaving no unused bits (except at the end of a row). The component values are stored as an array of type BYTE. Each scan line (row) is padded to the next BYTE boundary.

2 = CCITT Group 3 1-Dimensional Modified Huffman run length encoding.

32773 = PackBits compression, a simple byte-oriented run length scheme. See the PackBits section for details. Data compression applies only to raster image data. All other TIFF fields are unaffected.

13. An image is organized as a rectangular array of pixels. The dimensions of this array are stored in the following fields:

ImageLength

Tag = 257 (101.H)

Type = SHORT or LONG

14. The number of rows (sometimes described as scanlines) in the image.

ImageWidth

Tag = 256 (100.H)

Type = SHORT or LONG

15. The number of columns in the image, i.e., the number of pixels per scanline. Applications often want to know the size of the picture represented by an image. This information can be calculated from ImageWidth and ImageLength given the following resolution data:

ResolutionUnit

Tag = 296 (128.H)

Type = SHORT

Values:

1 = No absolute unit of measurement. Used for images that may have a non-square aspect ratio but no meaningful absolute dimensions.

2 = Inch.

3 = Centimeter.

Default = 2 (inch)

XResolution

Tag = 282 (11A.H)

Type = RATIONAL

16. The number of pixels per ResolutionUnit in the ImageWidth (typically, horizontal - see Orientation) direction.

YResolution

Tag = 283 (11B.H)

Type = RATIONAL

17. The number of pixels per ResolutionUnit in the ImageLength (typically, vertical) direction.

18. Compressed or uncompressed image data can be stored almost anywhere in a TIFF file. TIFF also supports breaking an image into separate strips for increased editing flexibility and efficient I/O buffering. The location and size of each strip is given by the following fields:

RowsPerStrip

Tag = 278 (116.H)

Type = SHORT or LONG

19. The number of rows in each strip (except possibly the last strip.) For example, if ImageLength is 24, and RowsPerStrip is 10, then there are 3 strips, with 10 rows in the first strip, 10 rows in the second strip, and 4 rows in the third strip. (The data in the last strip is not padded with 6 extra rows of dummy data.)

StripOffsets

Tag = 273 (111.H)

Type = SHORT or LONG

20. For each strip, the byte offset of that strip.

StripByteCounts

Tag = 279 (117.H)

Type = SHORT or LONG

21. For each strip, the number of bytes in that strip after any compression.

Offset (hex)	Description	Value (numeric values are expressed in hexadecimal notation)			
Header:					
0000	Byte Order	4D4D			
0002	42	002A			
0004	1st IFD offset	00000014			
IFD:					
0014	Number of Directory Entries	000C			
0016	NewSubfileType	00FE	0004	00000001	00000000
0022	ImageWidth	0100	0004	00000001	000007D0
002E	ImageLength	0101	0004	00000001	00000BB8
003A	Compression	0103	0003	00000001	8005 0000
0046	PhotometricInterpretation	0106	0003	00000001	0001 0000
0052	StripOffsets	0111	0004	000000BC	000000B6
005E	RowsPerStrip	0116	0004	00000001	00000010
006A	StripByteCounts	0117	0003	000000BC	000003A6
0076	XResolution	011A	0005	00000001	00000696
0082	YResolution	011B	0005	00000001	0000069E
008E	Software	0131	0002	0000000E	000006A6
009A	DateTime	0132	0002	00000014	000006B6
00A6	Next IFD offset	00000000			
Values longer than 4 bytes:					
00B6	StripOffsets	Offset0, Offset1, ... Offset187			
03A6	StripByteCounts	Count0, Count1, ... Count187			
0696	XResolution	0000012C 00000001			
069E	YResolution	0000012C 00000001			
06A6	Software	"PageMaker 4.0"			
06B6	DateTime	"1988:02:18 13:59:59"			

Figure 2

Implementation of the Code.

22. First, extensive study of the TIFF format was done by going through the TIFF 6.0 documentation on the adobe website.

23. The basic structure for parsing the values of the TIFF header was identified and the same has been given at figure 3 below.

```
struct TiffFrame {
        uint32_t width;
        uint32_t height;
        uint16_t compression;
        uint32_t rowsperstrip;
        uint32_t* stripoffsets;
        uint32_t* stripbytecounts;
        uint32_t stripcount;
        uint16_t samplesperpixel;
        uint16_t* bitspersample;
        uint16_t planarconfiguration;
        uint16_t sampleformat;
        uint32_t imagelength;
};
```

```
struct TiffFile {
        FILE* file;
        uint8_t systembyteorder;
        uint8_t filebyteorder;
        uint32_t firstrecord_offset;
        uint32_t nextifd_offset;
        uint64_t filesize;
        TiffFrame currentFrame;
};
```

Figure 3

24. Next, the basic structure for the code was identified. The layout of the code follows the TIFF 6.0 documentation. The overview of the structure of the code is as given in figure 5 below.

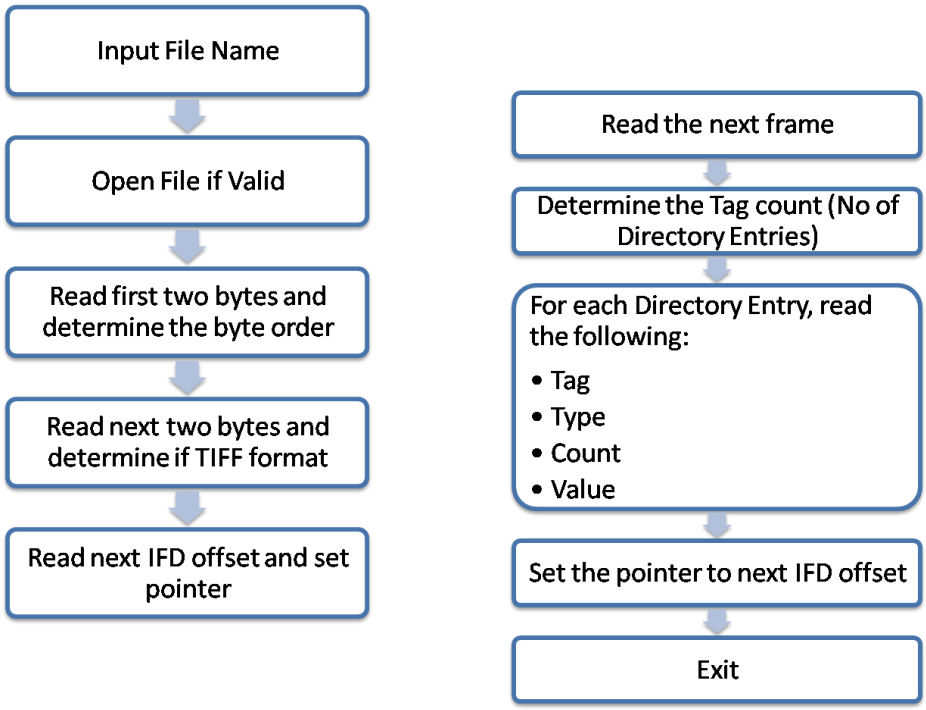

Figure 4

25. The overview of the structure of the code is given below:-

(a) Getting the Input file name from the user and reading the input with input validation.

(b) Check if the input file is valid, has valid length and only white listing the input filename, it is accepted. The file is opened if found to be valid.

(c) Read the first two bytes which gives the byte order such as the LITTLE ENDIAN or BIG ENDIAN. Depending on the byte order the bytes are read from most significant or least significant bit.

(d) Read the next two bytes to determine the TIFF format bits. Hexadecimal No 42h is used to identify the file as a TIFF image file.

(e) Read the next IFD offset and set the pointer to point to the first byte of directory entry.

(f) Determine the Tag Count which determine the number of directory entries. Each directory has information on specific property of the image.

(g) Set the pointer to the first directory entry and read the tag, the type, count and the value associated.

(h) Repeat the above mentioned step for the number of directories and display all the values associated with the property.

(j) Finally, when all the directory entries are read, the pointer is set to the next IFD offset and exits.

Further Scope.

26. The code can be taken further in the following ways:

(a) Improve the code structure by making it more modular.

(b) Reading of the more complicated tags.

(c) parsing of the data values of the image depending on the type of color. Eg Bilevel, RGB, Palette Colour etc.

(d) Further the code for reading TIFF format containing multiple IFD entries.

References

1. TIFF Revision 6.0 Final — June 3, 1992
https://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

2. https://en.wikipedia.org/wiki/TIFF