FreeXL

1.0.6

Generated by Doxygen 1.9.1

Chapter 1

Main Page

1.1 Introduction

FreeXL is an open source library to extract valid data from within an Excel (.xls) spreadsheet.

The FreeXL design goals are:

- to be simple and lightweight
- to be stable, robust and efficient
- to be easily and universally portable.
- completely ignore any GUI-related oddity

Note that the final goal means that FreeXL ignores at all fonts, sizes and alignments, and most formats. It ignores Pivot Table, Charts, Formulas, Visual Basic macros and so on.

FreeXL is structurally simple and quite light-weight (typically 40-80K of object code, stripped). FreeXL has one key dependency - GNU libiconv, which is used for character set conversions. This is often provided as part of the C library on Linux systems, and is widely available.

Building and installing FreeXL is straightforward:

```
./configure
make
make install
```

Linking FreeXL to your own code is usually simple:

```
gcc my_program.c -o my_program -lfreexl
```

On some systems you may have to provide a slightly more complex arrangement:

```
gcc -I/usr/local/include my_program.c -o my_program \
  -L/usr/local/lib -lfreexl -liconv -lm
```

FreeXL also provides pkg-config support, so you can also do:

```
gcc -I/usr/local/include my_program.c -o my_program `pkg-config --libs freexl`
```

I sincerely hope FreeXL could be useful to many of you. Excel ∗.xls spreadsheets are widespread, and although Microsoft itself is strongly pushing the new XML based formats, there is still a lot of legacy data stored in the older binary formats.

So in an era of open data, a simple and easy way to extract data from .xls is surely useful. The original use of FreeXL was to support the SQLite / SpatiaLite VirtualXL driver (implementing direct access to .xls files via SQL). However there are many other possibilities, including use with shell scripts and simple wrappers for Python, Perl and other very high level languages.

FreeXL is licensed under the MPL tri-license terms: you are free to choose the best-fit license between:

- the MPL 1.1
- the GPL v2.0 or any subsequent version
- the LGPL v2.1 or any subsequent version

Enjoy, and happy coding

Chapter 2

The background story for FreeXL

Where, when and why a new free software library was born...

At the end of April 2011 Markus Neteler [http://en.wikipedia.org/wiki/Markus_Neteler] and I (Sandro) were in Udine in Northern Italy, attending the annual Italian gvSIG Users conference. So, on a lovely hot and sunny spring evening, accompanied by a picturesque sunset, we were sitting in the town centre in the pleasant Piazza Matteotti aka Piazza San Giacomo, peacefully drinking some spritz while eating chips and peanuts.

Figure 2.1 Piazza

You'll have to admit, it was a really dangerous situation: as a general safety rule, never let two developers sit idle for too long. Some odd and crazy idea will inevitably occur to them (of course, drinking too much spritz can contribute as well).

Markus was desperately attempting to convince me that implementing a VirtualTable driver for SpatiaLite supporting direct SQL reading ofExcel .xls files was a good initiative. Surely it would be useful for many users. I strongly resisted, fiercely fighting and rejecting such idea, on the basis that attempting to read proprietary closed formats such as Excel was a complete nonsense, and probably a very difficult if not impossible task.

At the end of this very animated discussion, Markus pronounced the magic spell that suddenly convinced me about the absolute validity of his suggestions: *I can raise some funding for this project* After hearing such wise and significant words from Markus I immediately realized that developing a new driver for accessing Excel .xls documents surely was an exciting and useful task after all.

So FreeXL / VirtualXL was conceived at that exact moment, and slowly began its development cycle.

Chapter 3

File format specifications and source information

The .xls binary file format is extensively documented and is publicly available.

The most authoritative source is made available by Microsoft at http://msdn.microsoft.com/en-us/library/cc313154%28v=office.12%29.aspx

A simpler option is made available by Open Office: http://sc.openoffice.org/excelfileformat.pdf

Searching the web you'll easily find several other valuable information sources.

Chapter 4

About the .xls binary format

What a .xls binary file really is

(Prepare yourself to be continuously surprised by many unexpected revelations ...)

You may already know that there are many different versions of .xls files. Different versions have different capabilities. So we'll start by reviewing the Excel evolutionary history and and we'll introduce some Microsoft jargon because it's central to understanding the underlying operations.

4.1 CFBF

Unexpected Revelation #1: *There is no .xls file format. Its really a common file suffix applied to many different things.*

Recent Microsoft Office document files are based on a common container layout named CFBF (Compound File Binary Format). This container format is the same for Excel (.xls), Word (.doc_ and PowerPoint (.ppt) amongst other applications. More information:

- http://en.wikipedia.org/wiki/Compound_File_Binary_Format
- http://msdn.microsoft.com/en-us/library/dd942138%28v=prot.13%29.aspx

Unexpected Revelation #2: *CFBF is more of a file system than a file format*

A CFBF file is divided into many equal-sized blocks named sectors. Such sectors cannot be directly accessed. In order to retrieve sectors in the expected logical order a FAT (File Allocation Table) is allocated within the CFBF file. A CFBF file is internally organized as if it was a raw physical disk. The design is based on Microsoft own FAT file-system as used by MS-DOS and early versions of Windows. The first sector of the CFBF file acts as if it was a kind-of MBR (Master Boot Record) - this first sector provides information about the layout and type of the CFBF file, such as block/sector size and version. A FAT chain allows a reader to re-assemble the sectors in the required logical order. There is a list of free block, and very large files may use a double indirection (DIFAT - Double Indirection FAT). A CFBF file always has at least a root directory: but a complete directory tree can be provided.

A CFBF file can contain many and many distinct independent files. Just to make things a little clearer, Microsoft calls such pseudo-files (I mean: the many fake ones contained within the real CFBF file) *streams*

The practical consequence is that any software tool attempting to access an Excel binary document must first be able to correctly access this CFBF container format.

4.2 BIFF

Unexpected Revelation #3: *An Excel document will contain a stream (pseudo-file) named Workbook in the root directory of the CFBF file (filesystem).*

The Workbook stream is internally structured accordingly to the BIFF (Binary Interchange File Format) specifications. You can think of the BIFF as the real Excel binary format (following more conventional naming rules). Several BIFF versions were introduced during the years: and there are significant differences between them.

An useful correspondence table relating corresponding Excel and BIFF versions:

Excel Version	Commercial Name	BIFF Version	Release Year	Notes
2.x	Excel 2.0	BIFF2	1987	Before CFBF. File is the BIFF stream, containing a single worksheet.
3.0	Excel 3.0	BIFF3	1990	Before CFBF. File is the BIFF stream, containing a single worksheet.
4.0	Excel 4.0	BIFF4	1992	Before CFBF. File is the BIFF stream, containing a single worksheet.
5.0	Excel 5.0	BIFF5	1993	Starting with BIFF5, a single Workbook can internally store many individual Worksheets. The BIFF stream is stored in the CFBF file container.
7.0	Excel 95	BIFF5	1995	
8.0	Excel 98	BIFF8	1998	
9.0	Excel 2000	BIFF8	1999	
10.0	Excel XP	BIFF8	2001	
11.0	Excel 2003	BIFF8	2003	
12.0	Excel 2007	BIFF8	2007	Introduced alternate XML format, which is usually the default for new files.
14.0	Excel 2010	BIFF8	2010	XML format is usually the default for new files.

Note that FreeXL does not support the new XML format which is a completely different and unrelated format.

Perhaps you are now expecting that BIFF will simply and directly encode your spreadsheet data. Unfortunately, you should have know better given the steps we took to get here...

Unexpected Revelation #4: *Any BIFF stream (pseudo-file stored within a CFBF container file) is internally organized as a collection of variable-length records..*

Each record starts with

- a 16 bit unsigned integer specifying the record type
- another 16 bit unsigned integer specifies the record data length (in bytes) excluding the standard type-size prefix.

Note that there are many different record types, and the record size / layout may differ for different BIFF versions.

Three record types have an absolutely special meaning:

- a BOF [Beginning Of File] record marks starting of a different sub-stream.
- an EOF [End Of File] record marks ending of current sub-stream.
- a CONTINUE record means that the previous record exceeded the maximum size for a record, and the previous record data payload will be spanned on following CONTINUE records for as many CONTINUE records as are required to store the full data size.

Unexpected Revelation #5: *So a BIFF stream (pseudo-file) isn't really a file - it's more like a collection of individual sub-streams, each one of which is enclosed between BOF / EOF markers.*

The most recent BIFF8 requires that at least the following internal sub-streams are be defined:

- the first sub-stream contains workbook level global data and metadata, such as author, password protection, styles, formats, window settings and so on
- list of individual worksheets included into the Workbook, where each worksheet is identified by a name and by a type (data Worksheet, Chart, Visual Basic module ... visible, hidden ...), and relative offset position of the corresponding BOF record allows for fast positioning.
- any text string is stored here into the SST [Shared String Table], so individual text cells simply refer the corresponding SST entry by index (in all previous BIFF version text strings are directly stored into the appropriate cell).
- any subsequent sub-stream represents a single Worksheet, and the most relevant data stored at Worksheet level are dimension (number of valid rows and columns) and any cell value data.

We will now see how BIFF encodes individual data types with several further amazing surprises are still to come. Be prepared!

4.3 BIFF

Leaving aside special values such as images, OLE, COM, Visual Basic related items and so on, the basic data types are supported in BIFF:

- text strings
- numbers (both integers and decimals)
- dates, date-times and times
- NULL (empty cell)

Note that any multi-byte value is stored in BIFF accordingly using Little Endian byte ordering (i.e. least significant byte comes first, most significant byte comes last).

BIFF Record Type	BIFF Version	Content Type
INTEGER	BIFF2	16 bit unsigned integer
NUMBER	BIFF2 BIFF3 BIFF4 BIFF5 BIFF8	64 bit floating point (double precision)
RK	BIFF3 BIFF4 BIFF5 BIFF8	number, variant-type: INTEGER FLOAT DATE DATETIME TIME (please see the corresponding detailed description)
MULRK	BIFF5 BIFF8	a variable-sized array of elementary RK values. associated to a range of consecutive cells on the same row
LABEL	BIFF2 BIFF3 BIFF4 BIFF5 BIFF8	text string, variable-length. (please see the corresponding detailed description)
LABELSST	BIFF8	text string, variable-length. based on the global SST [Shared String Table] stored at the workbook level, as a LABELSST simply requires providing the corre-

So the BIFF record type that is easy to handle is NUMBER, which is essentially a C-style *double*. Other record types require additional handling.

4.3.1 RK values

An RK value is a 32 bit value.

The least significant two bits are a bit-mask (in little endian order, so the least two significant bits in the first byte that is read):

- if 0x02 is set the RK value represents a 30 bit signed integer, otherwise it represents a 64 bit floating point double precision number requiring special reconstruction.
- if 0x01 is set the corresponding value needs to be divided by 100, so even an integer actually becomes a floating point double precision.

When interpreting RK values as a signed integer, right shifting two bits is required:
```
int value = rk_value » 2;
```

When interpreting RK values as a 64 bit floating point, two steps are required:

- the RK value requires appropriate masking:
  ```
  int value = rk_value & 0xfffffffc;
  ```
- then the 32 bit value will be copied into a 64 bit buffer, and the least significant four bytes need to be initialized as zeroes: 0x00000000.

As a final step, if 0x01 was set into the bit-mask, now we have to divide by 100 before returning the effective cell value. So for 32 bit integers:
```
double final_value = (double)value / 100.0;
```

and for 64 bit floats:
```
double final_value = value / 100.0;
```

4.3.2 Text values

Any BIFF version from BIFF2 to BIFF5 simply supports CodePage based character encoding, i.e. each character simply requires 8 bits to be represented (single byte). Correct representation of characters requires knowing which one CodePage table has to be applied. This can be determined from the workbook or worksheet metadata (it is the CODEPAGE record).

BIFF8 is much more sophisticated, since any text string is usually encoded as Unicode in UTF-16 Little Endian [UTF-16LE] format. This encoding is a multi-byte encoding (two bytes are required to represent a single character), but being universal no character table is required.

BIFF text strings are never null-terminated. The actual length is always explicitly stated, as an 8 bit unsigned int or as a 16 bit unsigned int (depending on BIFF versions).

FreeXL is intended to be strictly interoperable with SQLite and SpatiaLite, so any text string has to be converted to UTF-8 encoding. GNU libiconv can easily handle any required charset conversion. So we can simply fetch the appropriate bytes, then call iconv() as appropriate, and we'll immediately get back the corresponding UTF-8 encoded text string.

Converting Unicode based text strings is a little more complex, because each Unicode string is prefixed by a mask byte, specifying how the string is encoded:

- if 0x01 is set, then the string really is 16 bit per character Unicode, otherwise a stripped notation is used instead. Stripped notation means that the characters are actually represented as single bytes, so already have the UTF8 equivalent.
- if 0x04 and/or 0x08 are set, than some further variable-length data (providing information on text decoration such as italics, bold, underline) is inserted immediately before and after the text string itself, so we must carefully skip over this extra data so to maintain the right byte alignment.

Note that the string length is expressed in characters, not in bytes, so the actual length in bytes is twice the indicated length.

4.3.3 Retrieving Date, DateTime and Time values.

Dates, DateTimes and Time values are also a little complicated. Any Date is expressed as an Integer (number of days since the conventional reference day):

- for Windows Excel the reference day (day 0) is 1900, January 1
- for Mac Excel the reference day (day 0) is 1904, January 2

There is no possible ambiguity, because the DATEMODE metadata record specifies tells which reference day is to be used.

An odd bug affects Excel, which (incorrectly) treats 1900 as a leap year. Therefore, the non-existent 29 February 1900 has to be included in the days calculation so to get the expected Date value.

Any Time is expressed as a Fraction (percent of seconds since midnight). 0.5 corresponds to 12:00:00 (midday), 0.25 corresponds to 06:00:00, 0.75 corresponds to 18:00:00 and so on.

So a DateTime is simply the sum of a Date value and of a Time value. Dates can be represented by Integers: but Times and DateTimes require a floating point number.

The complication with Dates, DateTimes and Time values is that the data-type does not specify when a cell values has to be interpreted as a Date or Time - it is simply an Integer or Float numbers like any other. A further indirection has to applied so to correctly recognize Dates, DateTimes and Times:

- each NUMBER, RK or MULRK value exposes an index referencing the XF (Extended Format) entry associated with the corresponding cell.
- each XF record specifies an unique combination of font, alignment, color and so on, however a further indirection specifies the corresponding FORMAT entry
- each FORMAT record specifies an output format, such as M/D/YY, h:mm:ss AM/PM or M/D/YY h:mm: and this finally gives us a good chance to guess which cell values are intended to represent Date/Time values.

Both XF and FORMAT records are globally stored at the Workbook level, and represent ordered arrays.

If you haven't yet given up, if you aren't yet become totally mind-boggled, and if you are still awake and conscious, then you now know how .xls files are internally organized and structured.

Be happy and feel proud of yourself.

Chapter 5

Other tools and libraries

There an impressively wide choice of Free and open source libraries and tools supporting the .xls format.

A sample:

- Gnumeric [http://projects.gnome.org/gnumeric/] seems to be the pioneer of them all, and probably was the first FLOSS tool able to read and write .xls (circa 2001). Part of the Gnome Office project.
- KSpread / Caligra Tables [http://www.calligra-suite.org/tables/] / KCells [http://www.koffice.org/kcells/] are similar (but now distinct) programs from the KOffice and Calligra Office projects.
- Open Office Calc [http://www.openoffice.org/product/calc.html] and LibreOffice Calc [http://www.libreoffice.org/features/calc/] are similar (but now distinct) spreadsheet applications originally from the StartOffice code base. Probably the most comprehensive support in FLOSS.
- Apache POI-HSSF [http://poi.apache.org/spreadsheet/index.html] is a sophisticated Java library fully supporting .xls files.
- JExcelAPI [http://jexcelapi.sourceforge.net/] is another Java library (much simpler and lighter than POI-HSSF) supporting .xls files.
- several C or C++ libraries exist as well: quite curiously one is named libxls and another xlslib, but they are two absolutely distinct and unrelated packages
- There are other implementations are available based on .NET or PHP.

A quick critical review:

- GUI tools implementations are difficult to re-use. They focus on import of all formulas, GUI presentation and so on, which is really a different use.
- Java libraries seem to be really interesting, but Java is difficult to call from a C or C++ program.
- Several C/C++ libraries exist, but none of them seems to be sufficient and stable as required. Some are still marked to be “beta-stage” despite being released some four or five years ago - project activity seems to be very low, and download statistics are discouraging.

Conclusion: a suitable C/C++ library supporting data extraction from .xls files doesn't seem to exists: or at least, there is no obvious reference choice.

So we'll go on the hardest way, we'll develop yet another .xls reading library: FreeXL.

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 8

Data Structure Documentation

8.1 FreeXL_CellValue_str Struct Reference

Container for a cell value.

`#include <freexl.h>`

Data Fields

- unsigned char type

  The type of data stored in this cell.
- union {
  int int_value
  if type is FREEXL_CELL_INT, then the corresponding value will be returned as int_value
  double double_value
  if type is FREEXL_CELL_DOUBLE, then the corresponding value will be returned as double_value
  const char * text_value
  if type is FREEXL_CELL_TEXT, FREEXL_CELL_SST_TEXT, FREEXL_CELL_DATE, FREEXL_CELL_DATETIME or FRE
  } value

  The value of the data stored in the cell.

8.1.1 Detailed Description

Container for a cell value.

freexl_get_cell_value() takes a pointer to this structure, and fills in the appropriate values.

```
FreeXL_CellValue val;
freexl_get_cell_value(..., &val);
switch (val.type)
{
    case FREEXL_CELL_INT:
     printf("Int=%d\n", val.value.int_value;
     break;
    case FREEXL_CELL_DOUBLE:
     printf("Double=%1.2f\n", val.value.double_value;
     break;
    case FREEXL_CELL_TEXT:
    case FREEXL_CELL_SST_TEXT:
     printf("Text='%s'\n", val.value.text_value;
     break;
    case FREEXL_CELL_DATE:
```

```
    case FREEXL_CELL_DATETIME:
    case FREEXL_CELL_TIME:
     printf("DateOrTime='%s'\n", val.value.text_value;
     break;
    case FREEXL_CELL_NULL:
     printf("NULL\n");
     break;
    default:
     printf("Invalid data-type\n");
     break;
}
```

Examples

xl2sql.c.

8.1.2 Field Documentation

8.1.2.1 type

`unsigned char FreeXL_CellValue_str::type`

The type of data stored in this cell.

Can be one of the following:

- FREEXL_CELL_NULL the cell contains a NULL value.
- FREEXL_CELL_INT the cell contains an INTEGER value.
- FREEXL_CELL_DOUBLE the cell contains a DOUBLE value.
- FREEXL_CELL_TEXT or FREEXL_CELL_SST_TEXT the cell contains a text string (always UTF-8 encoded)
- FREEXL_CELL_DATE the cell contains a date, encoded as a 'YYYY-MM-DD' string value
- FREEXL_CELL_DATETIME the cell contains a date and time, encoded as a 'YYYY-MM-DD HH:MM:SS' string value
- FREEXL_CELL_TIME the cell contains a time, encoded as a 'HH:MM:SS' string value

Examples

xl2sql.c.

8.1.2.2

`union { ... } FreeXL_CellValue_str::value`

The value of the data stored in the cell.

Which part of the union is valid is determined by the type value.

Examples

xl2sql.c.

The documentation for this struct was generated from the following file:

- headers/freexl.h

Chapter 9

File Documentation

9.1 headers/freexl.h File Reference

Function declarations and constants for FreeXL library.

Data Structures

- struct FreeXL_CellValue_str

  Container for a cell value.

Macros

- #define FREEXL_UNKNOWN 0

  query is not applicable, or information is not available
- #define FREEXL_CFBF_VER_3 3

  CFBF file is version 3.
- #define FREEXL_CFBF_VER_4 4

  CFBF file is version 4.
- #define FREEXL_CFBF_SECTOR_512 512

  CFBF file uses 512 byte sectors.
- #define FREEXL_CFBF_SECTOR_4096 4096

  CFBF file uses 4096 (4K) sectors.
- #define FREEXL_BIFF_VER_2 2

  BIFF file is version 2.
- #define FREEXL_BIFF_VER_3 3

  BIFF file is version 3.
- #define FREEXL_BIFF_VER_4 4

  BIFF file is version 4.
- #define FREEXL_BIFF_VER_5 5

  BIFF file is version 5.
- #define FREEXL_BIFF_VER_8 8

  BIFF file is version 9.
- #define FREEXL_BIFF_MAX_RECSZ_2080 2080

  Maximum BIFF record size is 2080 bytes.

- #define FREEXL_BIFF_MAX_RECSZ_8224 8224

  Maximum BIFF record size is 8224 bytes.
- #define FREEXL_BIFF_DATEMODE_1900 1900

  BIFF date mode starts at 1 Jan 1900.
- #define FREEXL_BIFF_DATEMODE_1904 1904

  BIFF date mode starts at 2 Jan 1904.
- #define FREEXL_BIFF_OBFUSCATED 3003

  BIFF file is password protected.
- #define FREEXL_BIFF_PLAIN 3004

  BIFF file is not password protected.
- #define FREEXL_BIFF_ASCII 0x016F

  BIFF file uses plain ASCII encoding.
- #define FREEXL_BIFF_CP437 0x01B5

  BIFF file uses CP437 (OEM US format) encoding.
- #define FREEXL_BIFF_CP720 0x02D0

  BIFF file uses CP720 (Arabic DOS format) encoding.
- #define FREEXL_BIFF_CP737 0x02E1

  BIFF file uses CP737 (Greek DOS format) encoding.
- #define FREEXL_BIFF_CP775 0x0307

  BIFF file uses CP775 (Baltic DOS format) encoding.
- #define FREEXL_BIFF_CP850 0x0352

  BIFF file uses CP850 (Western Europe DOS format) encoding.
- #define FREEXL_BIFF_CP852 0x0354

  BIFF file uses CP852 (Central Europe DOS format) encoding.
- #define FREEXL_BIFF_CP855 0x0357

  BIFF file uses CP855 (OEM Cyrillic format) encoding.
- #define FREEXL_BIFF_CP857 0x0359

  BIFF file uses CP857 (Turkish DOS format) encoding.
- #define FREEXL_BIFF_CP858 0x035A

  BIFF file uses CP858 (OEM Multilingual Latin 1 format) encoding.
- #define FREEXL_BIFF_CP860 0x035C

  BIFF file uses CP860 (Portuguese DOS format) encoding.
- #define FREEXL_BIFF_CP861 0x035D

  BIFF file uses CP861 (Icelandic DOS format) encoding.
- #define FREEXL_BIFF_CP862 0x035E

  BIFF file uses CP862 (Hebrew DOS format) encoding.
- #define FREEXL_BIFF_CP863 0x035F

  BIFF file uses CP863 (French Canadian DOS format) encoding.
- #define FREEXL_BIFF_CP864 0x0360

  BIFF file uses CP864 (Arabic DOS format) encoding.
- #define FREEXL_BIFF_CP865 0x0361

  BIFF file uses CP865 (Nordic DOS format) encoding.
- #define FREEXL_BIFF_CP866 0x0362

  BIFF file uses CP866 (Cyrillic DOS format) encoding.
- #define FREEXL_BIFF_CP869 0x0365

  BIFF file uses CP869 (Modern Greek DOS format) encoding.
- #define FREEXL_BIFF_CP874 0x036A

  BIFF file uses CP874 (Thai Windows format) encoding.
- #define FREEXL_BIFF_CP932 0x03A4

  BIFF file uses CP932 (Shift JIS format) encoding.
- #define FREEXL_BIFF_CP936 0x03A8

BIFF file uses CP936 (Simplified Chinese GB2312 format) encoding.

- #define FREEXL_BIFF_CP949 0x03B5

  BIFF file uses CP949 (Korean) encoding.
- #define FREEXL_BIFF_CP950 0x03B6

  BIFF file uses CP950 (Traditional Chinese Big5 format) encoding.
- #define FREEXL_BIFF_UTF16LE 0x04B0

  BIFF file uses Unicode (UTF-16LE format) encoding.
- #define FREEXL_BIFF_CP1250 0x04E2

  BIFF file uses CP1250 (Central Europe Windows) encoding.
- #define FREEXL_BIFF_CP1251 0x04E3

  BIFF file uses CP1251 (Cyrillic Windows) encoding.
- #define FREEXL_BIFF_CP1252 0x04E4

  BIFF file uses CP1252 (Windows Latin 1) encoding.
- #define FREEXL_BIFF_CP1253 0x04E5

  BIFF file uses CP1252 (Windows Greek) encoding.
- #define FREEXL_BIFF_CP1254 0x04E6

  BIFF file uses CP1254 (Windows Turkish) encoding.
- #define FREEXL_BIFF_CP1255 0x04E7

  BIFF file uses CP1255 (Windows Hebrew) encoding.
- #define FREEXL_BIFF_CP1256 0x04E8

  BIFF file uses CP1256 (Windows Arabic) encoding.
- #define FREEXL_BIFF_CP1257 0x04E9

  BIFF file uses CP1257 (Windows Baltic) encoding.
- #define FREEXL_BIFF_CP1258 0x04EA

  BIFF file uses CP1258 (Windows Vietnamese) encoding.
- #define FREEXL_BIFF_CP1361 0x0551

  BIFF file uses CP1361 (Korean Johab) encoding.
- #define FREEXL_BIFF_MACROMAN 0x2710

  BIFF file uses Mac Roman encoding.
- #define FREEXL_CELL_NULL 101

  Cell has no value (empty cell)
- #define FREEXL_CELL_INT 102

  Cell contains an integer value.
- #define FREEXL_CELL_DOUBLE 103

  Cell contains a floating point number.
- #define FREEXL_CELL_TEXT 104

  Cell contains a text value.
- #define FREEXL_CELL_SST_TEXT 105

  Cell contains a reference to a Single String Table entry (BIFF8)
- #define FREEXL_CELL_DATE 106

  Cell contains a number intended to represent a date.
- #define FREEXL_CELL_DATETIME 107

  Cell contains a number intended to represent a date and time.
- #define FREEXL_CELL_TIME 108

  Cell contains a number intended to represent a time.
- #define FREEXL_CFBF_VERSION 32001

  Information query for CFBF version.
- #define FREEXL_CFBF_SECTOR_SIZE 32002

  Information query for CFBF sector size.
- #define FREEXL_CFBF_FAT_COUNT 32003

  Information query for CFBF FAT entry count.

- #define FREEXL_BIFF_VERSION 32005

  Information query for BIFF version.
- #define FREEXL_BIFF_MAX_RECSIZE 32006

  Information query for BIFF maximum record size.
- #define FREEXL_BIFF_DATEMODE 32007

  Information query for BIFF date mode.
- #define FREEXL_BIFF_PASSWORD 32008

  Information query for BIFF password protection state.
- #define FREEXL_BIFF_CODEPAGE 32009

  Information query for BIFF character encoding.
- #define FREEXL_BIFF_SHEET_COUNT 32010

  Information query for BIFF sheet count.
- #define FREEXL_BIFF_STRING_COUNT 32011

  Information query for BIFF Single String Table entry count (BIFF8)
- #define FREEXL_BIFF_FORMAT_COUNT 32012

  Information query for BIFF format count.
- #define FREEXL_BIFF_XF_COUNT 32013

  Information query for BIFF extended format count.
- #define FREEXL_OK 0

  No error, success.
- #define FREEXL_FILE_NOT_FOUND -1

  .xls file does not exist or is not accessible for reading
- #define FREEXL_NULL_HANDLE -2

  Null xls_handle argument.
- #define FREEXL_INVALID_HANDLE -3

  Invalid xls_handle argument.
- #define FREEXL_INSUFFICIENT_MEMORY -4

  some kind of memory allocation failure
- #define FREEXL_NULL_ARGUMENT -5

  an unexpected null argument
- #define FREEXL_INVALID_INFO_ARG -6

  invalid "what" parameter
- #define FREEXL_INVALID_CFBF_HEADER -7

  the .xls file does not contain a valid CFBF header
- #define FREEXL_CFBF_READ_ERROR -8

  Read error.
- #define FREEXL_CFBF_SEEK_ERROR -9

  Seek error.
- #define FREEXL_CFBF_INVALID_SIGNATURE -10

  The .xls file does contain a CFBF header, but the header is broken or corrupted in some way.
- #define FREEXL_CFBF_INVALID_SECTOR_SIZE -11

  The .xls file does contain a CFBF header, but the header is broken or corrupted in some way.
- #define FREEXL_CFBF_EMPTY_FAT_CHAIN -12

  The .xls file does contain a CFBF header, but the header is broken or corrupted in some way.
- #define FREEXL_CFBF_ILLEGAL_FAT_ENTRY -13

  The file contains an invalid File Allocation Table record.
- #define FREEXL_BIFF_INVALID_BOF -14

  The file contains an invalid BIFF format entry.
- #define FREEXL_BIFF_INVALID_SST -15

  The file contains an invalid Single String Table.
- #define FREEXL_BIFF_ILLEGAL_SST_INDEX -16

The requested Single String Table entry is not available.

- #define FREEXL_BIFF_WORKBOOK_NOT_FOUND -17

  BIFF does not contain a valid workbook.
- #define FREEXL_BIFF_ILLEGAL_SHEET_INDEX -18

  The requested worksheet is not available in the workbook.
- #define FREEXL_BIFF_UNSELECTED_SHEET -19

  There is no currently active worksheet.
- #define FREEXL_INVALID_CHARACTER -20

  Charset conversion detected an illegal character (not within the declared charset)
- #define FREEXL_UNSUPPORTED_CHARSET -21

  The requested charset conversion is not available.
- #define FREEXL_ILLEGAL_CELL_ROW_COL -22

  The requested cell is outside the valid range for the sheet.
- #define FREEXL_ILLEGAL_RK_VALUE -23

  Conversion of the RK value failed.
- #define FREEXL_ILLEGAL_MULRK_VALUE -23

  Conversion of the MULRK value failed.
- #define FREEXL_INVALID_MINI_STREAM -24

  The MiniFAT stream is invalid.
- #define FREEXL_CFBF_ILLEGAL_MINI_FAT_ENTRY -25

  The MiniFAT stream contains an invalid entry.
- #define FREEXL_CRAFTED_FILE -26

  A severely corrupted file (may be purposely crafted for malicious purposes) has been detected.

Typedefs

- typedef struct FreeXL_CellValue_str FreeXL_CellValue

  Typedef for cell value structure.

Functions

- FREEXL_DECLARE const char * freexl_version (void)

  Return the current library version.
- FREEXL_DECLARE int freexl_open (const char *path, const void **xls_handle)

  Open the .xls file, preparing for future functions.
- FREEXL_DECLARE int freexl_open_info (const char *path, const void **xls_handle)

  Open the .xls file for metadata query only.
- FREEXL_DECLARE int freexl_close (const void *xls_handle)

  Close the .xls file and releasing any allocated resource.
- FREEXL_DECLARE int freexl_get_info (const void *xls_handle, unsigned short what, unsigned int *info)

  Query general information about the Workbook and Worksheets.
- FREEXL_DECLARE int freexl_get_worksheet_name (const void *xls_handle, unsigned short sheet_index, const char **string)

  Query worksheet name.
- FREEXL_DECLARE int freexl_select_active_worksheet (const void *xls_handle, unsigned short sheet_↩ index)

  Set the currently active worksheets.
- FREEXL_DECLARE int freexl_get_active_worksheet (const void *xls_handle, unsigned short *sheet_index)

  Query the currently active worksheet index.

- FREEXL_DECLARE int freexl_worksheet_dimensions (const void *xls_handle, unsigned int *rows, unsigned short *columns)

  Query worksheet dimensions.
- FREEXL_DECLARE int freexl_get_SST_string (const void *xls_handle, unsigned short string_index, const char **string)

  Retrieve string entries from SST.
- FREEXL_DECLARE int freexl_get_FAT_entry (const void *xls_handle, unsigned int sector_index, unsigned int *next_sector_index)

  Retrieve FAT entries from FAT chain.
- FREEXL_DECLARE int freexl_get_cell_value (const void *xls_handle, unsigned int row, unsigned short column, FreeXL_CellValue *value)

  Retrieve individual cell values from the currently active worksheet.

9.1.1 Detailed Description

Function declarations and constants for FreeXL library.

9.1.2 Macro Definition Documentation

9.1.2.1 FREEXL_BIFF_UNSELECTED_SHEET

```
#define FREEXL_BIFF_UNSELECTED_SHEET -19
```

There is no currently active worksheet.

Possibly a forgotten call to freexl_select_active_worksheet()

9.1.2.2 FREEXL_CFBF_ILLEGAL_MINI_FAT_ENTRY

```
#define FREEXL_CFBF_ILLEGAL_MINI_FAT_ENTRY -25
```

The MiniFAT stream contains an invalid entry.

Possibly a corrupt file.

9.1.2.3 FREEXL_CFBF_READ_ERROR

```
#define FREEXL_CFBF_READ_ERROR -8
```

Read error.

Usually indicates a corrupt or invalid .xls file

9.1.2.4 FREEXL_CFBF_SEEK_ERROR

```
#define FREEXL_CFBF_SEEK_ERROR -9
```

Seek error.

Usually indicates a corrupt or invalid .xls file

9.1.2.5 FREEXL_ILLEGAL_MULRK_VALUE

```
#define FREEXL_ILLEGAL_MULRK_VALUE -23
```

Conversion of the MULRK value failed.

Possibly a corrupt file or a bug in FreeXL.

9.1.2.6 FREEXL_ILLEGAL_RK_VALUE

```
#define FREEXL_ILLEGAL_RK_VALUE -23
```

Conversion of the RK value failed.

Possibly a corrupt file or a bug in FreeXL.

9.1.2.7 FREEXL_INVALID_MINI_STREAM

```
#define FREEXL_INVALID_MINI_STREAM -24
```

The MiniFAT stream is invalid.

Possibly a corrupt file.

9.1.3 Typedef Documentation

9.1.3.1 FreeXL_CellValue

```
typedef struct FreeXL_CellValue_str FreeXL_CellValue
```

Typedef for cell value structure.

See also

FreeXL_CellValue_str

9.1.4 Function Documentation

9.1.4.1 freexl_close()

```
FREEXL_DECLARE int freexl_close (
            const void * xls_handle )
```

Close the .xls file and releasing any allocated resource.

Parameters

xls_handle	the handle previously returned by freexl_open()

Returns

FREEXL_OK will be returned on success

Note

After calling freexl_close() any related resource will be released, and the handle will no longer be valid.

Examples

test_xl.c, and xl2sql.c.

9.1.4.2 freexl_get_active_worksheet()

```
FREEXL_DECLARE int freexl_get_active_worksheet (
             const void * xls_handle,
             unsigned short * sheet_index )
```

Query the currently active worksheet index.

Parameters

xls_handle	the handle previously returned by freexl_open()
sheet_index	the index corresponding to the currently active Worksheet (return value)

Returns

FREEXL_OK will be returned on success

See also

freexl_select_active_worksheet() for how to select the worksheet

Examples

test_xl.c.

9.1.4.3 freexl_get_cell_value()

```
FREEXL_DECLARE int freexl_get_cell_value (
             const void * xls_handle,
             unsigned int row,
             unsigned short column,
             FreeXL_CellValue * value )
```

Retrieve individual cell values from the currently active worksheet.

Parameters

xls_handle	the handle previously returned by freexl_open()
row	row number of the cell to query (zero base)
column	column number of the cell to query (zero base)
value	the cell type and value (return value)

Returns

FREEXL_OK will be returned on success

Examples

xl2sql.c.

9.1.4.4 freexl_get_FAT_entry()

```
FREEXL_DECLARE int freexl_get_FAT_entry (
            const void * xls_handle,
            unsigned int sector_index,
            unsigned int * next_sector_index )
```

Retrieve FAT entries from FAT chain.

Parameters

xls_handle	the handle previously returned by freexl_open()
sector_index	the index identifying the Sector entry (base 0).
next_sector_index	the index identifying the next Sector to be accessed in logical order (return value).

Note

The following values imply special meaning:

- 0xffffffff free / unused sector
- 0xfffffffe end of chain
- 0xfffffffd sector used by FAT (map of sectors)
- 0xfffffffc double-indirect FAT sector (map of FAT sectors)

Returns

FREEXL_OK will be returned on success

Note

This function is not normally required, since FreeXL will handle FAT table entries transparent to the user. It is mainly intended for debugging purposes.

Examples

test_xl.c.

9.1.4.5 freexl_get_info()

```
FREEXL_DECLARE int freexl_get_info (
             const void * xls_handle,
             unsigned short what,
             unsigned int * info )
```

Query general information about the Workbook and Worksheets.

Parameters

xls_handle	the handle previously returned by freexl_open()
what	the info to be queried.
info	the corresponding information value (return value)

Note

FREEXL_UNKNOWN will be returned in `info` if the information is not available, not appropriate or not supported for the file type.

Returns

FREEXL_OK will be returned on success

Valid values for `what` are:

- FREEXL_CFBF_VERSION (returning FREEXL_CFBF_VER_3 or FREEXL_CFBF_VER_4)
- FREEXL_CFBF_SECTOR_SIZE (returning FREEXL_CFBF_SECTOR_512 or FREEXL_CFBF_SECTOR↩ _4096)
- FREEXL_CFBF_FAT_COUNT (returning the total count of FAT entries in the file)
- FREEXL_BIFF_VERSION (return one of FREEXL_BIFF_VER_2, FREEXL_BIFF_VER_3, FREEXL_BIFF↩ _VER_4, FREEXL_BIFF_VER_5, FREEXL_BIFF_VER_8)
- FREEXL_BIFF_MAX_RECSIZE (returning FREEXL_BIFF_MAX_RECSZ_2080 or FREEXL_BIFF_MAX_↩ RECSZ_8224)
- FREEXL_BIFF_DATEMODE (returning FREEXL_BIFF_DATEMODE_1900 or FREEXL_BIFF_DATEMODE↩ _1904)
- FREEXL_BIFF_PASSWORD (returning FREEXL_BIFF_OBFUSCATED or FREEXL_BIFF_PLAIN)
- FREEXL_BIFF_CODEPAGE (returning FREEXL_BIFF_ASCII, one of FREEXL_BIFF_CP∗,FREEXL_BIFF↩ _UTF16LE or FREEXL_BIFF_MACROMAN)
- FREEXL_BIFF_SHEET_COUNT (returning the total number of worksheets)
- FREEXL_BIFF_STRING_COUNT (returning the total number of Single String Table entries)
- FREEXL_BIFF_FORMAT_COUNT (returning the total number of format entries)
- FREEXL_BIFF_XF_COUNT (returning the number of extended format entries)

Examples

test_xl.c, and xl2sql.c.

9.1.4.6 freexl_get_SST_string()

```
FREEXL_DECLARE int freexl_get_SST_string (
             const void * xls_handle,
             unsigned short string_index,
             const char ** string )
```

Retrieve string entries from SST.

Parameters

xls_handle	the handle previously returned by freexl_open()
string_index	the index identifying the String entry (base 0).
string	the corresponding String value (return value)

Returns

FREEXL_OK will be returned on success

Note

This function is not normally required, since freexl_get_cell_value will return the string where appropriate. It is mainly intended for debugging purposes.

Examples

test_xl.c.

9.1.4.7 freexl_get_worksheet_name()

```
FREEXL_DECLARE int freexl_get_worksheet_name (
             const void * xls_handle,
             unsigned short sheet_index,
             const char ** string )
```

Query worksheet name.

Parameters

xls_handle	the handle previously returned by freexl_open()
sheet_index	the index identifying the worksheet (base 0)
string	the name of the worksheet (return value)

Returns

FREEXL_OK will be returned on success

Examples

test_xl.c, and xl2sql.c.

9.1.4.8 freexl_open()

```
FREEXL_DECLARE int freexl_open (
            const char * path,
            const void ** xls_handle )
```

Open the .xls file, preparing for future functions.

Parameters

path	full or relative pathname of the input .xls file.
xls_handle	an opaque reference (handle) to be used in each subsequent function (return value).

Returns

FREEXL_OK will be returned on success, otherwise any appropriate error code on failure.

Note

You are expected to freexl_close() even on failure, so as to correctly release any dynamic memory allocation.

Examples

test_xl.c, and xl2sql.c.

9.1.4.9 freexl_open_info()

```
FREEXL_DECLARE int freexl_open_info (
            const char * path,
            const void ** xls_handle )
```

Open the .xls file for metadata query only.

This is similar to freexl_open(), except that an abbreviated parsing step is performed. This makes it faster, but does not support queries for cell values.

Parameters

path	full or relative pathname of the input .xls file.
xls_handle	an opaque reference (handle) to be used in each subsequent function (return value).

Returns

FREEXL_OK will be returned on success, otherwise any appropriate error code on failure.

Note

You are expected to freexl_close() even on failure, so as to correctly release any dynamic memory allocation.

9.1.4.10 freexl_select_active_worksheet()

```
FREEXL_DECLARE int freexl_select_active_worksheet (
             const void * xls_handle,
             unsigned short sheet_index )
```

Set the currently active worksheets.

Within a FreeXL handle, only one worksheet can be active at a time. Functions that fetch data are implictly working on the selected worksheet.

Parameters

xls_handle	the handle previously returned by freexl_open()
sheet_index	the index identifying the worksheet (base 0)

Returns

FREEXL_OK will be returned on success

Examples

test_xl.c, and xl2sql.c.

9.1.4.11 freexl_version()

```
FREEXL_DECLARE const char* freexl_version (
             void  )
```

Return the current library version.

Returns

the version string.

9.1.4.12 freexl_worksheet_dimensions()

```
FREEXL_DECLARE int freexl_worksheet_dimensions (
             const void * xls_handle,
             unsigned int * rows,
             unsigned short * columns )
```

Query worksheet dimensions.

This function returns the number of rows and columns for the currently selected worksheet.

Parameters

xls_handle	the handle previously returned by freexl_open()
rows	the total row count (return value)
columns	the total column count (return value)

Returns

FREEXL_OK will be returned on success

Note

Worksheet dimensions are zero based, so if you have a worksheet that is four columns and two rows (i.e. from A1 in the top left corner to B4 in the bottom right corner), this will return rows equal to 1 and columns equal to 3). This is to support C style looping.

Examples

test_xl.c, and xl2sql.c.

Chapter 10

Example Documentation

10.1 test_xl.c

test_xl.c is a simple demonstration and diagnostic tool for the Excel (.xls) file format.This sample code provides an example of:

- opening the .xls file
- querying general information
- querying Workbooks, SST entries and FAT entries
- error handling
- closing the .xls file when no further operations are required

Here is an example of a typical run:

```
./test_xl multi.xls

Excel document: multi.xls
============================================================
CFBF Version .........: 3
CFBF Sector size ....: 512
CFBF FAT entries ....: 128
BIFF Version .........: 8 [Excel 98/XP/2003/2007/2010]
BIFF Max record size : 8224
BIFF DateMode ........: 0 [day#1 = '1900-01-01']
BIFF Password/Crypted: NO, clear data
BIFF CodePage ........: UTF-16LE [Unicode]
BIFF Worksheets .....: 2
BIFF SST entries ....: 24
BIFF Formats .........: 2
BIFF eXtendedFormats : 24

Worksheets:
============================================================
  0] I'm a Worsheet
        ok, Worksheet succesfully selected: currently active: 0
        12 Rows X 7 Columns

  1] Yet another
        ok, Worksheet succesfully selected: currently active: 1
        302 Rows X 4 Columns
```

Here is another example. Note that this earlier version (Excel 3.0) format does not use the CFBF container, so no information is provided for the first three entries.

```
# ./test_xl v3sample.xls

Excel document: v3sample.xls
============================================================
CFBF Version .........: UNKNOWN
CFBF Sector size ....: UNKNOWN
CFBF FAT entries ....: 0
BIFF Version .........: 3 [Excel 3.0]
BIFF Max record size : UNKNOWN
BIFF DateMode .......: 0 [day#1 = '1900-01-01']
BIFF Password/Crypted: NO, clear data
BIFF CodePage .......: CP1252 [Windows Latin 1]
BIFF Worksheets .....: 1
BIFF Formats .........: 21
BIFF eXtendedFormats : 25

Worksheets:
============================================================
  0] Worksheet
       ok, Worksheet succesfully selected: currently active: 0
       17 Rows X 6 Columns
```

For more information, or to aid with debugging, you can specify a -verbose flag, as shown in this example:

```
# ./test_xl multi.xls -verbose

Excel document: multi.xls
============================================================
...

Worksheets:
============================================================
...

SST [Shared String Table]:
============================================================
       0] uno
       1] one
       2] due
       3] two
       4] tre
       5] three
...
      18] dieci
      19] ten
      20] undici
      21] eleven
      22] dodici
      23] twelve

FAT entries [File Allocation Table]:
============================================================
       0 -> 0xfffffffe FATSECT
       1 -> 0xffffffff FREESECT
       2 ->        3
       3 ->        4
...
      36 ->       37
      37 ->       38
      38 -> 0xfffffffe ENDOFCHAIN
      39 -> 0xfffffffe ENDOFCHAIN
      40 ->       41
      41 -> 0xfffffffe ENDOFCHAIN
      42 ->       43
      43 -> 0xfffffffe ENDOFCHAIN
      44 -> 0xffffffff FREESECT
...
     127 -> 0xffffffff FREESECT
```

```
/*
/ test_xl.c
/
/ FreeXL Sample code
/
/ Author: Sandro Furieri a.furieri@lqt.it
/
/ ------------------------------------------------------------------------------
/
/ Version: MPL 1.1/GPL 2.0/LGPL 2.1
/
/ The contents of this file are subject to the Mozilla Public License Version
/ 1.1 (the "License"); you may not use this file except in compliance with
/ the License. You may obtain a copy of the License at
/ http://www.mozilla.org/MPL/
/
/ Software distributed under the License is distributed on an "AS IS" basis,
/ WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License
/ for the specific language governing rights and limitations under the
/ License.
/
/ The Original Code is the FreeXL library
/
/ The Initial Developer of the Original Code is Alessandro Furieri
/
/ Portions created by the Initial Developer are Copyright (C) 2011
/ the Initial Developer. All Rights Reserved.
/
/ Contributor(s):
/ Brad Hards
/
/ Alternatively, the contents of this file may be used under the terms of
/ either the GNU General Public License Version 2 or later (the "GPL"), or
/ the GNU Lesser General Public License Version 2.1 or later (the "LGPL"),
/ in which case the provisions of the GPL or the LGPL are applicable instead
/ of those above. If you wish to allow use of your version of this file only
/ under the terms of either the GPL or the LGPL, and not to allow others to
/ use your version of this file under the terms of the MPL, indicate your
/ decision by deleting the provisions above and replace them with the notice
/ and other provisions required by the GPL or the LGPL. If you do not delete
/ the provisions above, a recipient may use your version of this file under
/ the terms of any one of the MPL, the GPL or the LGPL.
/
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "freexl.h"
int
main (int argc, char *argv[])
{
    const void *handle;
    int ret;
    unsigned int info;
    unsigned int fat_count;
    unsigned int sst_count;
    unsigned int worksheet_count;
    unsigned int format_count;
    unsigned int xf_count;
    unsigned int idx;
    unsigned int next_sector;
    int biff_v8 = 0;
    const char *utf8_string;
    int verbose = 0;
    if (argc == 2 || argc == 3)
      {
          if (argc == 3)
            {
                if (strcmp (argv[2], "-verbose") == 0)
                    verbose = 1;
            }
      }
    else
      {
          fprintf (stderr, "usage: text_xl path.xls [-verbose]\n");
          return -1;
      }
/* opening the .XLS file [Workbook] */
    ret = freexl_open (argv[1], &handle);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "OPEN ERROR: %d\n", ret);
          return -1;
      }
/*
 * reporting .XLS information
 */
```

```
    printf ("\nExcel document: %s\n", argv[1]);
    printf ("=========================================================\n");
/* CFBF version */
    ret = freexl_get_info (handle, FREEXL_CFBF_VERSION, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_CFBF_VERSION] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_CFBF_VER_3:
          printf ("CFBF Version ........: 3\n");
          break;
      case FREEXL_CFBF_VER_4:
          printf ("CFBF Version ........: 4\n");
          break;
      case FREEXL_UNKNOWN:
          printf ("CFBF Version ........: UNKNOWN\n");
          break;
      };
/* CFBF sector size */
    ret = freexl_get_info (handle, FREEXL_CFBF_SECTOR_SIZE, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_CFBF_SECTOR_SIZE] Error: %d\n",
                   ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_CFBF_SECTOR_512:
          printf ("CFBF Sector size ....: 512\n");
          break;
      case FREEXL_CFBF_SECTOR_4096:
          printf ("CFBF Sector size ....: 4096\n");
          break;
      case FREEXL_UNKNOWN:
          printf ("CFBF Sector size ....: UNKNOWN\n");
          break;
      };
/* CFBF FAT entries */
    ret = freexl_get_info (handle, FREEXL_CFBF_FAT_COUNT, &fat_count);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_CFBF_FAT_COUNT] Error: %d\n", ret);
          goto stop;
      }
    printf ("CFBF FAT entries ....: %u\n", fat_count);
/* BIFF version */
    ret = freexl_get_info (handle, FREEXL_BIFF_VERSION, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_VERSION] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_BIFF_VER_2:
          printf ("BIFF Version ........: 2 [Excel 2.0]\n");
          break;
      case FREEXL_BIFF_VER_3:
          printf ("BIFF Version ........: 3 [Excel 3.0]\n");
          break;
      case FREEXL_BIFF_VER_4:
          printf ("BIFF Version ........: 4 [Excel 4.0]\n");
          break;
      case FREEXL_BIFF_VER_5:
          printf ("BIFF Version ........: 5 [Excel 5.0 / Excel 95]\n");
          break;
      case FREEXL_BIFF_VER_8:
          printf ("BIFF Version ........: 8 [Excel 98/XP/2003/2007/2010]\n");
          biff_v8 = 1;
          break;
      case FREEXL_UNKNOWN:
          printf ("BIFF Version ........: UNKNOWN\n");
          break;
      };
/* BIFF max record size */
    ret = freexl_get_info (handle, FREEXL_BIFF_MAX_RECSIZE, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_MAX_RECSIZE] Error: %d\n",
                   ret);
          goto stop;
      }
    switch (info)
```

```
      {
      case FREEXL_BIFF_MAX_RECSZ_2080:
          printf ("BIFF Max record size : 2080\n");
          break;
      case FREEXL_BIFF_MAX_RECSZ_8224:
          printf ("BIFF Max record size : 8224\n");
          break;
      case FREEXL_UNKNOWN:
          printf ("BIFF Max record size : UNKNOWN\n");
          break;
      };
/* BIFF DateMode */
    ret = freexl_get_info (handle, FREEXL_BIFF_DATEMODE, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_DATEMODE] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_BIFF_DATEMODE_1900:
          printf ("BIFF DateMode .......: 0 [day#1 = '1900-01-01']\n");
          break;
      case FREEXL_BIFF_DATEMODE_1904:
          printf ("BIFF DateMode .......: 1 [day#1 = '1904-01-02']\n");
          break;
      case FREEXL_UNKNOWN:
          printf ("BIFF DateMode .......: UNKNOWN\n");
          break;
      };
/* BIFF Obfuscated */
    ret = freexl_get_info (handle, FREEXL_BIFF_PASSWORD, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_PASSWORD] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_BIFF_OBFUSCATED:
          printf ("BIFF Password/Crypted: YES, obfuscated (not accessible)\n");
          break;
      case FREEXL_BIFF_PLAIN:
          printf ("BIFF Password/Crypted: NO, clear data\n");
          break;
      case FREEXL_UNKNOWN:
          printf ("BIFF Password/Crypted: UNKNOWN\n");
          break;
      };
/* BIFF CodePage */
    ret = freexl_get_info (handle, FREEXL_BIFF_CODEPAGE, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_CODEPAGE] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_BIFF_ASCII:
          printf ("BIFF CodePage .......: ASCII\n");
          break;
      case FREEXL_BIFF_CP437:
          printf ("BIFF CodePage .......: CP437 [OEM United States]\n");
          break;
      case FREEXL_BIFF_CP720:
          printf ("BIFF CodePage .......: CP720 [Arabic (DOS)]\n");
          break;
      case FREEXL_BIFF_CP737:
          printf ("BIFF CodePage .......: CP737 [Greek (DOS)]\n");
          break;
      case FREEXL_BIFF_CP775:
          printf ("BIFF CodePage .......: CP775 [Baltic (DOS)]\n");
          break;
      case FREEXL_BIFF_CP850:
          printf ("BIFF CodePage .......: CP850 [Western European (DOS)]\n");
          break;
      case FREEXL_BIFF_CP852:
          printf ("BIFF CodePage .......: CP852 [Central European (DOS)]\n");
          break;
      case FREEXL_BIFF_CP855:
          printf ("BIFF CodePage .......: CP855 [OEM Cyrillic]\n");
          break;
      case FREEXL_BIFF_CP857:
          printf ("BIFF CodePage .......: CP857 [Turkish (DOS)]\n");
          break;
      case FREEXL_BIFF_CP858:
          printf ("BIFF CodePage .......: CP858 [OEM Multilingual Latin I]\n");
```

```
        break;
    case FREEXL_BIFF_CP860:
        printf ("BIFF CodePage .......: CP860 [Portuguese (DOS)]\n");
        break;
    case FREEXL_BIFF_CP861:
        printf ("BIFF CodePage .......: CP861 [Icelandic (DOS)]\n");
        break;
    case FREEXL_BIFF_CP862:
        printf ("BIFF CodePage .......: CP862 [Hebrew (DOS)]\n");
        break;
    case FREEXL_BIFF_CP863:
        printf ("BIFF CodePage .......: CP863 [French Canadian (DOS)]\n");
        break;
    case FREEXL_BIFF_CP864:
        printf ("BIFF CodePage .......: CP864 [Arabic (864)]\n");
        break;
    case FREEXL_BIFF_CP865:
        printf ("BIFF CodePage .......: CP865 [Nordic (DOS)]\n");
        break;
    case FREEXL_BIFF_CP866:
        printf ("BIFF CodePage .......: CP866 [Cyrillic (DOS)]\n");
        break;
    case FREEXL_BIFF_CP869:
        printf ("BIFF CodePage .......: CP869 [Greek, Modern (DOS)]\n");
        break;
    case FREEXL_BIFF_CP874:
        printf ("BIFF CodePage .......: CP874 [Thai (Windows)]\n");
        break;
    case FREEXL_BIFF_CP932:
        printf ("BIFF CodePage .......: CP932 [Japanese (Shift-JIS)]\n");
        break;
    case FREEXL_BIFF_CP936:
        printf
            ("BIFF CodePage .......: CP936 [Chinese Simplified (GB2312)]\n");
        break;
    case FREEXL_BIFF_CP949:
        printf ("BIFF CodePage .......: CP949 [Korean]\n");
        break;
    case FREEXL_BIFF_CP950:
        printf
            ("BIFF CodePage .......: CP950 [Chinese Traditional (Big5)]\n");
        break;
    case FREEXL_BIFF_UTF16LE:
        printf ("BIFF CodePage .......: UTF-16LE [Unicode]\n");
        break;
    case FREEXL_BIFF_CP1250:
        printf ("BIFF CodePage .......: CP1250 [Windows Central Europe]\n");
        break;
    case FREEXL_BIFF_CP1251:
        printf ("BIFF CodePage .......: CP1251 [Windows Cyrillic]\n");
        break;
    case FREEXL_BIFF_CP1252:
        printf ("BIFF CodePage .......: CP1252 [Windows Latin 1]\n");
        break;
    case FREEXL_BIFF_CP1253:
        printf ("BIFF CodePage .......: CP1253 [Windows Greek]\n");
        break;
    case FREEXL_BIFF_CP1254:
        printf ("BIFF CodePage .......: CP1254 [Windows Turkish]\n");
        break;
    case FREEXL_BIFF_CP1255:
        printf ("BIFF CodePage .......: CP1255 [Windows Hebrew]\n");
        break;
    case FREEXL_BIFF_CP1256:
        printf ("BIFF CodePage .......: CP1256 [Windows Arabic]\n");
        break;
    case FREEXL_BIFF_CP1257:
        printf ("BIFF CodePage .......: CP1257 [Windows Baltic]\n");
        break;
    case FREEXL_BIFF_CP1258:
        printf ("BIFF CodePage .......: CP1258 [Windows Vietnamese]\n");
        break;
    case FREEXL_BIFF_CP1361:
        printf ("BIFF CodePage .......: CP1361 [Korean (Johab)]\n");
        break;
    case FREEXL_BIFF_MACROMAN:
        printf ("BIFF CodePage .......: MacRoman\n");
        break;
    case FREEXL_UNKNOWN:
        printf ("BIFF CodePage .......: UNKNOWN\n");
        break;
    };
/* BIFF Worksheet entries */
    ret = freexl_get_info (handle, FREEXL_BIFF_SHEET_COUNT, &worksheet_count);
    if (ret != FREEXL_OK)
      {
        fprintf (stderr, "GET-INFO [FREEXL_BIFF_SHEET_COUNT] Error: %d\n",
```

```
                  ret);
          goto stop;
      }
    printf ("BIFF Worksheets .....: %u\n", worksheet_count);
    if (biff_v8)
      {
          /* BIFF SST entries */
          ret = freexl_get_info (handle, FREEXL_BIFF_STRING_COUNT, &sst_count);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr,
                         "GET-INFO [FREEXL_BIFF_STRING_COUNT] Error: %d\n",
                         ret);
                goto stop;
            }
          printf ("BIFF SST entries ....: %u\n", sst_count);
      }
/* BIFF Format entries */
    ret = freexl_get_info (handle, FREEXL_BIFF_FORMAT_COUNT, &format_count);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_FORMAT_COUNT] Error: %d\n",
                   ret);
          goto stop;
      }
    printf ("BIFF Formats ........: %u\n", format_count);
/* BIFF XF entries */
    ret = freexl_get_info (handle, FREEXL_BIFF_XF_COUNT, &xf_count);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_XF_COUNT] Error: %d\n", ret);
          goto stop;
      }
    printf ("BIFF eXtendedFormats : %u\n", xf_count);
    printf
        ("\nWorksheets:\n============================================================\n");
    for (idx = 0; idx < worksheet_count; idx++)
      {
          /* printing BIFF Worksheets entries */
          unsigned short active;
          unsigned int rows;
          unsigned short columns;
          ret = freexl_get_worksheet_name (handle, idx, &utf8_string);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "GET-WORKSHEET-NAME Error: %d\n", ret);
                goto stop;
            }
          if (utf8_string == NULL)
              printf ("%3u] NULL (unnamed)\n", idx);
          else
              printf ("%3u] %s\n", idx, utf8_string);
          ret = freexl_select_active_worksheet (handle, idx);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "SELECT-ACTIVE_WORKSHEET Error: %d\n", ret);
                goto stop;
            }
          ret = freexl_get_active_worksheet (handle, &active);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "GET-ACTIVE_WORKSHEET Error: %d\n", ret);
                goto stop;
            }
          printf
              ("\tok, Worksheet successfully selected: currently active: %u\n",
               active);
          ret = freexl_worksheet_dimensions (handle, &rows, &columns);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "WORKSHEET-DIMENSIONS Error: %d\n", ret);
                goto stop;
            }
          printf ("\t%u Rows X %u Columns\n\n", rows, columns);
      }
    if (!verbose)
        goto stop;
    if (biff_v8)
      {
          /* printing BIFF SST entries */
          printf
              ("\nSST [Shared String Table]:\n============================================================\n");
          for (idx = 0; idx < sst_count; idx++)
            {
                ret = freexl_get_SST_string (handle, idx, &utf8_string);
                if (ret != FREEXL_OK)
                  {
```

```
                    fprintf (stderr, "GET-SST-STRING Error: %d\n", ret);
                    goto stop;
                }
            if (utf8_string == NULL)
                printf ("%8u] NULL (empty string)\n", idx);
            else
                printf ("%8u] %s\n", idx, utf8_string);
        }
    }
  printf
      ("\nFAT entries [File Allocation
     Table]:\n============================================================\n");
  for (idx = 0; idx < fat_count; idx++)
    {
      /* printing each FAT entry */
      ret = freexl_get_FAT_entry (handle, idx, &next_sector);
      if (ret != FREEXL_OK)
        {
          fprintf (stderr, "GET-FAT-ENTRY Error: %d\n", ret);
          goto stop;
        }
      if (next_sector == 0xffffffff)
          printf ("%8u -> 0xffffffff FREESECT\n", idx);
      else if (next_sector == 0xfffffffe)
          printf ("%8u -> 0xfffffffe ENDOFCHAIN\n", idx);
      else if (next_sector == 0xfffffffd)
          printf ("%8u -> 0xfffffffe FATSECT\n", idx);
      else if (next_sector == 0xfffffffc)
          printf ("%8u -> 0xfffffffe DIFSECT\n", idx);
      else
          printf ("%8u -> %8u\n", idx, next_sector);
    }
stop:
/* closing the .XLS file [Workbook] */
  ret = freexl_close (handle);
  if (ret != FREEXL_OK)
    {
      fprintf (stderr, "CLOSE ERROR: %d\n", ret);
      return -1;
    }
  return 0;
}
```

10.2 xl2sql.c

xl2sql a simple tool that takes an .xls file as input, and generates a SQL script as output. You can then use the SQL script to load the extracted data info a SQLite / SpatiaLite database.

Here is a typical usage example:

```
./xl2sql comuni_italiani.xls >comuni.sql
spatialite italy.sqlite <comuni.sql
```

The first command will parse the .xls document, extracting any data and generating the corresponding SQL script. The second command will create and populate a database from the SQL script. When using xl2sql this way, the first worksheet will become database table xl_table_00, the second worksheet will become database table xl_table_01 and so on.

As an alternative, if you pass a second argument to xl2sql, this argument will be used as the table prefix. For example:

```
./xl2sql comuni_italiani.xls italia >comuni.sql
spatialite italy2.sqlite <comuni.sql
```

This will result in the tables being named italia_00, italia_01 and so on.

This sample code provides an example of:

- selecting a worksheet to be active
- retrieving cell values

```
/*
/ xl2sql.c
/
/ FreeXL Sample code
/
/ Author: Sandro Furieri a.furieri@lqt.it
/
/ ------------------------------------------------------------------------------
/
/ Version: MPL 1.1/GPL 2.0/LGPL 2.1
/
/ The contents of this file are subject to the Mozilla Public License Version
/ 1.1 (the "License"); you may not use this file except in compliance with
/ the License. You may obtain a copy of the License at
/ http://www.mozilla.org/MPL/
/
/ Software distributed under the License is distributed on an "AS IS" basis,
/ WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License
/ for the specific language governing rights and limitations under the
/ License.
/
/ The Original Code is the FreeXL library
/
/ The Initial Developer of the Original Code is Alessandro Furieri
/
/ Portions created by the Initial Developer are Copyright (C) 2011
/ the Initial Developer. All Rights Reserved.
/
/ Contributor(s):
/ Brad Hards
/
/ Alternatively, the contents of this file may be used under the terms of
/ either the GNU General Public License Version 2 or later (the "GPL"), or
/ the GNU Lesser General Public License Version 2.1 or later (the "LGPL"),
/ in which case the provisions of the GPL or the LGPL are applicable instead
/ of those above. If you wish to allow use of your version of this file only
/ under the terms of either the GPL or the LGPL, and not to allow others to
/ use your version of this file under the terms of the MPL, indicate your
/ decision by deleting the provisions above and replace them with the notice
/ and other provisions required by the GPL or the LGPL. If you do not delete
/ the provisions above, a recipient may use your version of this file under
/ the terms of any one of the MPL, the GPL or the LGPL.
/
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "freexl.h"
static void
make_table_name (const char *prefix, unsigned short index, char *table_name)
{
/* generating an SQL clean table name */
    char buf[2048];
    char *in = buf;
    char *out = table_name;
    sprintf (buf, "%s_%02u", prefix, index);
/* masking for SQL */
    *out++ = '"';
    while (*in != '\0')
      {
          if (*in == '"')
              *out++ = '"';
          *out++ = *in++;
      }
    *out++ = '"';
    *out = '\0';
}
static void
print_sql_string (const char *string)
{
/* printing a well formatted SQL string */
    const char *p = string;
    putchar ('\'');
    putchar (' ');
    putchar ('\"');
    while (*p != '\0')
      {
          if (*p == '\"')
            {
                /* masking any ' as " */
                putchar ('\"');
            }
          putchar (*p);
```

```
          p++;
      }
    putchar ('\"');
}
int
main (int argc, char *argv[])
{
    unsigned int worksheet_index;
    const char *table_prefix = "xl_table";
    char table_name[2048];
    const void *handle;
    int ret;
    unsigned int info;
    unsigned int max_worksheet;
    unsigned int rows;
    unsigned short columns;
    unsigned int row;
    unsigned short col;
    if (argc == 2 || argc == 3)
      {
          if (argc == 3)
              table_prefix = argv[2];
      }
    else
      {
          fprintf (stderr, "usage: xl2sql path.xls [table_prefix]\n");
          return -1;
      }
/* opening the .XLS file [Workbook] */
    ret = freexl_open (argv[1], &handle);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "OPEN ERROR: %d\n", ret);
          return -1;
      }
/* checking for Password (obfuscated/encrypted) */
    ret = freexl_get_info (handle, FREEXL_BIFF_PASSWORD, &info);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_PASSWORD] Error: %d\n", ret);
          goto stop;
      }
    switch (info)
      {
      case FREEXL_BIFF_PLAIN:
          break;
      case FREEXL_BIFF_OBFUSCATED:
      default:
          fprintf (stderr, "Password protected: (not accessible)\n");
          goto stop;
      };
/* querying BIFF Worksheet entries */
    ret = freexl_get_info (handle, FREEXL_BIFF_SHEET_COUNT, &max_worksheet);
    if (ret != FREEXL_OK)
      {
          fprintf (stderr, "GET-INFO [FREEXL_BIFF_SHEET_COUNT] Error: %d\n",
                   ret);
          goto stop;
      }
/* SQL output */
    printf ("--\n-- this SQL script was automatically created by xl2sql\n");
    printf ("--\n-- input .xls document was: %s\n--\n", argv[1]);
    printf ("\nBEGIN;\n\n");
    for (worksheet_index = 0; worksheet_index < max_worksheet;
         worksheet_index++)
      {
          const char *utf8_worsheet_name;
          make_table_name (table_prefix, worksheet_index, table_name);
          ret =
              freexl_get_worksheet_name (handle, worksheet_index,
                                         &utf8_worsheet_name);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "GET-WORKSHEET-NAME Error: %d\n", ret);
                goto stop;
            }
          /* selecting the active Worksheet */
          ret = freexl_select_active_worksheet (handle, worksheet_index);
          if (ret != FREEXL_OK)
            {
                fprintf (stderr, "SELECT-ACTIVE_WORKSHEET Error: %d\n", ret);
                goto stop;
            }
          /* dimensions */
          ret = freexl_worksheet_dimensions (handle, &rows, &columns);
          if (ret != FREEXL_OK)
            {
```

```
            fprintf (stderr, "WORKSHEET-DIMENSIONS Error: %d\n", ret);
            goto stop;
        }
      printf ("--\n-- creating a DB table\n");
      printf ("-- extracting data from Worksheet #%u: %s\n--\n",
              worksheet_index, utf8_worsheet_name);
      printf ("CREATE TABLE %s (\n", table_name);
      printf ("\trow_no INTEGER NOT NULL PRIMARY KEY");
      for (col = 0; col < columns; col++)
          printf (",\n\tcol_%03u MULTITYPE", col);
      printf (");\n");
      printf ("--\n-- populating the same table\n--\n");
      for (row = 0; row < rows; row++)
        {
            /* INSERT INTO statements */
            FreeXL_CellValue cell;
            printf ("INSERT INTO %s (row_no", table_name);
            for (col = 0; col < columns; col++)
                printf (", col_%03u", col);
            printf (") VALUES (%u", row);
            for (col = 0; col < columns; col++)
              {
                  ret = freexl_get_cell_value (handle, row, col, &cell);
                  if (ret != FREEXL_OK)
                    {
                        fprintf (stderr,
                                 "CELL-VALUE-ERROR (r=%u c=%u): %d\n", row,
                                 col, ret);
                        goto stop;
                    }
                  switch (cell.type)
                    {
                    case FREEXL_CELL_INT:
                        printf (", %d", cell.value.int_value);
                        break;
                    case FREEXL_CELL_DOUBLE:
                        printf (", %1.12f", cell.value.double_value);
                        break;
                    case FREEXL_CELL_TEXT:
                    case FREEXL_CELL_SST_TEXT:
                        print_sql_string (cell.value.text_value);
                        break;
                    case FREEXL_CELL_DATE:
                    case FREEXL_CELL_DATETIME:
                    case FREEXL_CELL_TIME:
                        printf (", '%s'", cell.value.text_value);
                        break;
                    case FREEXL_CELL_NULL:
                    default:
                        printf (", NULL");
                        break;
                    };
              }
            printf (");\n");
        }
      printf ("\n-- done: table end\n\n\n\n");
    }
  printf ("COMMIT;\n");
stop:
/* closing the .XLS file [Workbook] */
  ret = freexl_close (handle);
  if (ret != FREEXL_OK)
    {
        fprintf (stderr, "CLOSE ERROR: %d\n", ret);
        return -1;
    }
  return 0;
}
```

Index