BIBTOOL

A Tool to Manipulate BibTeX Files

C Programmers Manual

Gerd Neugebauer

Abstract

BibTool provides a library of useful C functions to manipulate BibTeX files. This library has been used to implement the BibTool program. This document describes This library and allows you to write C programs dealing with BibTeX files.

— This documentation is still in a rudimentary form and needs additional efforts. —

This file is part of BibTool Version 2.50

Gerd Neugebauer
Im Lerchelsböhl 5
64521 Groß-Gerau (Germany)

WWW: `http://www.gerd-neugebauer.de/`

Net: `gene@gerd-neugebauer.de`

Contents

1

Introduction

The BibTool C library provides functions to deal with BibTeX files. These functions are described in this document. Thus it should be fairly easy to write new C program which handle BibTeX files. The reader is assumed to be familiar with BibTeX files. this documentation will not repeat an introduction into BibTeX.

This documentation can not only be used to write new C programs dealing with BibTeX files but also to understand BibTool—The Program which serves as one example for using the BibToolC library. In any case it is essential to understand some of the underlying concepts. Thus it is vital to read some sections very carefully. Especially the section

The BibTool program uses the BibTool C library. Well, in fact it is the other way round. Historically the BibTool program was first and then the library has been extracted from it. Nevertheless the BibTool program can serve as an example how the BibTool C libary can be used.

1.1 The Module `main.c`

This is the BibTool main module. It contains the `main()` function which evaluates the command line arguments and proceeds accordingly. This means that usually resource files and BibTeX files are read and one or more BibTeX files are written.

This file makes use of the BibTool C library but is not part of it. For this purpose it has to provide certain functions which are expected by the library. These functions are:

```
save_input_file()
save_macro_file()
save_output_file()
```

The arguments and the expected behaviour of these functions is described below.

If you are trying to understand the implementation of BibTool the file `resource.h` plays a central rôle. Consult the description of this file for further details.

If you are trying to write your own program to manipulate BibTeX files then this file can serve as a starting point. But you should keep in mind that this file has grown over several years and it contains the full complexity of the BibTool program logic. Thus you can reduce this file drastically if you start playing around with the BibTool C library.

`static int keep_selected()` Function

Returns:

`int keep_xref()` Function

```
DB     db;
Record rec;
```

Undelete crossreferenced entries

Returns:

`int main()` Function

```
int   argc;          Number of arguments
char *argv[];        Array of arguments
```

This is the main function which is automatically called when the program is started. This function contains the overall program logic. It has to perform the appropriate initializations, evaluate command line arguments, and run the main loop.

Returns: 0 upon success. Usually a failure raises an exception which leads to an `exit()`. Thus this function does not need to signal an error to the calling environment.

`static int rec_gt_cased()` Function

Returns:

`static int rec_lt_cased()` Function

Returns:

`void save_input_file()` Function

`char *file;` *File name to save.*

The input file pipe is a dynamic array of strings. This fifo stack is used to store the input BibTeX files to be processed by BibTool.

This function is called to push an string into the pipe. If neccesary the array has to be allocated or enlarged. This is done in larger junks to avoid lots of calls to `realloc()`.

Returns: nothing

`void save_macro_file()` Function

`char *file;` *File name to save*

Simply feed the macro file name into the static variable. This function is useful since it can be called from rsc.c

Returns: nothing

`void save_output_file()` Function

`char * file;` *File name to save*

Simply feed the output file name into the static variable. This function is useful since it can be called from rsc.c

Returns: nothing

2

The BibTool C Library

3

Creating and Using the BibTool C Library

3.1 Creating the BibTool C Library

Creating the BibTool library should not be too hard. Mainly make BibTool in the main directory according to the instructions given there. As a side effect various object files are created. These object files—except the one for main.c—have to be put into the library.

For UNIX this is prepared in the makefile. Usually an invocation of make should be enough:

```
make libbib.a
```

This invocation of make is in fact the same as the following two commands:

```
ar r libbib.a $OFILES
ranlib libbib.a
```

Here `$OFILES` denotes the list of object files as described above. On some systems no ranlib program is present and needed. In this case the second command can be omitted.

For other operating systems I simply do not know how things work there. I would be grateful to receive descriptions what to do there.

3.2 Using the BibTool C Library

If you have written a program which uses the BibTool C Library you have to include the library into the linking list. In addition the directory where the library can be found has to be specified. On UNIX this can be done with the compiler switches `-l` and `-L` respectively. Thus consider you have a program named `mybib.c` and you have created the object file `mybib.o` for it. The linking step can be performed with the following command:

```
cc mybib.o -L$DIR -lbib -o mybib
```

Here `$DIR` denotes the path containing the file `libbib.a`. This path can be omitted if the library has been installed in a "standard" place like `/usr/lib`.

4

Coding Standards

Several tools are used for the development of BibTool. Mostly they are home grown—maybe they will be replaced by some wider used tools some day. Among those tools are indentation routines for Emacs to format the comments contained in the source. There is also a Lisp function to generate the function prototypes contained in the header files and sometimes in the C files as well. And finally there is a Program to extract the documentation from the source files and generate a printable manual.

All those support programs rely on standards for coding. Some of those standards have been develped independantly but should be used for consistency. In the following sections these coding standards are described.

4.1 K&R-C vs. ANSI-C

BibTool tries hard to be portable to wide variety of C systems. Thus it can not be assumed that an ANSI C compiler is at hand. As a consequence the function heads are written in the old style which is also tolerated by ANSI compliant compilers. This means that the argument types are given after the argument list.

Here it is essential that the arguments type declarations are given in the same order as the arguments of the function. Each type variable must have a new type declaration in a line by it's own. This feature is used by the program which extracts the function prototypes.

Those function heads are use to generate function prototypes which can be understood by ANSI-C compilers as well as by of K&R compilers. This is achieved by the od trick to introduce a macro which expands to nothing on the old compilers and to its aregument on ANSI compilers. This macro is defined appropriately according to the existence of the macro `__STDC__` which should indicate an ANSI compliant compiler.

Index