GNU Emacs BIBTEX Mode

— version 1.5 —

Pehong Chen*

Computer Science Division
University of California
Berkeley, CA 94720

October 27, 1986

* Sponsored in part by the State of California MICRO Fellowship, by the National Science Foundation under Grant MCS-8311787, and by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by Naval Electronic Systems Command, under Contract No. N00039-84-C-0089.

Contents

1 Introduction

BIBTEX-mode is part of an Emacs-based environment for editing TEX documents [2]. It is a GNU Emacs [6] interface to BIBTEX databases. A BIBTEX database is a file with the name suffix '`.bib`' which contains one or more bibliography entries. BIBTEX is a system designed jointly by Leslie Lamport, Howard Trickey, and Oren Patashnik (implementation is due to Patashnik) as a bibliography preprocessor for LATEX documents [4,5]. The GNU Emacs *TEX-mode* [1] makes it possible for BIBTEX to work on plain TEX [3] and $\mathcal{AMS}$-TEX [7] documents as well. *BIBTEX-mode* supports all fourteen BIBTEX bibliography entry types as built-in functions so that to insert a new entry the user only has to specify a type. A skeleton instance of the specified type will be generated automatically with the various fields left empty for the user to fill in. A set of supporting functions such as scrolling, field copying, entry duplicating, ..., etc. is provided to facilitate this content-filling process. Other major features of the mode include an extended abbreviation mechanism and a draft making facility.

BIBTEX-mode comprises four subsystems: `bibtex-mode.el`, `bibtex-ops.el`, `bibtex-misc.el`, and `bibtex-abv.el`. The first file contains the function `bibtex-mode` which defines such attributes as syntax entry modifications, key bindings, and global and buffer-specific variables for *BIBTEX-mode*. Also included in the file is the information for the built-in entries and the basic supporting functions shared by programs in other subsystems. This file will be loaded when the first `.bib` file is visited in an Emacs session.

Entry and field operations such as *scroll*, *copy*, *delete*, etc. are defined in file `bibtex-ops.el`. Functions defined in `bibtex-misc.el` fall into two categories: (1) A cleanup facility that deletes redundant information contained in the entry skeleton, such as banners, trailing commas, etc. (2) A draft making facility which prepares a formatted draft bibliography out of the current `.bib` file and allows one to preview and print it. Also, this draft making process can be used to debug the current `.bib` file. An error positioning mechanism allows one to locate error spots. Finally, the file `bibtex-abv.el` includes an extended abbreviation mechanism which allows one to browse and interpolate abbreviations defined in any files. Each of these three files is autoloaded whenever a function defined in it is invoked.

This document describes each user-level function available in version 1.5 of *BIBTEX-mode*. It assumes the reader knows the basics of Emacs and BIBTEX. The goal is to show you how to use the system with some instructive examples. The next chapter gives some general guidelines to making *BIBTEX-mode* work under your environment. Chapters 3 and 4 deal with the various entry and field operations, respectively. Chapter 5 covers the extended abbreviation mechanism. The draft making facility is explained in Chapter 6. Finally all user level functions are summarized in Chapter 9, followed by two sets of indices in the last two chapters.

Disclaimer. Although *BIBTEX-mode* has been extensively tested, there is no warranty that the functions described in this document are bug-free. The author does not accept any responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all.

Bugs/Comments. Bugs and comments on both the code and this document are welcome. Please send them via electronic mail to `phc@renoir.berkeley.edu`.

2 Installation and Startup

To get *BIBTEX-mode* autoloaded, add the following two lines of code to your `.emacs`:

```
(setq auto-mode-alist (cons '("\\.bib\$" .  bibtex-mode) auto-mode-alist))
(autoload 'bibtex-mode "bibtex-mode" "Major mode for editing BibTeX database  files" t)
```

You can also enter *BIBTEX-mode* manually by loading the file `bibtex-mode` first and then invoke the function `bibtex-mode` at the `M-x` prompt.

2.1 BIBTEX-mode Hook

The variable `bibtex-mode-hook` is the last object that gets evaluated whenever the function `bibtex-mode` is invoked, which means whatever defined in the hook overwrites the default. A typical `bibtex-mode-hook` is defined in the following way:

```
(setq bibtex-mode-hook
  (function
    (lambda ()
      <BODY>)))
```

For the convenience of version control (typing `bibtex-mode-version` at the `M-x` prompt returns the current *BIBTEX-mode* version), the user is advised to put his local changes in the hook rather than modifying the various files of *BIBTEX-mode* directly. The abbreviation table (see below) is one example that goes into the hook, your preferred key bindings may be another, and other *BIBTEX-mode* subsystems that you've developed can also be loaded from the hook.

2.2 Minor Mode for Abbreviations

There are no abbreviations defined in *BIBTEX-mode*, but the user can use `bibtex-mode-hook` to define his own abbreviations. For instance, a typical `.emacs` may contain a `bibtex-mode-hook` whose body is:

```
(define-mode-abbrev "tx"  "{\\TeX}")
(define-mode-abbrev "atx" "{\\AmSTeX})
(define-mode-abbrev "btx" "{\\BibTeX}")
(define-mode-abbrev "ltx" "{\\LaTeX}")
(define-mode-abbrev "stx" "{\\SliTeX}")
(abbrev-mode 1))))
```

Note that the `Abbrev` minor mode is set by invoking the function `abbrev-mode` with a positive number as argument. It can be reset by passing 0 as the argument.

The command **C-c C-a SPC** (`bibtex-abbrev-enable`) unconditionally enables the `Abbrev` minor mode. Conversely **C-c C-a DEL** (`bibtex-abbrev-disable`) disables it unconditionally.

2.3 Minor Mode for Auto-Filling

By default, *BIBTEX-mode* has the `Auto Fill` minor mode turned off. If this minor mode is desired, put

```
(auto-fill-mode 1)
```

in your `tex-mode-hook`. When the mode is set one can keep typing beyond the right margin without any explicit **RET** or **LFD** and the line will wrap around automatically. The variable `fill-column` is set to 78 in *BIBTEX-mode* instead of the Emacs default value of 70. If instead N is desired column boundary, put

```
(setq fill-column N)
```

in your hook.

The standard Emacs line wrapping is modified a bit such that the wrapped line is indented to the column where the text of the current field begins. Typing **LFD** (`bibtex-newline-indent`) in *BIBTEX-mode* has the same effect. We shall find this feature useful in Section 4.5. Incidently, **TAB** is a self-inserting character in *BIBTEX-mode* which simply moves the cursor to the next tab position.

The command **C-c LFD SPC** (`bibtex-autofill-enable`) enables the `Auto Fill` minor mode unconditionally. Conversely **C-c LFD DEL** (`bibtex-autofill-disable`) disables it unconditionally.

2.4 System Dependencies

There are some system-dependent spots in *BIBTEX-mode*. First, a couple of external programs are invoked inside the mode: `bibtex`, `tex`, `dvitool`†, and `lpr` are used in `bibtex-misc.el`. You need these programs to make *BIBTEX-mode* fully functional.

Second, some of the default settings in *BIBTEX-mode* may not be right for your local environments. You can redefine the variables involved in your `bibtex-mode-hook`. For instance, the last step of **C-c C-\ d** (`bibtex-make-draft`) will ask you if you want the draft printed. If so, it prompts you for the printer option. The default is the list `(ip, cx, dp, gp)` which represents the names of laser printers available to our group. If what you have is the list `(a, b, c)` with 'a' being the one you use most often, you could say

```
(setq bibtex-printer-list "(a, b, c)" "Printers available locally")
(setq bibtex-printer-default "a" "Default printer")
```

in your hook. Similarly if you have a different previewer called `previewtool` or if you are using a different printing scheme called `print`, you can alter the default values by saying

```
(setq bibtex-softcopy "previewtool" "My DVI previewer")
(setq bibtex-hardcopy "print" "My printing scheme")
```

in your `bibtex-mode-hook`.

† The program `dvitool` is a TEX DVI previewer running on the SUN workstation. It is available through Berkeley VORTEX distribution (dist-vortex@berkeley.edu).

2.5 Site Initialization

It is possible to setup a local *BIBTEX-mode* environment by redefining the variables in a startup file called `bibtex-init.el`. This file does not come with *BIBTEX-mode*. But if either the file or its compiled form (`bibtex-init.elc`) exists in `EMACSLOADPATH`, the command

```
(load "bibtex-init")
```

will be executed whenever the mode is invoked. Otherwise the default values remain intact unless they are redefined in the user's `bibtex-mode-hook`.

In effect this facility provides a site-wide initialization for everyone using the mode. Site-specific attributes like the speller, previewer, printers, etc., which would probably be different from the default settings but identical for most users in the community, can be redefined in this initialization file. For example, for site X the *BIBTEX-mode* administrator can put

```
(setq bibtex-printer-list "(a, b, c)" "Printers available at site X")
(setq bibtex-printer-default "a" "Default printer of site X")
(setq bibtex-softcopy "previewtool" "DVI previewer used at site X")
(setq bibtex-hardcopy "print" "printing/spooling scheme at site X")
```

in the file `bibtex-init.el`, thereby alleviating users from having to deal with these attributes in their individual `bibtex-mode-hook`'s.

3 Entry Operations

All of BIBTEX's fourteen standard bibliography entry types are supported by *BIBTEX-mode*. The following is a list of these entries.

`@article`	`@book`	`@booklet`	`@conference`
`@inbook`	`@incollection`	`@inproceedings`	`@manual`
`@masterthesis`	`@misc`	`@phdthesis`	`@proceedings`
`@techreport`	`@unpublished`		

3.1 Invoking Entries

To invoke a new entry, specify its type at the `M-x` prompt. Command completion can be used here. For example, the command `M-x@mi` **RET** will have a new entry of type `@MISC` inserted below the current entry. Here is an instance of the skeleton entry:

```
@MISC{█,
================================ OPTIONAL FIELDS ================================
     AUTHOR = {},
     TITLE = {},
     HOWPUBLISHED = {},
     MONTH = ,
     YEAR = {},
     NOTE = {}
}
```

The cursor will be placed at the comma position in the first line where the entry name is to be entered (as denoted by █).

3.2 Setting up Entries

Some of the entry settings may be changed. By default, entry and field delimiters are both a pair of curly braces (i.e. `{...}`) and the indentation for field labels is 5 places. Alternatively, BIBTEX accepts a pair of parentheses as entry delimiters (i.e. `(...)`) and a pair of double quotes (i.e. `"..."`) as field delimiters. The flag `bibtex-entry-use-parens`, if set non-nil, instructs *BIBTEX-mode* to use parentheses as entry delimiters. Similarly, the flag `bibtex-field-use-quotes`, if set non-nil, tells the system to use double quotes as field delimiters. The variable `bibtex-field-indent`, if assigned N, makes field indentation N places instead the default value of 5.

3.3 Scrolling Entries

The command **C-c c** (`bibtex-current-entry`) positions the cursor to the first column of current entry's name field. Two other commands having to do with entry scrolling are **C-c p** (`bibtex-previous-entry`) and **C-c c** (`bibtex-next-entry`) which move the cursor to the previous and the next entry, respectively. With prefix argument N, the two commands move the cursor to the N^{th} previous or next entry. Thus **C-u 2 C-c n** goes to the second next entry. Furthermore, **C-u 0 C-c n** is the same as **C-c c** and **C-u -1 C-c n** is equivalent to **C-c p**. The command **C-c c** does not take any prefix argument.

Any white space between two entries are associated with the entry sitting above. Suppose E_1, E_2, and E_3 are consecutive entries and the cursor is located in a blank line between E_2 and E_3. The commands **C-c p**, **C-c c**, and **C-c n** refer, respectively, to E_1, E_2, and E_3.

3.4 Duplicating/Killing Entries

Entries can be duplicated. The command **C-c C-d c** (`bibtex-dup-current-entry`) inserts a duplicate of current entry right above next entry. Similarly **C-c C-d p** (`bibtex-dup-previous-entry`) puts out a copy of the previous entry above current entry and **C-c C-d n** (`bibtex-dup-next-entry`) does it for the next entry below current entry. Entries can also be killed. The command prefix **C-c C-k** followed by **c**, **p**, or **n** kills the current, previous, or next entry. As before, operations referring to previous and next entries take an optional prefix argument.

3.5 Renaming Entry Types

You can also switch the current entry from one type to another. This feature is useful when you have entered most fields in an entry of type A and discovered that type B is more appropriate. At this point, you can either invoke a new entry of type B, copy each field from the previous entry (see the next section on field operations), and kill the previous entry when done, or you can simply invoke the command **C-c C-r** (`bibtex-rename-current-entry`) which does all of that for you automatically. You will be prompted for the destination type. Remember an entry type must start with an `@` and command completion can be used at the prompt.

4 Field Operations

Fields in a typical BIBTEX entry fall into two basic categories: *required* and *optional*. In BIBTEX each entry type has a predefined set of required fields and another set for optional fields. By required, we mean unless those fields are present with legal text (non-empty), the entry will be considered erroneous. In contrast, optional fields may or may not appear in the entry and those with empty text are simply discarded. Required fields have two special cases: *exclusive OR* and *inclusive OR*. In the exclusive OR case, exactly one out of a set of two or more fields must appear in the entry. In the inclusive case, at least one must be present.

When an entry is invoked in *BIBTEX-mode*, the information with regard to the category each field belongs to is also displayed. The best example is perhaps the entry type `@INBOOK` which contains not only ordinary required and optional fields, but both variants (exclusive and inclusive OR's) as well. Figure 1 is an instance of this type. The lines giving field category information will be deleted when the cleanup operation is invoked. Also to be deleted by the cleanup operation are the unfilled optional fields and empty exclusive OR field(s), provided exactly one in the set has been entered. Refer to Section 6 below for more details about the cleanup operation.

4.1 Scrolling Fields

Commands for field scrolling are compatible with the entry operations discussed in Section 3.3. That is, **C-c C-c** (`bibtex-current-field`) moves the cursor to the position following current field's opening delimiter. Similarly, **C-c C-p** (`bibtex-previous-field`) moves the cursor to the previous field and **C-c C-n** (`bibtex-next-field`) to the next, both work across entry boundaries. As usual, an operation referring to a previous or next field takes an optional prefix argument.

There are many different patterns of fields in a BIBTEX database. Field scrolling in *BIBTEX-mode* recognizes them all, yielding a set of commands that works uniformly on every possible pattern. Normally, the content (text) of a field must be quoted by a pair of delimiters (either `{...}` or `"..."`). But the text may be abbreviated (see Section 5). Abbreviated fields must not be quoted, hence some fields in an entry may not have enclosing delimiters for their text. Furthermore, every field in an entry must be terminated by a comma, with the exception of the last one, which must not be terminated by anything. Combining these two scenarios together, we get four different patterns. Furthermore, an entry name is considered a field in scrolling, but its string pattern is actually different than any of the first four patterns — a fifth pattern. Moreover, *BIBTEX-mode* allows one to invoke abbreviations anywhere in the file which means an abbreviation (i.e. the left hand side of '`=`' in `@STRING`) and its text (the RHS) must both be considered as scrollable fields — two more instances. Complicated pattern matching like this underscores the slowness of field scrolling, especially when a system is heavily loaded.

```
@INBOOK{,
=============================== REQUIRED FIELDS ===============================
    -------------- Exclusive OR fields:  specify exactly one --------------
    AUTHOR = {},
    EDITOR = {},
    -------------- Inclusive OR fields:  specify one or both --------------
    CHAPTER = {},
    PAGES = {},
    ------------- Rest of required fields:  specify every one -------------
    TITLE = {},
    PUBLISHER = {},
    YEAR = {},
=============================== OPTIONAL FIELDS ===============================
    VOLUME = {},
    SERIES = {},
    ADDRESS = {},
    EDITION = {},
    MONTH = ,
    NOTE = {}
}
```

Figure 1. A skeleton instance of the entry type `@INBOOK`.

4.2 Copying Fields

The content of a field may be copied from a previous or next entry. The source field must have a label which is either identical or similar (see Section 4.3) to that of the current field. For instance, the command **C-c C-t p** (`bibtex-text-previous-entry`) inserts before point the content of the field having the same label from previous entry. Similarly **C-c C-t n** (`bibtex-text-next-entry`) gets text from the next entry. Again, a prefix argument can be specified for these commands.

4.3 Equivalent Fields

Some entries may have different field labels denoting similar things. For example, the `JOURNAL` field in an `@ARTICLE` entry probably would mean the same thing as the `BOOKTITLE` field in a `@CONFERENCE` entry. So when one is copying the `BOOKTITLE` field from an `@ARTICLE` entry, he probably would not mind getting text from its `JOURNAL` field, since there is no `BOOKTITLE` field available. *BIBTEX-mode* has an equivalence table of these similar field labels. While copying text from an source entry to the current field (destination), if the current label is not found in the source, the next alternative in the equivalence table is used as key in searching. If an alternative match is found, the user will be asked to confirm insertion (no questions asked if a direct match is found in the first place). The same mechanism is used by `bibtex-rename-current-entry` discussed in Section 3.5.

Figure 2 is the equivalence table of *BIBTEX-mode*'s field labels. The first column item is considered the primary key. In a copying operation, if the primary key is not found in the source entry, the alternatives will be tried in the order shown above.

4.4 Erasing Fields

Field delimiters can be erased using the command **C-c C-e d** (`bibtex-erase-delimiters`). This is useful when an abbreviation is entered as the content of a field, therefore the delimiters must be deleted. Field

AUTHOR	EDITOR	
BOOKTITLE	JOURNAL	
EDITOR	AUTHOR	
INSTITUTION	ORGANIZATION	SCHOOL
JOURNAL	TITLE	
ORGANIZATION	INSTITUTION	SCHOOL
SCHOOL	INSTITUTION	ORGANIZATION
TITLE	BOOKTITLE	

Figure 2. Field label equivalence table.

text can be erased by **C-c C-e t** (`bibtex-erase-text`). Lastly the entire field can be erased by **C-c C-e f** (`bibtex-erase-field`).

For example, given the following excerpt from an `@INBOOK` entry,

```
SERIES = {···},
ADDRESS = {AWA█},
EDITION = {···},
```

typing the three erasing commands at the █ position each produces a different result.

(1) **C-c C-e d** produces

```
SERIES = {···},
ADDRESS = AWA,
EDITION = {···},
```

(2) **C-c C-e t** produces

```
SERIES = {···},
ADDRESS = {},
EDITION = {···},
```

(3) Finally **C-c C-e f** yields

```
SERIES = {···},
EDITION = {···},
```

4.5 Line Wrapping

As mentioned in Section 2.3, if the Emacs minor mode `Auto Fill` is turned on in *BIBTEX-mode*, the typing can go beyond the column `fill-column` and an automatic line wrapping mechanism will be triggered when a space is hit. The wrapped line will be indented to the column where the text of the current field begins. For instance, with `Auto Fill` mode set, entering a text string longer than `fill-column` for the following `SCHOOL` field without typing any explicit **RET** or **LFD** will trigger automatic line wrapping. That is, when the space immediately following `at` is hit (the \ at the right margin marks the continuation of the current line, as is done in Emacs, and the word *Berkeley* is italicized to indicate that it is typed after line wrapping), the field

```
    SCHOOL = "Computer Science Division, EECS Dept., University of California\
at █Berkeley"}
```

will become

```
        SCHOOL = "Computer Science Division, EECS Dept., University of California
                  at █Berkeley"},
```

where `at` is lined up with the first column of the current field text.

A similar effect can be achieved manually by typing **LFD** (`bibtex-newline-indent`) at the end of a line. It opens a new line and indents to the beginning of current text field, as in

```
        SCHOOL = "Computer Science Division, University of California, Berkeley, LFD
                 █},
```

where █ is the current cursor position.

Typing **ESC LFD** (`bibtex-newline-indent-label`), on the other hand, opens a new line and indents it by the amount of `bibtex-field-indent` (cf. Section 3.2). In the example that follows, the **ESC LFD** typed at the end of a line opens a new line with the cursor lined up with the other field labels, i.e.

```
        SCHOOL = "Computer Science Division, University of California, Berkeley},ESC LFD
        █
```

Again, █ denotes the current cursor position, where a new field label may be entered.

By comparison, **RET** is not bound to anything special; it simply opens a new line but does no indentation at all, that is

```
        SCHOOL = "Computer Science Division, University of California at Berkeley}, RET
    █
```

where █ is physically sitting at the left margin (column 0).

5 Abbreviation Facility

BIBTEX has a simple abbreviation scheme that works as follows. An abbreviation is specified by `@STRING`, such as:

```
@STRING{UCBCS = "Computer Science Division, University of California at Berkeley"}
```

The abbreviation `UCBCS` can then be entered as the content of a field, as in:

```
    SCHOOL = UCBCS,
```

Note that abbreviations as field text must not be quoted. Also, abbreviations must be defined before used.

One would normally like to put all abbreviations in a central place and not repeat them in each individual `.bib` files. BIBTEX does not explicitly support the notion of externally defined abbreviation entries. A somewhat kludgy approach is to list the file which contains all abbreviations as the *first* argument of the `\bibliography{...}` command in a document. It is important that it be listed first because BIBTEX does not allow forward referencing in terms of abbreviations. Note that the reference to external files happens in the document where citations are made. The `.bib` databases themselves know nothing about where the abbreviations are, they just use them. In other words, the program `bibtex` gets the information from the `\bibliography` command specified in your `.tex` files, not from any `.bib` files.

In *BIBTEX-mode* abbreviations can be specified by typing `@string` at the `M-x` prompt. The command `@abbreviation` is an alias of `@string`. Once invoked using either command, you get

```
@STRING{█= ""}
```

right above the current field with the cursor sitting on the left hand side of the equal sign. Both sides of `@STRING` are considered scrollable fields, as mentioned earlier.

BIBTEX-mode makes a number of extensions to BIBTEX's simple abbreviation scheme. The rest of this section discusses each of them.

5.1 Default Abbreviated Fields

Notice that by default the field labeled `MONTH` in an entry does not have field delimiters (as shown in the `@MISC` entry of Section 3.1). This is because one would normally give an abbreviate the `MONTH` field by saying `Jan`, `Feb`, ..., etc. Abbreviated text must not be quoted, according to the rule of BIBTEX. The other field which is assumed to be always using abbreviations is `JOURNAL`. The variable `bibtex-abbrev-fields` is bound to the list `("JOURNAL" "MONTH")` initially. Suppose you always abbreviate the `ADDRESS` field in the various entry types, you can put the following line in your hook:

```
(setq bibtex-abbrev-fields (cons "ADDRESS" bibtex-abbrev-fields))
```

```
@CONFERENCE{reid:bib,
     AUTHOR = {Brian K. Reid and David Hanson},
     TITLE = {An Annotated Bibliography of Background Material on Text Manipulation},
C-@  BOOKTITLE = {Proc.  of {SIGPLAN/SIGOA} Symposium on Text Manipulation},
     YEAR = {1981},
     PAGES = {157--160},
     ORGANIZATION = {ACM},
     ADDRESS = {Portland, Oregan},
     MONTH = {June 8--10},
     NOTE = {Available as {\it SIGPLAN Notices\/} 16(6)
             or {\it SIGOA Newsletter\/} 2(1--2).}█
}
```

Figure 3. A filled instance of the entry type `@CONFERENCE`.

5.2 Creating Group Abbreviations

The notion of group abbreviations, *a la* Unix *refer*, is implemented in *BIBTEX-mode*. BibTeX itself supports abbreviations of single fields only. In *BIBTEX-mode*, chunks of fields may be collected to form a group abbreviation under the heading of `%GROUP`. Other than the heading, a group abbreviation looks just like a regular BibTeX entry. Facilities are provided to create this type of abbreviations as well as for making references to them. This extension takes advantage of the BibTeX feature that text not enclosed in `@STRING` or any of the entry types is simply ignored in bibliography processing.

There are two ways to create group abbreviations. The first method is to make one out of an existing entry using the command **C-c C-\ g** (`bibtex-make-group`). It copies the text between mark and point to the space between current and previous entries. The text will be enclosed under the `%GROUP` heading whose name is supplied interactively by the user.

The following example demonstrates the idea. Suppose I want to include a number of papers in the conference proceedings of the 1981 ACM SIGPLAN/SIGOA Symposium on Text Manipulation in a `.bib` file. Except for the `AUTHOR`, `TITLE`, and `PAGES` fields, all the rest will be identical for every paper in that proceedings. I would invoke a new entry by typing `M-x@inp`**RET** and then enter the text for each field (see Figure 3.)

In addition to entering the text, I have also made a region between the beginning of `BOOKTITLE` and the end of `NOTE` (i.e. everything between **C-@** and █). Now if I type **C-c C-\ g** at the █ position and answer `TM81` to its prompt, I will get what is shown in Figure 4 in the space right above the `reid:bib` entry. This so-called group abbreviation will be registered and future references to it will have its content inserted. The `PAGES` field will remain to be that particular range, so I need to scroll to that field and replace its content. The schemes for interpolating group abbreviations will be discussed later.

The second method is similar to the invocation of regular BibTeX entries. Typing `%group` or `@group` at the `M-x` prompt invokes a group abbreviation skeleton. In this case the user has to specify a type upon which the group abbreviation is based. The type can be any of the entry types BibTeX knows.

For instance, after typing `%gr`**RET** at `M-x`, the following prompt will appear in the mini-buffer:

```
Group abbrev of type: █
```

and answering `@misc`**RET** (again, command completion can be used here) will have a skeleton group abbreviation inserted as Figure 5. A group abbreviation is regarded as a scrollable entry in a `.bib` file. Notice that except for `%GROUP`, all field labels in this skeleton are exactly those of the entry type `@MISC` (cf. Section 3). It is unlikely that fields such as `AUTHOR` and `TITLE` will be part of a group abbreviation. *BIBTEX-mode* isn't clever enough to exclude them. The user is responsible for deleting them using the command **C-c C-e f** (cf. Section 4.4). Empty fields can be deleted manually, or by the cleanup command **C-c ESC e** (cf. Section 6).

```
%GROUP{TM81,
     BOOKTITLE = {Proc.  of {SIGPLAN/SIGOA} Symposium on Text Manipulation},
     YEAR = {1981},
     PAGES = {157--160},
     ORGANIZATION = {ACM},
     ADDRESS = {Portland, Oregan},
     MONTH = {June 8--10},
     NOTE = {Available as {\it SIGPLAN Notices\/} 16(6)
             or {\it SIGOA Newsletter\/} 2(1--2).}
}
```

Figure 4. A group abbreviation made out of Figure 3.

```
%GROUP{,
=============================== OPTIONAL FIELDS ===============================
     AUTHOR = {},
     TITLE = {},
     HOWPUBLISHED = {},
     MONTH = ,
     YEAR = {},
     NOTE = {}
}
```

Figure 5. A skeleton group abbreviation based on the entry type `@MISC`.

5.3 Compiling and Loading

BIBTEX-mode has a "compiling" facility that pulls all abbreviations of both types to the beginning of the file and converts them to Lisp objects. These objects are entered into the Emacs database so that they can be browsed efficiently. At the same time they are also saved in a file so that loading them in the future is faster. The command **C-c C-s** (`bibtex-save-abbrev`) issued in the buffer bound to file `foo.bib` compiles all abbreviations and saves the converted objects in file `foo.abv`. What was originally in that file gets overwritten.

For each `.bib` file visited in *BIBTEX-mode* there is a database which keeps track of the abbreviation entries currently active. Group abbreviations created by **C-c C-\ g** (`bibtex-make-gabbrev`) are entered into the database automatically. Other abbreviations created by invoking `%group` or `@string` at the `M-x` prompt must be compiled explicitly using **C-c C-s** (`bibtex-save-abbrev`), which compiles each abbreviation entry in the buffer. Since abbreviations under *BIBTEX-mode* tend to be in a central file and the compiling command is not frequently invoked anyhow, this overhead seems tolerable.

In the database the name of a group abbreviation or the LHS of a field abbreviation is considered the key. The same key may be used in both categories with no problems. However, if the same key is used more than once in the same type, the new overwrites the old. *BIBTEX-mode* does not complain about multiply-defined entries, so it is the user's responsibility to assure a naming uniqueness.

The command **C-c C-l** (`bibtex-load-abbrev`), on the other hand, loads a `.abv` file into the database. When invoked, the load command prompts the user with:

```
Load abbreviations, file name base: █
```

Answering **RET** refers to the default case, that is, the base of the current file name concatenated with `.abv`. In most cases, loading is implied by the *browse* and *insert* commands, as described below.

5.4 Browsing/Inserting Group Abbreviations

The command **C-c C-ig** (`bibtex-insert-gabbrev`) can be used to browse or insert a group abbreviation. It starts with the prompt:

```
    Group abbrev name (default browsing mode): █
```

If the name of the group abbreviation is known, its text is inserted before point. Otherwise, answering **RET** enters *browsing mode* in which the current active group abbreviations are browsed one by one in the minibuffer for confirmation. If there are no abbreviations active, files in `bibtex-abbrev-files` as well as `foo.abv` are loaded, where `foo.bib` is the current file. A typical confirmation prompt looks like

```
    Confirm "FOO" (RET/y, SPC, DEL, s, g, f, C-r, ?=help)?
```

where `FOO` is the current key. The menu has following meaning:

- **RET**/'`y`' — Confirm and exit, the abbreviated text is inserted.
- **SPC** — Ignore current entry. Advance to the next.
- **DEL** — Ignore current entry. Go back to the previous.
- '`s`' — Show the content of current abbreviation in the other window.
- '`g`' — Go to the abbreviation whose name is greater than the specified key.
- '`f`' — Not in current list of abbreviations. Read more from a new file.
- **C-r** — Enter recursive edit. Return to browsing by **ESC C-c**.
- '`?`' – This help message.

If the desired entry is not found in the list being browsed, skipping the last entry triggers:

```
    Can't find next abbreviation, exit?  (y or n)
```

Answering '`n`' or **DEL** wraps around and starts the list all over again. Answering '`y`' or **SPC**, on the other hand, gets

```
    Group abbreviations not found, read from another file?  (y or n)
```

This prompt also appears as a result of **C-c C-ig** if no group abbreviations are loaded from the default files. A negative answer aborts the operation, whereas answering '`y`' or **SPC** at this point triggers

```
    Load file (default path TEXBIB): █
```

Entering a new `.abv` file name at this prompt starts another loading process and any group abbreviations found will become active.

Notice that the first time browsing mode is invoked, *BIBTEX-mode* attempts to load global abbreviation files from the list `bibtex-abbrev-files`. This allows you to put all abbreviations in a `.bib` file which contains nothing else but abbreviations. You can have the following code in your `bibtex-mode-hook`:

```
    (setq bibtex-abbrev-files '(abbrev))
```

which tells *BIBTEX-mode* to load abbreviations from `abbrev.abv` the first time browsing is invoked. Furthermore, a hook to *TEX-mode*'s bibliography preprocessor is available so that files in `bibtex-abbrev-files` will be inserted at the front of the `\bibliography{...}` command.

5.5 Browsing/Inserting Field Abbreviations

The command **C-c C-if** (`bibtex-insert-fabbrev`) enters browsing mode automatically. Once invoked, field abbreviations will be loaded from the files in `bibtex-abbrev-files` and from `foo.abv`, where `foo.bib` is the current file visited. When loaded, the abbreviation keys will be displayed in the minibuffer one by one in a lexicographical order. The options the user has are identical to those available in browsing group abbreviations, plus the following:

- **ESC** — Confirm and exit. Instead of inserting the text corresponding to the current key as in the case of **RET**, the key itself is inserted. The pair of field delimiters, if any, will be erased automatically.

For example, browsing field abbreviations at

```
    SCHOOL = {█},
```

and confirming `UCBCS` using **ESC** produces

```
    SCHOOL = UCBCS,
```

whereas answering **RET** or '`y`' produces

```
    SCHOOL = {Computer Science Division, University of California, Berkeley},
```

By comparison, group abbreviations are always expanded because BIBTEX knows nothing about it.

Again, if none of the field abbreviations currently available in the database is selected, the user has the option of loading another abbreviation file and start the browsing again.

6 Cleanup Mechanism

The cleanup operation in *BIBTEX-mode* performs the following tasks in sequence:

1. correcting empty fields,
2. deleting banners,
3. erasing trailing comma,
4. pulling abbreviations to the beginning of buffer,
5. time stamping.

Step 1 corrects or deletes empty fields. Empty optional fields will be deleted, so will empty exclusive OR fields, provided exactly one of them has been filled. Empty required fields will be caught, and the user will be asked to fill in the content interactively. For instance, let us assume Figure 1 is being cleaned up. With every field empty, the first prompt we get is

```
AUTHOR -- first XOR field to be filled: (RET if none) █
```

If a non-null string is given at the prompt, it will be inserted as the text of the `AUTHOR` field and the other exclusive OR field, `EDITOR`, will be deleted automatically. If, however, what we really want is the `EDITOR` field, answering **RET** will trigger a reconfirmation prompt:

```
Killing current field, are you sure? (y or n)
```

If the answer is positive, the empty `AUTHOR` field will be deleted and the next prompt

```
EDITOR -- the only XOR field remaining: █
```

will appear in the minibuffer. This time it won't let you get away unless a non-empty string is given because of the exclusive OR situation.

A similar situation applies to the inclusive OR case except that if the text for the first field is entered at prompt, the second will not be deleted automatically. The rest of the required fields (ordinary required fields), if empty, will all be prompted and you will not get through unless some non-empty text is given. Optional fields, on the other hand, are simply discarded. Every field in a group abbreviation is considered optional, therefore all empty fields get deleted regardless of their labels.

Each iteration of this mechanism searches only for an empty field and determines whether it's an exclusive OR, inclusive OR, oridnary required, or optional field, and finally what to do with it. If an empty field is found in an unknown entry type, the user is warned by

```
@FOO -- unknown entry type, kill it? (y or n)
```

where `@FOO` is the entry type in question. It is really harmless to leave unknown entries in the file because BIBTEX will pay no attention to them. But since this detection comes for free from the searching, and sometimes the unknown may be the result of typing errors, it is implemented anyhow. The premise here is the empty field. Anomalies such as more than one exclusive OR field are filled or some required fields are missing, which happen in an entry with no empty fields will not be detected. Errors like these can only be caught by BIBTEX.

The next step deletes banners which give hints on the category their following fields belong to. Then comes step 3 which erases the trailing comma in each entry. BIBTEX requires that fields in an entry be separated by a comma, but the last field in an entry must not. Since there is no way of telling which of the optional fields will be the last (because some of them may be left empty and will thus be deleted), *BIBTEX-mode* simply terminates each field with a comma in the skeleton entry and erases the trailing one after all empty fields are gone.

Step 4 involves pulling all abbreviations to the top of the file, so that they are physically separated from the main entries. The operation completes by time stamping the file at the beginning of file.

In *BIBTEX-mode*, one can either clean up the entire buffer using **C-c ESC b** (`bibtex-cleanup-buffer`), or a region in buffer using **C-c ESC r** (`bibtex-cleanup-region`), or consecutive entries using **C-c ESC e** (`bibtex-cleanup-entry`). In the region case, the starting bound is extended to the beginning of an entry implicitly, if it is enclosed in one. Similarly the ending bound is extended to the end of an entry, if it is inside an entry. In the entry case, an optional prefix argument is allowed to clean up neighborhood entries altogether. Thus **C-u 3 C-c ESC e** cleans up current and previous two entries, and **C-u -3 C-c ESC e** does it for the next three entires, excluding the current entry.

7 Draft Making and Debugging

The command **C-c C-\ d** (`bibtex-make-draft`) invokes a facility which prepares a draft bibliography file of all the entries listed in the current `.bib` file. A well-formatted draft can be previewed on a display using a **DVI** previewer or be sent to a printer for hard copies. This process also helps to debug the current `.bib` file for if there are errors, BibTEX will catch them before they are actually referenced in `.tex` files. *BIBTEX-mode* supports an error positioning mechanism which points the author to the spot in file where the next error occurs.

The BibTEX processor `bibtex` expects a `.aux` file as its argument. It processes `\citation{...}` entries in the `.aux` file and generates a `.bbl` file, the actual bibliography. In the original design, LATEX is supposed to pick up `\cite` and `\nocite` entires from source files and turn them into `\citation` entries in the `.aux` file. In *TEX-mode* [1], the `.aux` file is produced by searching through source files directly without invoking the formatter. *BIBTEX-mode* uses the same technique in draft making. That is, it lists the name of every entry in the current file `foo.bib` as the argument of `\citation` in file `foo+.aux`. If `foo.bib` has been modified when **C-c C-\ d** is issued, the user will be asked to confirm cleanup (see Section 6) first. The reason for the '`+`' attachment is to avoid clobbering `foo.aux` which is likely to be there as a result of processing file `foo.tex` by LATEX.

Before `foo+.aux` can be processed by `bibtex`, one more piece of information is needed, namely the style of the bibliography. The menu

```
Style (RET=plain, 1=unsrt, 2=alpha, 3=abbrv, else=your-own-style): █
```

gives the user options for the desired style. The first four options correspond to those available in BibTEX. The default style `plain` is an order based on the last name of the first author using numeric values as actual references (e.g. [1], [2]); `unsrt` is also numeric, but the sequence is based on the order the entries appear in file; the `alpha` style is very much the same as `plain` except the actual references are alphanumeric in terms of author name and year of publication (e.g. [Kn84]). Finally the `abbrv` style uses a compact form of name, month, etc. in the actual bibliography file. Its reference field is numeric. If you have any user-defined style, simply type in the name (assuming BibTEX knows about its existence). The selected style will be used as the argument of the `\bibstyle` command in `foo+.aux`.

Another interesting thing that happens behind the scenes in preparing the `.aux` file for BibTEX is the processing of global abbreviation files bound to the variable `bibtex-abbrev-files` (cf. Section 5.4). Suppose in your `bibtex-mode-hook` you have

```
(setq bibtex-abbrev-files '(abbrev))
```

which means all your abbreviations are in file `abbrev.bib`. In `foo+.aux` there is one more important command,

```
\bibdata{abbrev,foo}
```

which informs BIBTEX to process the global abbreviation file before processing the `.bib` file in question. This important hook is also used by *TEX-mode* in preparing actual bibliographies.

When this `foo+.aux` is done, the program `bibtex` is invoked with argument `foo+` in the inferior shell process. During the execution of `bibtex`, the prompt

```
Continue formatting the draft?  [Wait till finish if 'y']
```

remains in the minibuffer until a command '`y`' or '`n`' (**SPC** or **DEL**) is given. As the message warns: you must wait till the job is finished if your answer is '`y`'. You can type '`n`' any time to abort the BIBTEX job, however. If there are no errors found when an abort is demanded, you will get the following message:

```
Making a draft bibliography...abort (temporary files deleted)
```

which means the file `foo+.aux` is gone. If there are errors detected and an abort is issued before BIBTEX completes, you get

```
Making a draft bibliography...abort (use C-c C-@ to locate BibTeX errors)
```

The command **C-c C-@** positions you to the next error in the current `.bib` file. Meanwhile, the shell window will recenter itself to show you the corresponding error message as much as possible.

If you decide to continue even if there are BIBTEX errors, you will be asked to reconfirm:

```
BibTeX errors found, are you sure you want to continue? (y or n)
```

If '`n`' or **DEL** is given this time, it terminates with the second abort message shown above.

If BIBTEX finishes successfully with no errors and the user decides to continue, or if there are errors but the user wants to go on anyway, the message will be

```
Making a draft bibliography...continuing
```

What happens next is the preparation of the actual bibliography file `foo+.tex`. A `.bib` entry like Figure 3 will be converted to

```
\bibitem{79}{reid:bib}{
Brian~K. Reid and David Hanson.
\newblock An annotated bibliography of background material on text manipulation.
\newblock In {\it Proc.  of the ACM SIGPLAN/SIGOA Symposium on Text
  Manipulation}, pages~157--160, ACM, Portland, Oregan, June 8--10 1981.
\newblock Available as {\it SIGPLAN Notices\/} 16(6)
          or {\it SIGOA Newsletter\/} 2(1--2).}
```

in `foo+.tex`. The macro `\newblock` along with other definitions having to do with the format of this draft are given in the preamble of this TEX file (see Figure 6).

In addition to defining formatting details, the preamble has in its first line an input command which loads the file `misc.tex`. This is another hook *BIBTEX-mode* requires to be able to properly format the draft for previewing or printing. In this case, I have the following statement in my `bibtex-mode-hook`:

```
(setq bibtex-context "misc")
```

which implies that I am using something in the `.bib` file whose definitions reside in `misc.tex`. A good example of this is the BIBTEX logo, which the `plain` package knows nothing about. Therefore I'm including the file `misc.tex` which presumably has the definition for the macro `\BibTeX`.

```
\input misc
\def\newblock{\hskip .11em plus .33em minus -.07em}
\def\bibitem#1#2#3{
  {\bigskip  \advance\leftskip by 1in
   \item{\hbox to 1.25in{\hss$\lbrace#2\rbrace$}
   \quad\hbox to .6in{\hss[#1]}}
   #3\par}}
\font\big=cmbx10 scaled\magstep3
\nopagenumbers
\footline={{\bf Time:  }{\sl Fri Sep  5 00:51:50 1986}\hfil
           {\bf File:  }``{\tt ~/bib/foo.bib}''\hfil
           {\bf Page:  } \folio}
\centerline{\big Bibliography}
\vskip .15truein
```

Figure 6: Draft bibliography preamble.

When this `foo+.tex` file is done, `tex` takes over and formats it. Similar to the previous `bibtex` job, the prompt

```
Preview, print, or save the draft?  [Wait till finish if 'y']
```

will be displayed in the minibuffer with the same protocol. If you decide to do either of the three things listed, you must wait till the job completes before hitting **SPC** or '`y`'. If, however, you don't want to proceed, typing **DEL** or '`n`' at any moment will abort the job, with all temporary files deleted.

When the formatting is finished, the **DVI** file is then ready for previewing or printing. If previewing is desired (first prompt), the program bound to `bibtex-softcopy` is invoked. If printing is needed (second prompt), the program `bibtex-hardcopy` is used to generate a hard copy in the designated printer. Finally, if you wish to keep these temporary files (after all, they are just for proof reading the draft), there will be a third prompt asking for your confirmation on saving `foo+.tex` and `foo+.dvi`. If not, all temporary files will be removed. Files such as the `.aux`, `.log`, `.blg` get deleted in any case.

8 Final Remarks

In summary, *BIBTEX-mode* commands and their key bindings obey the following convention:

Command	*Meaning*
C-c [*cpn*]	entry scrolling
C-c C-k [*cpn*]	entry killing
C-c C-d [*cpn*]	entry duplicating
C-c C-[*cpn*]	field scrolling
C-c C-e [*dtf*]	field erasing
C-c C-e [*pn*]	field copying
C-c C-i [*gf*]	abbreviation insertion
C-c ESC [*bre*]	cleanup
`M-x` *name*	entry invocation

While in *BIBTEX-mode*, the command **C-c C-h** (`bibtex-mode-help`) displays in the other window a table of the commands available. A version of this document will be translated to TEXInfo [7], the official GNU Emacs documentation format. When that is available you will be able to run the `info` system in Emacs to consult this manual interactively.

Acknowledgements. I would like to thank the following people for making constructive suggestions on *BIBTEX-mode* at various stages of the development: Mike Harrison, Art Werschulz, Ben Zorn, and Jim Larus.

9 References

[1] Peehong Chen. *GNU Emacs TEX Mode.* Technical Report, Computer Science Division, University of California, Berkeley, California, 1986.

[2] Peehong Chen, Michael A. Harrison, John Coker, Jeffrey W. McCarrell and Steve Procter, "An improved user environment for TEX", in *Proc. of the 2nd European Conference on TEX for Scientific Documentation*, Strasbourg, France, June 19–21, 1986. To be published by Springer-Verlag.

[3] Donald E. Knuth. *The TEX Book.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

[4] Leslie Lamport. *LaTEX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[5] Oren Patashnik. *BIBTEXing.* Computer Science Department, Stanford University, Stanford, California, March 1985.

[6] Richard M. Stallman. *GNU Emacs Manual*, 4th Edition, Version 17, Free Software Foundation, Cambridge, Massachusetts, February 1986.

[7] Richard M. Stallman. *TEXInfo, The GNU Documentation Format.* first edition, Free Software Foundation, Cambridge, Massachusetts, June 1985.

[8] Michael D. Spivak. *The Joy of TEX*, American Mathematical Society, Providence, RI., 1985.

10 Summary

10.1 Installation and Startup (cf. Chapters 1, 2, and 8)

`bibtex-mode.el` **file**

A *BIBTEX-mode* file which defines basic attributes and key bindings for *BIBTEX-mode*. Also included is a collection of supporting functions shared by other subsystems. To begin with only this file is loaded.

`tex-misc.el` **file**

A *BIBTEX-mode* file which defines the cleanup mechanism as well as the draft making facility and the debugging aid. This file is autoloaded whenever a function defined in it is invoked.

`tex-ops.el` **file**

A *BIBTEX-mode* file which defines entry and field operations. This file is autoloaded whenever a function defined in it is invoked.

`tex-abv.el` **file**

A *BIBTEX-mode* file which includes the extended abbreviation facility. This file is autoloaded whenever a function defined in it is invoked.

`tex-init.el` **file**

A file which may be created locally to redefine site-specific attributes. This file is loaded whenever the function `bibtex-mode` is invoked.

`bibtex-mode` **major mode function**

Major mode for editing BIBTEX database files.

`bibtex-mode-version` **function**

Return the current *BIBTEX-mode* version.

`bibtex-mode-help` **C-c C-h**

Display a summary of *BIBTEX-mode* commands in the other window.

`bibtex-mode-hook` **variable**

Variable to be bound to `(function (lambda () <body>))` where `<body>` is a sequence of statements having to do with abbreviations, redefinition of key bindings, non-default settings of *BIBTEX-mode* variables, loading of other functions, etc.

`abbrev-mode` **minor mode function**

An Emacs minor mode which enables the expansion of abbreviated text. By default this mode is turned off in *BIBTEX-mode*. Invoking this function with a positive integer turns the mode on.

`bibtex-abbrev-enable` **C-c C-a SPC**

Unconditionally enables the `Abbrev` minor mode.

`bibtex-abbrev-disable` **C-c C-a DEL**

Unconditionally disables the `Abbrev` minor mode.

`auto-fill-mode` **minor mode function**

An Emacs minor mode which enables auto line wrapping when a space is typed beyond column `fill-column`. In *BIBTEX-mode* `fill-column` is set to 78 but this mode is turned off by default. Invoking this function with a positive integer turns the mode on.

`bibtex-autofill-enable` **C-c LFD SPC**

Unconditionally enables the `Auto Fill` minor mode.

`bibtex-autofill-disable` **C-c LFD DEL**

Unconditionally disables the `Auto Fill` minor mode.

10.2 Entry Operations (cf. Chapter 3)

`@article` **function**

Invoke a new entry skeleton of type `@ARTICLE`, an article from a journal or magazine. Command completion can be used at the `M-x` prompt.

`@book` **function**

Invoke a new entry skeleton of type `@BOOK`, a book with explicit publisher. Command completion can be used at the `M-x` prompt.

`@booklet` **function**

Invoke a new entry skeleton of type `@BOOKLET`, a work that is printed and bound, but without a named publisher or sponsoring institution. Command completion can be used at the `M-x` prompt.

`@conference` **function**

Invoke a new entry skeleton of type `@CONFERENCE`, the same as type `@INPROCEEDINGS`. The entry type `@CONFERENCE` is included for compatibility with *Scribe*. Command completion can be used at the `M-x` prompt.

`@inbook` **function**

Invoke a new entry skeleton of type `@INBOOK`, a part of a book, which may be a chapter and/or a range of pages. Command completion can be used at the `M-x` prompt.

`@incollection` **function**

Invoke a new entry skeleton of type `@INCOLLECTION`, a part of a book having its own title. Command completion can be used at the `M-x` prompt.

`@inproceedings` **function**

Invoke a new entry skeleton of type `@INPROCEEDINGS`, an article in a conference proceedings. Command completion can be used at the `M-x` prompt.

`@manual` **function**

Invoke a new entry skeleton of type `@MANUAL`, a technical documentation. Command completion can be used at the `M-x` prompt.

`@masterthesis` **function**

Invoke a new entry skeleton of type `@MASTERTHESIS`, a master's thesis. Command completion can be used at the `M-x` prompt.

`@misc` **function**

Invoke a new entry skeleton of type `@MISC`, a wild card. Use this type when nothing else fits. Command completion can be used at the `M-x` prompt.

`@phdthesis` **function**

Invoke a new entry skeleton of type `@PHDTHESIS`, a Ph.D. thesis. Command completion can be used at the `M-x` prompt.

`@proceedings` **function**

Invoke a new entry skeleton of type `@PROCEEDINGS`, the proceedings of a conference. Command completion can be used at the `M-x` prompt.

`@techreport` **function**

Invoke a new entry skeleton of type `@TECHREPORT`, a report published by a school or other institution. Command completion can be used at the `M-x` prompt.

`@unpublished` **function**

Invoke a new entry skeleton of type `@UNPUBLISHED`, a document having an author and title, but not formally published. Command completion can be used at the `M-x` prompt.

`bibtex-current-entry` **C-c c**

Position to the name field of current entry.

`bibtex-previous-entry` **C-c p**

Go back to the name field of previous entry. With prefix argument N, go back to the N^{th} previous entry. Report error if the target entry is not found.

`bibtex-next-entry` **C-c n**

Advance to the name field of next entry. With prefix argument N, advance to the N^{th} next entry. Report error if the target entry is not found.

`bibtex-dup-current-entry` **C-c C-d c**

Make a duplicate of the current entry above. Position the cursor at the name field of the new entry.

`bibtex-dup-previous-entry` **C-c C-d p**

Make a duplicate of the previous entry above the current one. Position the cursor at the name field of the new entry. With prefix argument N, duplicate the N^{th} previous entry. Report error if the target entry is not found.

`bibtex-dup-next-entry` **C-c C-d n**

Make a duplicate of the next entry below the current one. Position the cursor at the name field of the new entry. With prefix argument N, duplicate the N^{th} next entry. Report error if the target entry is not found.

`bibtex-kill-current-entry` **C-c C-k c**

Kill the current entry. The cursor is positioned at the name field of next immediate entry, if any. If current entry is the last one in file, position the cursor at the previous entry.

`bibtex-kill-previous-entry` **C-c C-k p**

Kill the previous entry. Cursor remains at the same place. With prefix argument N, kill the N^{th} previous entry. Report error if the target entry is not found.

`bibtex-kill-next-entry` **C-c C-k n**

Kill the next entry. Cursor remains at the same place. With prefix argument N, kill the N^{th} next entry. Report error if the target entry is not found.

`bibtex-rename-current-entry` **C-c C-r**

Switch the type of current entry to a target type specified at prompt (with command completion). Fields with same labels are copied, and those belonging to the same equivalence group will be copied upon receiving confirmation from the user. Fields whose labels are not in the target type or their equivalence set are discarded.

10.3 Field Operations (cf. Chapter 4)

`bibtex-current-field` **C-c C-c**

Go to the beginning of current field.

`bibtex-previous-field` **C-c C-p**

Go to the beginning of the previous field. With prefix argument N, go to the N^{th} previous field. Go across entry boundaries if necessary. Report error if the target field is not found.

`bibtex-next-field` **C-c C-n**

Go to the beginning of the next field. With prefix argument N, go to the N^{th} next field. Go across entry boundaries if necessary. Report error if the target field is not found.

`bibtex-text-previous-entry` **C-c C-c p**

Copy text from previous entry having the same label as the current field. Insert the text before point. If the same label is not found, try an equivalent one, if any, and prompt for confirmation. With prefix argument N, look up in the the N^{th} previous entry. Report error if the target field or the target entry is not found.

`bibtex-text-next-entry` **C-c C-c n**

Copy text from next entry having the same label as the current field. Insert the text before point. If identical label is not found, try an equivalent one, if any, and prompt for confirmation. With prefix argument N, look up in the the N^{th} next entry. Report error if the target field or the target entry is not found.

`bibtex-erase-delimiters` **C-c C-e d**

Erase the current field delimiter. This is most useful when a field abbreviation is inserted as text. Since abbreviations must not be quoted, this command can be used to eliminate them.

`bibtex-erase-text` **C-c C-e t**

Erase the text in the current field. Field label and field delimiter remain intact.

`bibtex-erase-field` **C-c C-e f**

Erase the current field completely.

`bibtex-newline-indent` **LFD**

open a newline indent to the beginning of the current field text.

`bibtex-newline-indent-label` **ESC LFD**

open a newline indent to the beginning of the current field label.

10.4 Abbreviation Facility (cf. Chapter 5)

`@abbreviation` **function**

An alias of `@string`.

`@string` **function**

Invoke a field abbreviation skeleton. Command completion can be used at the `M-x` prompt.

`@group` **function**

An alias of `%group`.

`%group` **function**

Invoke a group abbreviation skeleton of the type specified at prompt, which must be one of the 14 entry types defined above. Unfilled fields can be cleaned up manually or by **C-c ESC e**. Unwanted entries, if filled, can be eliminated by **C-c C-e f**. Command completion can be used at both `M-x` and the second prompts.

`bibtex-make-gabbrev` **C-c C-\ g**

Make a group abbreviation out of the current region with the name given at prompt. The new group abbreviation is inserted above the current entry. The `[group, name, abbrev]` tuple is entered into a per buffer database at the same time. Unwanted fields must be manually erased, and then recompiled.

`bibtex-save-abbrev` **C-c C-s**

Compile both field and group abbreviations in current buffer (file) `foo.bib` into Lisp objects, enter them in Emacs database, and save these objects in file `foo.abv` for efficient loading.

`bibtex-load-abbrev` **C-c C-l**

Load all abbreviation files listed in `bibtex-abbrev-files` and then the current file. For each abbreviation file involved, if a compiled `.abv` file is found, it is loaded directly; else the file is first compiled (and therefore loaded), and a new `.abv` file is created. When loaded, all field and group abbreviations defined in those files are entered in a per buffer database.

`bibtex-insert-gabbrev` **C-c C-i g**

Insert the content of a group abbreviation. All group abbreviations currently available in database, plus those in files list in `bibtex-abbrev-files`, if not already loaded, can be browsed. If none of them is selected, the user has the option to load another abbreviation file.

`bibtex-insert-fabbrev` **C-c C-i f**

Insert the content or the name of a field abbreviation. Enter browsing mode automatically. All field abbreviations currently available in database, plus those in files list in `bibtex-abbrev-files`, if not already loaded, can be browsed. If none of them is selected, the user has the option to load another abbreviation file.

10.5 Cleanup Mechanism (cf. Chapter 6)

`bibtex-cleanup-buffer` **C-c ESC b**

Clean up the current buffer, including (1) deleting and correcting empty fields, (2) deleting banners, (3) deleting trailing commas, (4) pulling all abbreviations to the top of file, and (4) time stamping the file for the changes.

`bibtex-cleanup-region` **C-c ESC r**

Clean up the current region. The work involved is the same as in the buffer case. If the region boundary is in the middle of an entry, it gets extended to the beginning or end of that region implicitly.

`bibtex-cleanup-entry` **C-c ESC e**

Clean up the current entry. The work involved is the same as in the buffer case. With positive prefix argument N, clean up the current and the previous $N-1$ entries. If N is negative, clean up the next N entries.

10.6 Draft Making (cf. Chapter 7)

`bibtex-make-draft` **C-c C-\ d**

Make a draft bibliography for previewing or printing. The final bibliography source will be in `foo+.tex` where `foo.bib` is the current file.

`bibtex-context` **variable**

A list of files with macro definitions used in the current `.bib` file. These files will be loaded automatically at the beginning of `foo+.tex` so that the formatting of draft can be done smoothly.

`bibtex-softcopy` **variable**

Name of the DVI previewer (default "`/usr/local/dvitool -E`").

`bibtex-hardcopy` **variable**

Name of the DVI printing and spooling scheme (default "`lpr -d -Pxp`", where "`xp`" is the printer name given by the user).

`bibtex-printer-list` **variable**

List of available printers (default "`(ip, cx, dp, gp)`").

`bibtex-printer-default` **variable**

Name of the default printer (default "`gp`").

11 Index to Function Names and Variables

12 Index to Key Bindings

— Installation and Startup —

(See Chapters **1**, **2**, **8** and Section **10.1**)

file	`bibtex-mode.el`
file	`bibtex-misc.el`
file	`bibtex-ops.el`
file	`bibtex-abv.el`
file	`bibtex-init.el`
major mode function	`bibtex-mode`
function	`bibtex-mode-version`
C-c C-h	`bibtex-mode-help`
variable	`bibtex-mode-hook`
minor mode function	`abbrev-mode`
C-c C-a SPC	`tex-abbrev-enable`
C-c C-a DEL	`tex-abbrev-disable`
minor mode function	`auto-fill-mode`
C-c LFD SPC	`tex-autofill-enable`
C-c LFD DEL	`tex-autofill-disable`

— Built-in Entry Types —

(See Chapters **3** and **10.2**)

M-x@article	`an article from a journal or magazine`
M-x@book	`a book with explicit publisher`
M-x@booklet	`a work printed without a named publisher`
M-x@conference	`same as M-x@inproceedings`
M-x@inbook	`a part of a book`
M-x@incollection	`a part of a book having its own title`
M-x@inproceedings	`an article in a conference proceedings`
M-x@manual	`technical documentation`

M-x@masterthesis	`a master's thesis`
M-x@misc	`wild card type, use when nothing else fits`
M-x@phdthesis	`a Ph.D. thesis`
M-x@proceedings	`the proceedings of a conference`
M-x@techreport	`a report published by a school/institution`
M-x@unpublished	`a document not formally published`

— Entry Operations —

(See Chapter **3** and Section **10.2**)

C-c c	`bibtex-current-entry`
C-c p	`bibtex-previous-entry`
C-c n	`bibtex-next-entry`
C-c C-d c	`bibtex-dup-current-entry`
C-c C-d p	`bibtex-dup-previous-entry`
C-c C-d n	`bibtex-dup-next-entry`
C-c C-k c	`bibtex-kill-current-entry`
C-c C-k p	`bibtex-kill-previous-entry`
C-c C-k n	`bibtex-kill-next-entry`
C-c C-r	`bibtex-rename-current-entry`

— Field Operations —

(See Chapters **4** and **10.3**)

C-c C-c	`bibtex-current-field`
C-c C-p	`bibtex-previous-field`
C-c C-n	`bibtex-next-field`
C-c C-c p	`bibtex-text-previous-entry`
C-c C-c n	`bibtex-text-next-entry`
C-c C-e d	`bibtex-erase-delimiters`
C-c C-e t	`bibtex-erase-text`
C-c C-e f	`bibtex-erase-field`
LFD	`bibtex-newline-indent`
ESC LFD	`bibtex-newline-indent-label`

— Abbreviation Facility —

(See Chapter **5** and Section **10.4**)

M-x@abbreviation	`same as M-x@string`
M-x@string	`field abbrev invocation`
M-x@group	`same as M-x%group`
M-x%group	`group abbrev invocation`
C-c C-\ g	`bibtex-make-gabbrev`
C-c C-s	`bibtex-save-abbrev`
C-c C-l	`bibtex-load-abbrev`
C-c C-i g	`bibtex-insert-gabbrev`
C-c C-i f	`bibtex-insert-fabbrev`

— Cleanup Mechanism —

(See Chapter **6** and Section **10.5**)

C-c ESC b	`bibtex-cleanup-buffer`
C-c ESC r	`bibtex-cleanup-region`
C-c ESC e	`bibtex-cleanup-entry`

— Draft Making —

(See Chapter **7** and Section **10.6**)

C-c C-\ d	`bibtex-make-draft`

— Useful Variables —

`fill-column`	**2.3, 10.1**
`bibtex-abbrev-fields`	**5.1, 10.4**
`bibtex-abbrev-files`	**5.4, 7, 10.4**
`bibtex-context`	**7, 10.4**
`bibtex-entry-use-parens`	**3.2, 10.2**
`bibtex-field-use-quotes`	**3.2, 10.2**
`bibtex-field-indent`	**3.2, 10.2**
`bibtex-printer-list`	**2, 10.6**
`bibtex-printer-default`	**2, 10.6**
`bibtex-hardcopy`	**2, 7, 10.6**
`bibtex-softcopy`	**2, 7, 10.6**