Using BibTeX

David Young
young@maths.anu.edu.au

May 15, 2002

Contents

1 Introduction

BibTeX is a companion program to LaTeX, and was written by Oren Patashnik. It was written to easily generate bibliographic reference lists for use with LaTeX in a flexible way.

This course will go through some of the basics of using BibTeX with LaTeX, and will also (towards the end) go through a few of the niftier things you can do with BibTeX and related packages. I'll give suggestions of things to muck around and do to explore BibTeX a little in boxes. Of course, feel free to try out anything you want!

(One of the things to remember about BibTeX is that there's no easy way to remember everything. What's best is to be familiar with a lot of things that you'll find useful, and know where to find information about them when you can't remember *everything*. This is why I do encourage you to play around with stuff so you know what's going on, particularly for when you're not getting exactly what you want!)

There are several very good sources of information about BibTeX. Appendix B of Leslie Lamport's book *LaTeX: a Document Preparation System* [Lam94] is devoted to BibTeX. For an updated and augmented version of this, take a look at Chapter 13 of Goossens et al. *The LaTeX Companion* [GMS94]. Other good sources of information about all things TeX, including LaTeX and BibTeX are the TeX Users' Group: `http://www.tug.org/` [TUG], and the Comprehensive TeX Archive Network: `http://www.ctan.org/` [CTAN].

It pays to know where your LaTeX distribution resides on your system, because there are a lot of interesting packages, styles and so on hidden there. On this system, the user files for the distribution are based at

```
/usr/local/teTeX/share/texmf/
```

but it may be different on different systems. If I refer to any files, you may have to change the initial part of the path so that it matches your local LaTeX distribution. One important file I will mention immediately is the file

```
/usr/local/teTeX/share/texmf/bibtex/bib/base/xampl.bib
```

which is a good example of the many types of entries in a `.bib` file. Feel free to use this when mucking around. Also the file

```
/usr/local/teTeX/share/texmf/doc/bibtex/base/btxdoc.dvi
```

gives a good introduction to BibTeX.

Some other files which you might find useful can be found at `http://wwwmaths.anu.edu.au/~young/bibtex` or in `/students/s3269901/bibtex` on the machine `bohm`: `notes.pdf` is this file in PDF format, `course.tex` is the `.tex` file used to produce those notes, and `refs.bib` is the bibliography database file that goes with it. Feel free to use these to play around with different things, or use your own files.

Part I
The Basics

2 Doing it manually

It's easy enough to create a bibliography yourself within a LaTeX document without using BibTeX, at least for small bibliographies.

The basic ingredients are as follows:

- a `\begin{thebibliography}` command with an argument which is a piece of text the same width as the widest item label in the source list. For example, if you have fewer than 100 entries and they are labelled with numbers, `\begin{thebibliography}{88}` should do fine. In the `article` document class the output reference list is labeled "References", while in the `book` and `report` classes it is labeled "Bibliography".
- a list of references sorted as you want them to appear. Each should begin with a `\bibitem` command, the argument of which is the citation key of that entry. In addition, an optional argument can be used to change the label as it appears in the output. For example, `\bibitem{COMPANION}` or `\bibitem[Lam94]{LaTeX}`. What follows should be what you want to appear in the bibliography, formatted as you want it to appear.
- an `\end{thebibliography}` command.

In your text you can include `\cite` commands whose argument is one (or more) of the citation keys. `\cite` also has an optional argument, which can be used, for example, to give a range of pages.

The argument of the `\bibitem` command is needed, even if you don't want to refer to that item — if you don't include it, LaTeX will take the first letter of that item as the citation key, and so that letter will be missing from your output!

Within your entries you can provide a number of `\newblock` commands. This breaks the information up into blocks which are formatted differently depending on whether the LaTeX compiles the document with the `openbib` option set, that is, using this as the optional argument to the `\documentclass`, for example

```
\documentclass[openbib]{article}.
```

If it is set, a `\newblock` will start a new line which is slightly indented, otherwise it has no effect.

If we use this method, on the first run-through LaTeX won't know what refers to what, but will make a note of it in the `.aux` file. Then, on the second run through it will use all the information gathered on the previous time through to create the citations.

Create a file with some text, and include a short bibliography using the above method, with `\begin` and `\end{thebibliography}` and `\bibitem`s, and `\cite` some of the entries (with and without the optional argument, and with multiple keys as well). Also use `\newblock`s, and see how the formatting changes when you use the `openbib` option for the document class.

3 Why we should use BibTeX

This is all well and good for small reference lists in a small number of files, but it starts getting harder to manage when we need more references or need to write more documents. Moreover, it is inherently inflexible – if we change our mind about how we want the information presented, such as what should be put in italics, or what data actually needs to be included, then we have to go through and make all the changes by hand.

BibTeX is designed to be a middle-man for the bibliography generation process, and do all the formatting each time every time, pulling the information it needs from a possibly larger list we give it. Using BibTeX we don't have to make any final formatting decisions about our bibliographic data when we enter it; this means that we never have to worry about converting a bibliography laid out in one style to a different style. Moreover, we don't have to worry whether we're actually using one consistent style, or slipping into several in the one file! What's more, rather than separately entering a reference list into each paper or book we write, with its own formatting, we can keep one (possibly large) list of all the references we ever use, adding to it when we need to, and creating a different bibliography in a different format for each document we need using this one list.

In summary we can:

- keep all our bibliographic data in one place (or more, if we feel like it) for use in whatever LaTeX files we need, if at all
- change the formatting style for each document with no more than changing a `\bibliographystyle` command
- be sure that our style is consistent within each document

4 The basic ingredients

Essentially using BibTeX involves taking all the information that might be needed for your references, putting them in a separate file, and then letting

BibTeX do the formatting for you. What it produces is actually in the `\bibitem` format we would use if we were doing it ourselves.

The basic process is as follows. In your `.tex` file you'll have included some references with `\cite` commands. (Remember that this command has an optional argument, and you can have more than one citation for a single `\cite` command. Also, if you want a reference included in the bibliography without being referred to use a `\nocite` command, or even `\nocite{*}`.) When you need to refer to a new work, you'll add its details fo a `.bib` file, in a format that BibTeX will understand. At the end of your document (or wherever you want the bibliography to appear) you'll put the command `\bibliographystyle` whose argument is the style you want, and the command `\bibliography`, whose argument is the file or files where your bibliographic data is kept. (The list of bibliographic data files needs to be separated by commas, with no whitespace between names.)

When you first run LaTeX on the file you'll get warning messages something like

```
LaTeX Warning: Citation 'COMPANION' on page 3 undefined on input line 62.
```

You'll also get the warnings about other undefined references, like those produced with `\ref` commands. Running LaTeX again will resolve these ones, but not the citation warnings – the unresolved citations will appear in your output as [?], and there will also be no bibliography.

At this stage LaTeX will have listed in the `.aux` file the citations it came across.

Run LaTeX on a new `.tex` file once and have a look inside the `.aux` file and see what LaTeX has written.

Now, when we run BibTeX on our file, we will actually be running it on our `.aux` file, rather than our `.tex` file. For the moment this is a moot point, as they both have the same stem. BibTeX looks in the `.aux` file for what references it needs to look out for, what file or files this information will be in, and for what style it needs to use. Then it grabs the data from that file and formats it how we've asked it. This stuff it will put in a `.bbl` file.

Run BibTeX on the `.aux` file LaTeX produced above, and look at the format of the `.bbl` file.

This file should have been formatted by BibTeX just like how we described in Section 2, "Doing it Manually".

When we run LaTeX next it will incorporate this data in the reference list at the end. It will also add to the `.aux` file the citation keys we have used, and the labels that they refer to. We will need to run LaTeX another time to get all the `\cite` commands resolved, rather than showing [?]. This is what the

```
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
```

warning tell us.

You guessed it, run LaTeX again and look at how the `.aux` file has changed (what has been added to it).

To produce a fully-referenced document from, say, `paper.tex`, you thus need to run four commands:

```
latex paper
bibtex paper
latex paper
latex paper
```

Each time you add or remove a citation, you'll need to go through the whole LaTeX-BibTeX-LaTeX-LaTeXprocess again to get everything right.

Occasionally, such as if you're submitting a paper electronically, you will need your document to be self-contained. In this case we can still use BibTeX, but at the last step we simply need to "copy-and-paste" the information from the `.bbl` file into our `.tex` file in place of the `\bibliographystyle` and `bibliography` commands.

In the end, each of the following files may have been used:

`.tex` the source file of the document

`.aux` the file LaTeX produces which contains all the referencing information, and which BibTeX reads to find out this information. It contains information on what format to use, and where to find the bibliographic data. It also contains a list of all the citations referred to (so that BibTeX knows what to format), and a list of what stands for what (so it knows what to write when it comes across a `\cite` command)

`.bib` the file or files that contain the bibliographic data – the list of papers, books, and so one

`.bbl` the formatted output produced by BibTeX, containing a `\begin{bibliography}` command, a whole list of `\bibitem`s with the data, and an `\end{bibliography}` command

`.blg` the BibTeX log file, containing any warnings or errors it generated, and a list of all the internal BibTeX functions it used

`.bst` the bibliographic style file, containing information on how to format the data BibTeX is given

The way the `.aux` file is used by LaTeX to communicate with BibTeX is as follows. The command `\bibliographystyle` in our `.tex` file causes a `\bibstyle` command to be added to the `.aux` file, which lets BibTeX know what format it should produce its output in. The command `\bibliography` in the `.tex` file causes a `\bibdata` command to be added to the `.aux` file to let BibTeX know where to find its data. Moreover, the `\bibliography` command is where

LaTeX inserts the data in the `.bbl` file. Any `\cite` or `\nocite` command causes a `\citation` command to be added to the `.aux` file so BibTeX knows what entries it needs to use.

Try running LaTeX then BibTeX on a `.tex` file which contains a `\bibliography` command which refers to a non-existent (or misspelled) `.bib` file.

5 Different styles

Bibliography styles are located in files that end in `.bst` (not to be confused with document styles, part of TeX document formatting). You will generally not write your own bibstyle, but you might on occasion need to edit one slightly to get *exactly* what you want. However, usually you'll find that you'll be happy with the files that are included standard as part of the LaTeX distribution, or that can be downloaded from [CTAN], or that particular journals or societies will make available to you.

The style files themselves will in general not be located where your document is, but in the `bibtex` directory in the LaTeX distribution. However, BibTeX will look first in the current directory, so if you do have a style file with the same name as a standard one but which is different it will use this first. Note, however, that this will just end up being confusing (since your `.tex` file seemingly refers to the standard style), and may breach copyright rules listed at the top of the style file.

Since there are plenty of styles to be found it pays to look around and try each of the types until you find one you're happy with and which meets all your requirements. (Like I said above, some journals will actually provide you with a style file to meet their requirements.) The main ones are

`plain` entries are sorted alphabetically and are labelled with numbers.

`unsrt` the same as `plain` except entries appear in the order in which they were first cited.

`alpha` entries are sorted alphabetically, and the labels are made from the author's or authors' (or editor's) name and the year of publication, like [GMS94].

`abbrv` the same as `plain` except names, month names and journal names are abbreviated.

Some of the (non-standard) BibTeX styles modify the citation and bibliography list items compared with normal LaTeX, so a corresponding LaTeX package needs to be used in these cases (with a `\usepackage` command in the preamble).

On this system, the bibliography styles are located in directories off

`/usr/local/teTeX/share/texmf/bibtex/bst/`

Try using each of these styles on a given bibliography, for example `course.tex`, and even try some of the other styles available. Have a look at how the labels change, and also how the entries in the reference list change as well. Also try using the name of a non-existent style, and see what BibTeX tells you.

6 The format of the `.bib` file

The first thing I should say is that unlike LaTeX, BibTeX does not care about cases. I'll remind you of this when it's important.

Now the best way to explain the format of entries in the `.bib` file is with an example:

```
@article{hownotto,
   title = "How Not to Write Articles",
   AUthoR = {Young, David S.},
   journal = "J. Not Important Research",
   volume = 1,
   year = 2002,
   pages = "2--3"
}
```

The first bit states that this entry is an article. There are various entry types (I'll get to them in a bit), but you must start each with the `@` symbol. After the entry type we put a left brace (as above) or `(`. Whatever we put here has to be matched at the end of the entry, though. It's usually a good idea to use one consistent style, but in the rest of this document I'll use both, to remind you that both are valid.

The `hownotto` is the citation key. If we want to refer to this article somwhere in our paper we would use `\cite{hownotto}`. In fact, BibTeX doesn't care about cases, so we could use `\cite{HowNotTo}` instead if we wanted. However, LaTeX *does* care about cases, so if we did this we would have to do it the same *everywhere* it occurred in our file, else we will get two different `\citation` entries in our `.aux` file, and BibTeX will complain about a "Case mismatch error". There are various schemes for coming up with citation keys; the Harvard system takes the author's surname (in lower case) follwed by a colon, then the year of publication. It's a good idea to find a scheme you're happy with and settle on it.

After the citation key, we put a comma. Then it's time for the fields.

I've given the entry above six fields. There are different fields for the different entry types, some of which are optional. A field consists of its name (for example, `title`), followed by an "=" character, followed by the text. There can be spaces around the "=" if you want, but it's not required.

The text itself is a string of characters, with no unmatched braces (and here this includes `\{` and `\}`), which is surrounded by a pair of braces or a pair of `"` characters. (Again, it's probably a good idea to stick to a consistent style, but

I'll chop and change.) Actually, if the text is a number we don't need the braces or quotation marks around it, for example with the year `2002` above. However, if the field is a numeric range, for example the pages `2--3` above, then the text is not numeric and must have the braces or quotation marks. (And remember, the information you give is formatted then passed on to LaTeX, which is why we have the two hyphens in te numeric range.)

We separate the fields by commas.

BibTeX is pretty easy-going when it comes to spaces, which includes end-of-line characters and tabs. Thus there is no need for the exact formatting of the entry above, and we could add spaces around `"` symbols and commas if we like. However, it's a good idea to use a personal style you're happy with and stick to it — you're less likely to make mistakes, and if you do make a mistake it'll be easier to find it.

As I said before, BibTeX does not care about cases. The field `AUthoR` shows this – any capitalization of it would work. Once again, it's a good idea to find a personal style and stick with it. (A lot of people capitalize the field type to make it stand out, for example `AUTHOR = "Young, David Scott"`. Personally, I think it's a good idea, but I never actually used it for some reason.)

When BibTeX encounters a key which is the same as one which it has already come across, it will complain, and tell you:

```
Repeated entry---line 47 of file refs.bib
 : @article{hownotto
 :                  ,
I'm skipping whatever remains of this entry
```

One last thing – the `%` character is not a comment character inside a `.bib` database file, unlike a LaTeX `.tex` file. However, BibTeX is pretty smart, and not recognizing it as a valid command will usually ignore it, at least between entries. Withing entries you may get an error message, however.

6.1 Entry types

This is where "play around with something until you're happy with it" comes into play. When you're entering something into a database you need to decide what type of entry it is, which will decide how it is dealt with by BibTeX. SOme things, like journal articles or books, are obvious; others take a bit of playing.

Each entry has three types of fields:

required means that if the field is left out BibTeX will give you an error message, and the entry will probably not be formatted correctly. Chances are if the field doesn't make sense you should choose a different entry type.

optional means it will be used if you include it, but you won't get an error if you don't.

ignored anything else. BibTeX will not include any information that is not *required* or *optional*. Thus, if you want to include a short note for an entry, you

could add the field `mynote`, say, and BibTeX would ignore it. On the other hand, if you mistype a field name, say `jurnal`, BibTeX will ignore that field, and will possibly complain that required information is missing.

The following list gives the entry types and the required and optional fields for each, along with a short description:

article An article from a journal or magazine. Required: `author`, `title`, `journal`, `year`. Optional: `volume`, `number`, `pages`, `note`.

book A book with a publisher. Required: `author` or `editor`, `title`, `publisher`, `year`. Optional: `volume` or `number`, `series`, `address`, `edition`, `month`, `note`.

booklet A work that is printed and bound, but without a named publisher or sponsoring institution. Required: `title`. Optional: `author`, `howpublished`, `address`, `month`, `year`, `note`.

conference The same as `inproceedings`.

inbook A part of a book, usually untitled. It may be a chapter, section etc. and/or a range of pages. Required: `author` or `editor`, `title`, `chapter` and/or `pages`, `publisher`, `year`. Optional: `volume` or `number`, `series`, `type`, `address`, `edition`, `month`, `note`.

incollection A part of a book with its own title. Required: `author`, `title`, `booktitle`, `publisher`, `year`. Optional: `editor`, `volume` or `number`, `series`, `type`, `chapter`, `pages`, `address`, `edition`, `month`, `note`.

inproceedings An article in a conference proceedings. Required: `author`, `title`, `booktitle`, `year`. Optional: `editor`, `volume` or `number`, `series`, `pages`, `address`, `month`, `organization`, `publisher`, `note`.

manual Technical documentation. Required: `title`. Optional: `author`, `organization`, `address`, `edition`, `month`, `year`, `note`.

mastersthesis A master's thesis. Required: `author`, `title`, `school`, `year`. Optional: `type`, `address`, `month`, `note`.

misc When nothing else fits.... Required: none. Optional: `author`, `title`, `howpublished`, `month`, `year`, `note`.

phdthesis A Ph.D. thesis. Required: `author`, `title`, `school`, `year`. Optional: `type`, `address`, `month`, `note`.

proceedings The proceedings of a conference. Required: `title`, `year`. Optional: `editor`, `volume` or `number`, `series`, `address`, `month`, `organization`, `publisher`, `note`.

techreport A report published by a school or another institution, often numbered within a series. Required: `author`, `title`, `institution`, `year`. Optional: `type`, `number`, `address`, `month`, `note`.

unpublished A document with an author and title, but not formally published. Required: `author`, `title`, `note`. Optional: `month`, `year`.

Have a play around and see how the different entry types get formatted. For example, how does **inbook** differ from **inproceedings**?

One other field is available for all these entry types: `key`. For some bibliographic styles we can use this to tell BibTeX a more appropriate string to use to generate the citation label than the author, editor or organization fields. For example, if we had

```
organization = "School of Mathematical Sciences",
key = "SMS"
```

then we would produce something like [SMS02], which is more acceptable than [Sch02]. It is also used for some entry types without an `author` or `editor` to help BibTeX decide on sorting.

Don't be misled by the `chapter` field — it indicates any sectional unit, such as a "Section" or "Part", not just a chapter. To change from the default output of "Chapter" include a field entry like `type = "Section"`.

In fact, don't be misled by any of the field names – they are meant to be functional names, and won't necessarily match up *exactly* with your particular entry every time, but just so long as they format cirrectly you should be fine.

The `edition` field should be entered as an ordinal word with a leading capital, for example `Second`. The capital will be lowered if necessary by the style file.

The `year` field does not have to consist solely of a four-digit number, but it must end with a four-digit number. For example, `year = "circa 2001"` is allowed.

Finally, since a field which BibTeX doesn't recognize is ignored, you can add your own comments to bibliographic entries, such as by using a field

```
mycomment = "This paper is by my supervisor, so I have to refer to it"
```

Go ahead and add some entries to a database. Play around with entries, deliberately misspelling the names of required or optional fields, for example `jurnal`. See how the output is affected, and look at what errors or warnings BibTeX gives you. Finally, some of the entry types have for example `author` or `editor` — try adding both.

6.2 Name considerations

If you enter a name as a string as you would normally say it, such as "David S. Young" or "Linus van Pelt", BibTeX will have to try to work out which part is the surname, for example to work out how to sort or to give it a label. It will parse (break up) the list of words into three groups:

- the *von* part will begin with the first non-capitalized word, and will include everything from then up to (but not including) the next capitalized word. This part may be empty.
- the *Surname* consists of everything after the `von` part. If there is no `von` part then the surname is just the last word.
- the *Firstname* will be everything up to the `von` part or the surname (as worked out above).

For example, `David S. Young` will be grouped as "David S.", nothing, then "Young", whereas `Lucy van Pelt` will be grouped as "Lucy", "van", then "Pelt". BibTeX is actually quite smart, and will occasionally change spaces into unbreakable spaces (which use the LaTeX command `~`), for example next to initials.

BibTeX's guesswork usually works quite well, but sometimes can be outsmarted by tricky names.

To deal with tricky surnames, you can enter names in the form `von Surname, Firstname`. This will cope with, for example "Jon Bon Jovi" (which would be entered as `Bon Jovi, Jon`). It will also work for names that BibTeX would normally have no trouble with, such as "David S. Young" (which would be entered as `Young, David S.`).

We can use the same method to deal with tricky "firstnames", such as names which include non-capitalized parts. Say someone's name was "Giulio de Giorgi Caranti", where their surname is "Caranti", and "de Giorgi" is their middle name, so part of their first name. Then we would enter `Caranti, Giulio de Giorgi`.

People who use a suffix like "Jr" or "III" generally precede it with a comma. To accommodate that, enter their name in the form `von Surname, affix, Firstname`, such as `Nash, Jr, John Forbes` to get what you're after. If someone does not use a comma, simply include the suffix with the surname, enclosing both with braces: `{Young Jr}, David`.

BibTeX knows how to deal with hyphenated names, in particular how to abbreviate them.

Sometimes we want the `von` part to be capitalized, but still to be treated as the middle bit (for example, in sorting). Here we need to trick BibTeX. For example, if we enter `James {\uppercase{d}e La} Porta` BibTeX considers the middle "De La" part to be the `von` part, since it does not understand the LaTeX `uppercase` command and only sees the lower-case `de`. Then, for example, if we abbreviate the name, we get J. De La Porta, as different from the J. D. L. Porta which we would otherwise have gotten.

Finally, you will occasionally find yourself referring to someone whose surname comes *first*, so you will want them to be sorted in the reference list (and have a label coming from) the first-listed name. To "trick" BibTeX here, enclose the whole name in braces, as if their name was just one big surname; and entry with

```
author = "{Mao Zedong}"
```

will appear between entries with authors `Gus Logie` and `Keith Miller`, and will be labelled something like `[Mao70]` (in the relevant bibliographic styles).

In summary, there are three ways you can enter names:

`"First von Last"`	e.g. `{James van der Beek}`
`"von Last, First"`	e.g. `"van Halen, Eddie"`
`"von Last, Jr, First"`	e.g. `{di Caprio, III, Leonardo}`

6.3 Multiple names

When entering multiple names, separate them with `and`, even if there are three or more. This way BibTeX will know what the surnames are in order to form the correct label and sort properly:

```
author = "Goossens, Michel and Mittelbach, Frank and Samarin, Alexander"
```

If you have more authors or editors than you want to type, end the list of names with `and others`; this will generally be converted to "et al." by most bib styles.

If there is an occurrence of "and" which you don't want BibTeX to treat as a name separator, but which actually forms part of a name – for example, "Young and Jackson, Inc." – then enclose everything which must stay as one unit in braces:

```
EDITOR = "Toohey, Red and {Young and Jackson, Inc.}"
```

This trick can be used to prevent the breaking up of names that shouldn't be broken up:

```
author = {{Australian Bureau of Statistics}}
```

If this is entered without the extra pair of braces, would normally be considered to be by someone whose surname is "of Statistics" and whose firstnames are "Australian Bureau", and would be sorted and labelled accordingly.

6.4 Titles and capitalization

The bibliography style will decide whether a title should be capitalized; usually titles of books are capitalized, titles of articles are not. The way to enter a title is as if it would appear if it would be capitalized:

```
TITLE = "The Queen of the Night"
```

This is because BibTeX will only change capitals to lower case if they are not needed, but not the other way around.

Sometimes you will have a capital that must remain in a title, for example a proper name. In this case enclose the letter in braces; you can do just the letter itself, or the whole word:

```
TITLE = "Lampoons of {L}amport"
TITLE = "Lampoons of {Lamport}"
```

6.5 Dealing with LaTeX commands

BibTeX sometimes can get confused by LaTeX commands, such as those that generate accented characters. So that it doesn't get confused, surround the character by braces so that BibTeX knows it doesn't have to worry about it:

```
"Kurt G{\"{o}}del"
```

However, note that the braces must not be enclosed by more braces, and the backslash must be the first character inside the braces. This then interferes with using the braces to stop BibTeX converting characters to lower case, though! For example, if the above was part of a `title` field we would have to enter:

```
TITLE = "The Life of {Kurt} {G}{\"{o}}del"
```

Part II
Nifty things

7 Cross-references

Cross-referencing is another convenient way to avoid typing and ensuring that all data is consistent.

Suppose that you cite several parts of another cited work, for example different papers in the same conference proceedings, or different parts of a book which have different titles. Then you can make one entry for the larger work and refer to that entry in the entries for the included works by using the `crossref` field. Any fields which are already in the larger work do not then need to be given for the others if they are the same. However, every field that is required must be in one of the two works.

```
@INPROCEEDINGS{young:unimportant,
  author = "Young, David S.",
  title = "Some Unimportant Friends of {L}eslie {L}amport",
  pages = "333--334",
  crossref = "unimp:conf"
}
...
@PROCEEDINGS{unimp:conf,
  title = "Unimportance: 1st Annual Conference on Trivia",
  booktitle = "Unimportance: 1st Annual Conference on Trivia",
  editor = "One, N. E.",
  year = "2000"
}
```

There are three things to note about cross-referencing. Firstly, a cross-referenced entry (like the proceedings above) must occur *after* any entry that

refers to it. Secondly, a cross-referenced entry may not cross-reference another entry. Finally, if an item is cross-referenced by at least two other entries then it will automatically appear in the reference list, even if it hasn't been `\cite`d.

Here we have included the `booktitle` field in the `@PROCEEDINGS` even though it is not needed there because it will be needed in everything that refers to it. It is happily ignored in the reference to the proceedings itself.

8 Abbreviations and preambles

Sometimes you will have a string that you find yourself using often, and you wish you could just abbreviate it. For example, if you were writing a paper on things which aren't very important to anyone, you might find yourself entering

```
journal = "Journal of Unimportant Research"
```

over and over again. You can define an abbreviation for the text string that you can use in its place. If you defined `JUR` as an abbreviation for `Journal of Unimportant Research` then the above field entry could be given instead as

```
journal = JUR
```

Note that we do not surround the name of the abbreviation by quotation marks or braces, else BibTeX will think it is the title! The name of an abbreviation has to start with a letter and cannot contain a space or any of the following characters:

```
" # % ' ( ) , = { }
```

Some abbreviations are predefined by the bibliography style. For example, societies like to include abbreviations for all their journals. In addition, nearly every style contains abbreviations for the months using three letter abbreviations. For this reason, when entering a month you should use this three-letter abbreviation instead of the month itself, for example

```
month = apr
```

instead of

```
month = "April"
```

To define your own abbreviation, use the `@string` command:

```
@string{JUR = "Journal of Unimportant Research"}
```

The outermost braces can be replaced with parentheses, while the quotation marks may be replaced with braces. Moreover, since BibTeX ignores case, the above is equivalent to

```
@string{jUr = "Journal of Unimportant Research"}
```

The `@string` command must come before the first place the abbreviation is used, but other than that it can come anywhere before or between entries. The obvious place to put them then is at the start of the `.bib` file. You can even create a `.bib` file which just contains abbreviations, and make sure it is the first file listed with the `\bibliography` command.

Also note that a `@string` command in a `.bib` file overrides anything defined in the bibliographystyle, so you can redefine an abbreviation such as `Apr` if you're not happy with it.

We can also concatenate abbreviations. For example, suppose that the above journal has a second series. Rather than having to type out "Journal of Unimportant Research, Series 2", or defining a new abbreviation for it, we can instead concatenate the extra bit onto the previously-defined abbreviation with a `#`:

```
journal = jur # ", Series~2"
```

An example abbreviation file is one produced by the American Mathematical Society:

```
/usr/local/teTeX/share/texmf/bibtex/bib/ams/mrabbrev.bib
```

In a similar vein is the BibTeX command `@preamble`. This is especially useful to pass to LaTeX any LaTeX commands that need defining. The argument of the `@preamble` is a set of strings, concatenated by the `#` character, each of which is a valid LaTeX command. For example

```
@preamble{ "\providecommand{\sortignore}[1]{} "
        # "\providecommand{\swapthings}[2]{#2#1} " }
```

will end up providing two commands to LaTeX.

The first command, `\sortignore`, is actually an example of a very handy trick to get BibTeX to sort things properly. Suppose that you had two volumes of a book, but from different editions — say the first volume was a 1990 edition of a 1986 original, but the second volume was first published in 1988 and hasn't had a second edition. Ideally we would like the second edition to occur later than the first in a reference list, but BibTeX will usually sort by year, putting the second volume first. To get past this, in our `.bib` file we could have

```
volume = 1, year = "{\sortignore{1986}}1990"
...
volume = 2, year = "{\sortignore{1988}}1988"
```

LaTeX will happily ignore the arguments to `\sortignore`, producing "1990" and "1988" in the output as we want. BibTeX, on the other hand, will ignore the LaTeX command `\sortignore` *but not its argument* when sorting, and will try to sort them according as 19861990 comes before 19881988, which is what we want.

9 Maintaining and updating your database

Well, you've got a file, or a whole lot of files, which contain your bibliographic data. You'll find that a lot of electronic sources of bibliographic information can give you that information in BibTeX format; for example, the on-line mathematical reference and review database MathSciNet [MSN]. Thus, we can happily keep our bibliographic database expanding.

The most obvious thing we need to do is keep track of what citation keys refer to what.

It's easy enough to make a LaTeX document which will output all of our references, simply by putting a `\nocite{*}` command in a file which refers to all of our bibliographic files:

```
\documentclass{article}
\begin{document}

\nocite{*}

\bibliographystyle{alpha}                % or whatever
\bibliography{abbrev,refs,xample,misc}  % again, or whatever

\end{document}
```

However, this will not tell us what *citation key* we actually chose for each of these, and depending on what we were thinking, they could be counter-intuitive, and not easily gleaned from the reference itself!

To fix this problem, there are a few packages that can be used.

9.1 showkeys and showtags

I like to combine the above approach with the package `showtags`. This puts a box above each item in a reference list whioch contains the citation key used for that item. In our case, we can combine it with our complete bibliographic file:

```
\documentclass{article}
\usepackage{showtags}
\begin{document}

\nocite{*}

\bibliographystyle{alpha}                % or whatever
\bibliography{abbrev,refs,xample,misc}  % again, or whatever

\end{document}
```

A similar package is `showkeys`. This package (which should also come with the LaTeX distribution — if not, check out [CTAN]) can be used more generally to keep track of what you are `\cite`ing, `\label`ing and `\ref`ing throughout your

document. Whenever you use one of these commands, it will include in your output a small box with the relevant label.

> Try each of these packages on a LaTeX file, even if it doesn't contain *all* of your references, just to see their effect.

The `showkeys` package also has optional arguments `notref` and `notcite` which can be used to turn off the printing of the labels-in-boxes near `\ref` and related commands, or `\cite` and related commands. For instance,

```
\usepackage[notref,notcite]{showkeys}
```

will produce the boxes only adjacent to `\label` and related commands, which does include the entries in the reference list. The package should have documentation on your system — look in the folder

```
/usr/local/teTeX/share/texmf/doc/latex/tools/
```

for documentation on showkeys and other useful LaTeX packages.

9.2 biblist

This package works much like showkeys and showtags, in that it produces a reference list with the citation keys shown. It is formatted slightly different, of course.

If you `\usepackage{biblist}`, your `\documentclass` must be `article`, and you cannot use `twocolumn` or `multicol`. With this package, you do not need to include the `\nocite{*}` command, as by default all the entries are listed; if you do use explicit `\nocite` commands, only those entries will be listed:

```
\documentclass{article}
\usepackage{biblist}
\begin{document}

\bibliographystyle{alpha}                % or whatever
\bibliography{abbrev,refs,xample,misc}  % again, or whatever

\end{document}
```

Here you need only run LaTeX, BibTeX, LaTeX. (The final LaTeX run which is usually needed is not needed here. Of course, it doesn't hurt anything if you accidentally do it out of habit!)

9.3 Other tools

The following tools are less commonly available, but you might find them useful if you have them. For more information about them, or to obtain them, check out [CTAN].

aux2bib is a Perl script which, given an `.aux` file and the `.bib` files it refers to, creates a portable `.bib` file.

bibkey is a C-shell script that searches `.bib` files for entries containing a given keyword.

bibview can be used with an X-window graphics terminal to manipulate BibTeX databases.

looktex is a generalization of bibkey.

makebib makes an exportable `.bib` file from a given set of `.bib` files and an optional list of citations.

printbib creates a `.dvi` file from a `.bib` file, sorted by citation key.

10 Other useful packages

10.1 Changing the way citations appear

The default way for citations to appear is as a comma-separated list enclosed in square brackets. This is controlled by the internal LaTeX command `\@cite`, which is defined as follows:

```
\renewcommand{\@cite}[2]{%
              [{#1}\ifthenelse{\boolean{@tempswa}}{,#2}{}]}
```

(Here the temporary Boolean variable `\if@tempswa` is set to `true` if the `\cite` command has an optional argument.) It is possible to redefine this by hand, for example to get exponent-like (raised) references:

```
\renewcommand{\@cite}[2]{%
              {$^{#1}$\ifthenelse{\boolean{@tempswa}}{,#2}{}}}
```

If you're confident, you can try redefining `\@cite` to get your desired effect.

However, there are packages which already exist which can make the job easier for you:

- The cite package can collapse a list of three or more consecutive numbers into a range, for example [5,6,7,9,6,4] becomes [5–7,9,4,8]. An extra command `\citen` gives the numbers without the brackets, for example "`See also ref.~\citen{companion}`".
- The citesort package is like the previous package except it sorts the references before collapsing (removing duplicates), so the previous example becomes simply [4–9].
- The overcite package is like cite, but displays the citations as superscript numbers which are compressed (but not sorted like citesort)

Some bibliography styles define supplementary commands which are like `\cite` but which format the entry differently. For example, the chicago style has the following commands:

`\cite`	full author list and year: (Goossens et al. 1994)
`\citeA`	only full author list: (Goossens et al.)
`\citeN`	full author list and year, for use in flowing text: Goossens et al. (1994) state that ....
`\citeyear`	year only, in parentheses

There are also commands to give abbreviated output: `\shortcite`, `\shortciteA`, `\shortciteN`. In addition, each command has an equivalent without parentheses, obtained by appending `NP` to the name, for example `\citeNP`.

10.2 Multiple bibliographies in one document

Occasionally you will want to include multiple bibliographies in one document, for example with conference proceedings with many articles, or parts of a book with different authors. Unfortunately, with BibTeX multiple `\bibliography` commands give you the same reference list each time, each containing *all* the references from the document.

10.2.1 The chapterbib package

To use this package, you need to have included files with the LaTeX `\include` command. It is in these files that you are allowed to have `\bibliographystyle` and `\bibliography` commands (with only a single `\bibliography` command per file). Any occurrence of either of these commands in the root file will be ignored.

If you want to have a `\cite` command in the root file, then you have to provide a stand-alone bibliography within that file using the `\thebibliography` command, as discussed in the "Doing it manually" section.

You compile a document using this package as follows:

1. Run LaTeX on the root document, as you would normally. This will produce a `.aux` file for each of the `\included` files.
2. Run BibTeX for each `\included` file, that is, for each of the `.aux` files produced by the previous step, to produce individual `.bbl` files.
3. Run LaTeX on the root file again...
4. ...and run LaTeX on the root file again, as you would normally.

10.2.2 The bibunits package

This package can only be used (without modification) on Unix systems.

This package generates separate bibliographies for different specified units of the text, chapters, sections, or `bibunit` environments, specially defined by this package. It will separate the citations into separate files based on these units.

The command

```
\bibliographyunit[unit]
```

specifies for which document unit the reference lists will be generated — `unit` can be either `\chapter` or `\section`.

If the optional argument is not given, the document units must be specified as `bibunit`s (see below). Any citation made between such units will appear in the document's global reference list.

In either case, the commands `\bibliographystyle` and `\bibliography` specify the BibTeX style and files which are used by default in the specified units. In each unit, these can be overridden by the commands `\bibliographystyle*` and `\bibliography*`.

A `bibunit` is started with the command

```
\begin{bibunit}[style]
```

The `style` is optional, and specifies a style different from the default (if any). The end of a `bibunit` is marked by a

```
\end{bibunit}
```

command. Somewhere within each `bibunit` there must be a

```
\putbib[bibtex-files]
```

command, specifying which bibliographic data files (`.bib` files) to use for that `bibunit`. If the optional argument is not given, the default file or files are used.

When you first run LaTeX on a file which uses the package bibunits, it generates a file `filename.i.aux` for each document unit, where `i` is the sequence number of the unit. (This is why it will not work on many non-Unix systems.)

After this first run of LaTeX, you must run BibTeX on each of these `.aux` files. After this, run LaTeX twice as per usual.

Try using chapterbib and bibunits.

The chapterbib is the easier of the two, as it is the most intuitive. On the other hand, the bibunits package is more powerful and less restrictive (for example, you do not need to have separate `\included` files).

11 Creating and editing style files

It's probably best not to create a bibliographic style file from scratch, but you might want to edit one slightly to get exactly what you want (if it doesn't already exist somewhere out there). Again, remember if you edit an existing file, you'll probably be required to change the name of the file to avoid confusion.

Let's start with an example of something that you might want to do. Say you were using a style that ou were perfectly happy with a given style except for its habit of changing titles to lowercase for some entry types, such as article.

Looking at the `.bst` file, you notice that it starts with a list of allowed entries, defined with an `ENTRY` command, then a list of names of integer variables, given by `INTEGERS`, then a whole lot of definitions of functions, given by the command `FUNCTION`. Among these is the command `FUNCTION {article}`, and you surmise that the stuff immediately after this tells BibTeX how to format an `article` entry. Looking through, it seems that `format.title` is responsible for the dirty work done to `title`, and just having a quick look at `FUNCTION {book}`, we see that this entry type indeed does not use the function `format.title`.

The function itself is defined earlier in the file by

```
FUNCTION {format.title}
{ title empty$
    { "" }
    { title "t" change.case$ }
  if$
}
```

The `"t"` argument tells `change.case$` to change everything to lowercase except the first letter. If we change this function we should be happier; we change it to

```
FUNCTION {format.title}
{ title empty$
    { "" }
    { title }   % modified from original
  if$
}
```

with a little note to remind us what we've done. We'll need to change the functions `format.edition`, `format.chapter.pages`, `format.thesis.type` and `format.tr.number` as well to be truly happy.

A lot can be gleaned about how BibTeX works just be looking at a style file. Importantly, functions use the mathematicians' "Polish notation" — arguments are written immediately preceding a function. For example,

```
edition "l" change.case$ " edition" *
```

tells BibTeX to first change the case of the `edition` string constant to lowercase (the `"l"` argument), then to concatenate (the `*` function) this with the string " `edition`" (space and all). A full list of the 37 built-in functions can be found in [GMS94]. (If the name of a built-in function contains a letter then it ends with a `$`) The most useful are `*`, `change.case$` (which takes two arguments, the first of which is the string to be formatted, the second is `"t"`, `"l"` or `"u"` telling it how to format, with `"u"` meaning to uppercase), `add.period$`, which adds a period to its string argument if it then does not occur before `.`, `?` or `!`. The function `if$` is useful for controlling flow; it takes three arguments, the first numeric, the second and third function calls, and executes the first function if the numeric argument is non-zero, and executes the second function otherwise. Functions such as `missing$` return numeric values which are either 1 or 0.

Having changed the formatting of titles we might get more adventurous, and decide that we want to add a new field type `annote`, which adds an annotation to each entry. Looking through the file we decide that the function `fin.entry` is the obvious one to alter:

```
FUNCTION {fin.entry}
{ add.period$
  write$
  newline$
}
```

After the period is added we'd like a newline, then our citation key, and then our annotation. We'll, after we add `annote` to the field types defined with `ENTRY` earlier in the file, here's what we might do:

```
FUNCTION {fin.entry}
{ add.period$
  write$
  newline$
  "\begin{quotation}\noindent{\sc Key:\ }" cite$ * write$
    annote missing$
    'skip$
    { "\\{\sc Annotation:\ }" write$ annote write$ }
  if$
  "\end{quotation}" write$
  newline$
}
```

Now, when we've finished writing the "normal" entry, we put a period (if necessary) and a newline, then we begin a `quotation` environment. In the this we put "Key: " followed by our citation key, and then, if we have an `annote` field, "Annotation: " followed by its value. Then we finish the `quotation` environment.

Have a look inside a `.bst` file. See if you can understand how a BibTeX run would flow, using this style.

Oren Patashnik has also written a more comprehensive file called "Designing BibTeX Styles", which is available at

`/usr/local/teTeX/share/texmf/doc/bibtex/base/btxhak.dvi`

and which includes a full description of the different BibTeX style file functions and commands.

11.1 makebst

Although we've now got the bibstyle-changing bug, a cursory glance at a single `.bst` file indicates that writing one from scratch would be very difficult indeed.

That's why the makebst program exists. This takes a master file (which ends with `.mbs`), interactively determines the format of the bibliographic entities (which determines which options from the master file to use), and then processes the resulting file with the DOCSTRIP utility to produce a brand spanking new `.bst` file.

The program itself is actually a `.tex` file, and it is run by invoking `tex` or `latex` on it, as in

```
tex makebst
```

The program will first ask for the name of the master file, and then for a name for the output file. After this the program will go through all the available options, and will finally ask if the master file should be run. If the you say yes here, the program finishes the processing and produces a `.bst` file with the name you have specified. If you say no the program will produce an output file with the extexnsion `.dbj`. This is a *batch job file* which can be edited later if you want. When you want to produce the `.bst` file simply run tex on this batch job file, for example

```
tex mystyle.dbj
```

The default option for the `.mbs` file (`merlin.mbs`) will give the user quite a few good options for customization. There are also available various language-specific `.mbs` files, the output from which will need to included with other `.bst` files as they only deal with language formatting, not the bibliographic formatting.

On this system the documentation for makebst is available at

```
/usr/local/teTeX/share/texmf/doc/latex/custom-bib/makebst.dvi
```

The file `makebst.tex` and the `.mbs` files are located in the directory

```
/usr/local/teTeX/share/texmf/tex/latex/custom-bib/
```

Try running

```
tex /usr/local/teTeX/share/texmf/tex/latex/custom-bib/makebst
```

using the default `merlin.mbs` and your own preferences to construct your own `.bst` file. Also run makebst on a language file, such as `english.mbs`, and see the different questions you are asked, and the different form of the resulting `.bst` file.

12 If BibTeX can't get it perfect for you ...

After all this, if you can't get *exactly* what you want using BibTeX, you can always use BibTeX to get close to what you want, and then edit the `.bbl` file to something that is perfect for you. Be warned, though; if you accidentally

run BibTeX again you'll lose all your customizations! For this reason, it is a good idea to actually include the information from the BibTeX-produced `.bbl` file in your `.tex` file (in place of the `\bibliographystyle` and `\bibliography` commands) before you edit it.

For example, when creating this file, I had an entry

```
@misc{ctan,
title = "Comprehensive {\TeX} {A}rchive {N}etwork",
note = "\verb|http://www.ctan.org/|",
key = "{CTAN}"
}
```

in my file `refs.bib`. However, even though I specified a four-letter key (the full acronym of the web cite) within braces, BibTeX still shortens this to three letters. After my final BibTeX run I then changed the

```
\bibitem[{CTA}]{ctan}
```

line in my `.bbl` file to

```
\bibitem[{CTAN}]{ctan}
```

This is also a good way to make your file self-contained, not needing to refer to a separate `.bib` file, for example if you need to submit a paper electronically to a journal.

Of course, wait until the final version of the bibliography (after the final BibTeX run) before doing this!

References

[CTAN] *Comprehensive TeX Archive Network*, `http://www.ctan.org/`.

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin, *The LaTeX companion*, Addison-Wesley, Reading, Massachusetts, 1994.

[Lam94] Leslie Lamport, *LaTeX: a document preparation system*, second ed., Addison-Wesley, Reading, Massachusetts, 1994.

[MSN] *MathSciNet*, `http://www.ams.org/mathscinet/`.

[TUG] *TeX Users' Group*, `http://www.tug.org/`.