

BibTeX Notes

Paul Jungwirth

March 25, 2006

Intro

This bit of documentation is meant to supplement “Tame the BeaST” by Nicholas Markey, an excellent introduction to all things BibTeX, and as far as I know the only public documentation on .bst files. I rely heavily on his work and occasionally point out errors. Probably you should read his paper before you read this. For clarity’s sake, I generally steal his examples. My references are all to version 1.3 published 16 October 2005.

Brace Depth

Brace depth refers to strings from the .bib file, especially the `title` and `author` fields. A character’s brace depth is equal to the number of surrounding braces. For example, you might have a book titled:

```
"The {Latex} {C}ompanion"
```

Here, most of the string has brace depth 0, but `Latex` and `C` have brace depth 1.

Special Characters

A special character is an escape sequence immediately surrounded by braces, like `{\’E}` or `{\LaTeX}`. In spite of the braces, a special character is considered brace depth 0. However, special characters must themselves appear

at brace depth 0; if they are surrounded by other braces, they are not considered special characters. For example, suppose we wanted to correct the example above by using the `\LaTeX` macro. Then we write:

```
"The {\LaTeX} {C}ompanion"
```

Here, `C` is at brace depth 1, and everything else, even the special character `\LaTeX`, is at brace depth 0. Now suppose we wrote the title like this:

```
"The {\{\LaTeX}} {C}ompanion"
```

In this case, `\LaTeX` is not a special character, because it starts at brace depth 1, not 0. The string `\LaTeX` is considered to be at brace depth 2.

If an escape sequence lacks braces entirely, as in `The \LaTeX Companion`, it is not a special character, just plain text.

Note that an apparent contradiction appears on page 21 of “Tame the BeaST.” It correctly remarks that special characters must appear at brace depth 0, but then it states, “Anything in a special character is considered as being at brace depth 0, even if it is placed between another pair of braces.” The “even if” is a null hypothesis, because special characters are only special characters at brace depth 0.

purify\$

The `purify$` command reduces a string to only spaces and alphanumerics. Hyphens and tildes become spaces, and all other weird characters are removed. Inside of special characters, even spaces are removed. Thus `t^ete`, `t{\^e}te`, `t{\^e }te`, and `t{\^e }te` all become `tete`.

text.prefix\$

You can grab a string’s first `n` characters with `text.prefix$`. Braces are not included in the count of characters, but they are still output. Their closing braces are also output, appended, if necessary, to the end of the result. Special characters count as one character, but of course they must be at brace depth 0 to count as special characters. Thus, the first character of `{\’E}cole` is `{\’E}`, but the first character of `{{\’E}}cole` is `{{\}}` (the backslash is the character).

text.length\$

This command returns the number of characters in a given string. Special characters count as one character, and braces don't count at all. Thus, `a b c` is 5, `{a}` is 1, and `{\ 'a}` is 1, but `{{\ 'a}}` is 3. Since the last example isn't a special character, the `\ 'a` are counted as three separate characters.

substring\$

Given a string, a start position, and a number of characters, this command returns a substring. The position of the first character is 1. Unlike the commands above, `substring$` knows no secret meaning for "character." Every character is alike. Thus, the substring of `{\LaTeX}` from position 2 of length 3 is `\La`. (Note that "Tame the BeaST" on page 32 erroneously says the result is `\LaT`: 4 characters.)

sort.key\$

Reading "Tame the BeaST" page 32, I thought this variable was set automatically by some internal algorithm. It isn't. You must set it yourself (assuming you're writing a style file), with something like this:

```
title 'sort.key$ :=
```

change.case\$

Depending on the option, `change.case$` will set all letters to uppercase or lowercase, or (with the "t" option) set all letters to lowercase while leaving the first letter unchanged. It only affects letters at brace depth 0. Therefore special characters are changed, but characters inside regular braces are not. For example, with the "t" option, `t\^Ete` becomes `t\^ete` and `t{\^E}te` becomes `t{\^e}te`, but `t{{\^E}T}e` remains `t{{\^E}T}e`.

When you use the "t" option, if the first character is a special character, the whole special character retains its original capitalization. Thus, `{\ 'E}GAD` becomes `{\ 'E}gad` and `{\LaTeX} Companion` becomes `{\LaTeX} companion`.

Surviving Sorting

When you compose your .bib file, you should expect that the bib style will munge your entries to set the `sort.key$`. Probably it will run `purify$` on them. To ensure proper sorting, you should write your .bib entries so all the necessary data remains intact.

Surviving Capitalization

Your bib style will probably also run another transformation, at least on the title. It will use `change.case$` with the "t" option to set all letters after the first to lowercase. Since `change.case$` only affects letters at brace depth 0, use can use braces to preserve desired capitals. If you embed commands like `\LaTeX`, this is essential.

Surviving Sorting and Capitalization Together

Since most people want to use one .bib file with multiple .bst files, any style should basically honor the brace-tricks used to survive sorting (a.k.a. `purify$`) and capitalization (a.k.a. `change.case$`). You should write your bibliography entries with these two functions in mind. Suppose your bibliography includes this:

```
title = "The \LaTeX Companion"
```

That will fail, because `change.case$` will produce `\latex`, which is a command `LaTeX` doesn't know. It will also lowercase `Companion`. You could instead try:

```
title = "The {\LaTeX} {C}ompanion"
```

This is even worse! It does not solve the `change.case$` problem (except for the `C`), because `change.case$` still alters special characters. In addition, since `purify$` removes special characters, it will sort as `The Companion`, which will probably deposit it in the wrong place. To solve both problems at once, do this:

```
title = "The {{\LaTeX}} {C}ompanion"
```

Now `change.case$` will leave `\LaTeX` alone, since it is at brace depth 2, and `purify$` will pass it through as `LaTeX` (without the backslash), so it will sort properly.

Surviving Author Parsing

The `author` field endures similar munging, at least for `sort.key$`, but it is also parsed to find the first name, last name, and any intermediate “von”-like bits. This happens inside `format.name$`. If you enter the name as `von Last, First`, you needn’t worry as much, but if you give `First von Last`, you must be aware of how BibTeX parses the field.

There must be a `Last`, so it gets the last word in the field. Then BibTeX examines each word, from left to right. As long as the words are capitalized, they go to the `First`. The first non-capitalized word begins the `von`. To determine if a word is “capitalized,” BibTeX finds its first 0-depth character, including special characters, and checks whether it is uppercase or lowercase. If it is uppercase, the word is capitalized; otherwise, it is not. If there are no 0-depth characters (because the whole word is in braces), the word counts as capitalized and goes to the `First`.

Once the `von` begins, words can fall into either the `von` or the `Last`. These remaining words are examined in reverse order, from right to left. As long as the words are capitalized or in braces, they go to the `Last`, but as soon as a lowercase word is found, it and all remaining words go to the `von`.

A simpler way of putting this is that all contiguous capitals at the beginning are `First` and all contiguous capitals at the end are `Last` (where braced words count as capitals), and everything left is `von`. Note that capitalized words can still wind up in the `von` if they are surrounded by lowercase words. If all the words are capitals, then everything but the last word goes into the `First`. The last word always counts as a capital so that `Last` has at least one word, but the `First` may be empty if the first word is lowercase. A lowercase last word can still accumulate more words in the `Last`, as long as they are themselves capitalized.

One more wrinkle: you can group words by enclosing the intervening space in braces, like this: `Jean de La{ }Fontaine` or this: `Jean de {La Fontaine}`. BibTeX will treat these joined words as a single word. So in this first example, the algorithm will start by sending all of `La{ }Fontaine` to the `Last`.

For some names, you want to avoid all this parsing entirely, so that the

whole name goes into Last. In that case, you can enter names entirely surrounded with braces, like this:

```
author = "{Gregory the Theologian}"
```