Case changing: LaTeX reaches Unicode-land

Joseph Wright, LaTeX Project Team

1 Introduction

The concept of letters having case is familiar to speakers of several languages, most obviously those from Europe using Latin, Greek or Cyrillic scripts. The ability to convert between upper and lower case, *case changing*, is something we might take for granted both for people and for computer systems. However, there are subtleties that a careful implementation needs to take into account.

The Unicode Consortium have defined four different case-related operations that are required to support the range of applications that come up in practical use cases:

- Upper casing
- Lower casing
- Title casing
- Case folding

Here, *title* casing means making the first letter (broadly speaking) upper case, then making the rest of the text lower case. Case folding means removing case, and is needed for programmers: often lower casing is used, but this is not appropriate for true caseless text comparisons.

Unicode have also identified that case changing is not a simple fixed operation: depending on the context around a character, the right outcome can vary, while different languages can have different rules. All of this means that the TeX primitives `\lowercase` and `\uppercase` are *not* suitable for case changing with the variety of text we might see today.

2 An expl3 implementation

In 2015, I spoke at the TUG meeting about an expl3 implementation that sought to provide a full set of Unicode-compliance case changing tools. At that time, only Unicode engines (XeTeX and LuaTeX) were supported. However, the code could provide all of the requirements of Unicode in an approach that works by expansion: this meant that one could case change text *inside* an `\edef`.

Since that talk, the LaTeX team have refined ideas about the future of LaTeX, meaning that the case changer needed to be extended to work with pdfTeX. It also needed to be able to carry out something similar to `\protected@edef` by expansion. Both of those ideas have been covered over the past few years.

The expl3 implementation now also incorporates ideas from David Carlisle's textcase package. This is

doi.org/10.47397/tb/43-2/tb134wright-case

mainly about being able to exclude content from case changing: math mode material should never be case changed, and it's important to be able to mark individual items as unaffected by case operations. Both of those ideas are relatively easy to do by expansion, as we need to examine each token anyway.

3 Bring it to LaTeX 2$_\varepsilon$

Since `\uppercase` and `\lowercase` have long been known to have limitations, I am not the first person to look at supporting the needs of different languages. There have been a number of clever approaches to getting the required mappings from implementations using the TeX primitives. However, the new code provides a single consistent interface: it can handle different languages, multiple scripts and so on without needing to load potentially incompatible code.

More importantly for the team, we needed to look again at how active characters are handled in pdfTeX. The change, to use engine protection for these active bytes, makes life a lot easier in general but breaks the previous approaches to case changing these characters. The expl3 code, in contrast, works fine with the protected definitions. So this drove the change.

For users, the long-standing `\MakeUppercase` and `\MakeLowercase` commands stay unchanged: only the implementation has been adjusted. Titlecasing has some subtleties that mean it needs a dedicated document command, so now it has one: `\MakeTitlecase`. The command `\NoCaseChange`, originally defined by `textcase`, is also now integrated into the kernel.

For package authors, we have added a command `\AddToNoCaseChangeList` to add commands to the set which are skipped for case changing: things like `\ref` and `\label` are already there. We have also provided `\CaseSwitch`, a command that selects between different outcomes (no change, upper, lower, title case): this is useful if you have something that doesn't expand to text but where you want a different result inside and outside of case changing. Finally, we have added `\DeclareCaseChangeEquivalent` for situations where a package author needs to entirely swap the functionality of a command inside a case changing context.

4 The data: pdfTeX

One minor wrinkle is the data support, particularly for pdfTeX. For the Unicode engines, we can read all of the data automatically. For pdfTeX, we don't have `\lccode` and `\uccode` storage for the whole of Unicode, only for the 8-bit range. That means that most of the case changing data has to be stored in macros. To avoid wasting a lot of memory, only code points that are typically typeset with 8-bit engines are included here. That does mean that it's possible a few get missed: if there is something to add, please let us know.

5 Looking ahead

For the Spring 2022 release, we have not included support for language variation in the document command interface. But that's on the agenda, and likely to appear in the Fall 2022 LaTeX 2$_\varepsilon$ update. The most likely approach there is as an optional argument to `\MakeUppercase`, *etc.*: watch this space.

◇ Joseph Wright
Northampton, United Kingdom
joseph dot wright (at)
morningstar2.co.uk

◇ LaTeX Project Team