ctablestack.dtx
Catcode table stable support

David Carlisle and Joseph Wright

2015/10/01

Contents

1 Overview

This small package adds support for a stack of category code tables to the core support for LuaTEX provided by the LATEX kernel and available for plain users as `ltluatex.tex`. As such, the code here may be used with both plain TEX and LATEX, and requires either an up-to-date LATEX kernel (2016 onward), use of latexrelease with older kernels or loading `ltluatex.tex` for plain users.

The commands here are aimed mainly for use by package authors to develop environments needing specific catcode regimes. As such the package does not define any user level commands.

`\@setrangecatcode` `\@setrangecatcode{`⟨*start*⟩`}{`⟨*end*⟩`}{`⟨*catcode*⟩`}`

Sets all characters in the range ⟨*start*⟩–⟨*end*⟩ inclusive to have the ⟨*catcode*⟩ specified.

`\@pushcatcodetable` `\@pushcatcodetable`

`\@popcatcodetable` `\@popcatcodetable`

This pair of commands enable the current category code régime to be saved and restored meaning that arbitrary catcode changes can be made. This functionality will normally be used in concert with applying catcode tables. For example

```
\catcode`\Z=4 %
\@pushcatcodetable
\catcodetable\catcodetable@latex
% Code here
\@popcatcodetable
\showthe\catcode`\Z
```

will ensure that the 'content' is set with normal category codes but allow restoration of the non-standard codes at the conclusion. Importantly, it does not require that anything is known about the catcode situation in advance (*cf.* a more traditional approach to saving the state of targeting characters).

2 Implementation

```
⟨*package⟩
\edef\ctstackatcatcode{\the\catcode`\@}
\catcode`\@=11
```

Check for lltluatex functionality using `\newluafunction` as a marker.

```
\ifx\newluafunction\@undefined
  \input{ltluatex}%
\fi
```

`\@setcatcodetable` Save a catcode table specified in `#1` using the catcode settings specified in `#2`. These settings are executed in a local group to avoid affecting surrounding code. (Saving a catcode table is always a global operation.)

```
1
\def\@setcatcodetable#1#2{%
  \begingroup
    #2%
    \savecatcodetable#1%
  \endgroup
}
```

`\@setrangecatcode` Set a range of characters from `#1` to `#2` inclusive to the catcode specified in `#3`.

```
\def\@setrangecatcode#1#2#3{%
  \ifnum#1>#2 %
    \expandafter\@gobble
  \else
    \expandafter\@firstofone
  \fi
    {%
      \catcode#1=#3 %
      \expandafter\@setrangecatcode\expandafter
        {\number\numexpr#1+1\relax}{#2}{#3}%
    }%
}
```

`\@catcodetablelist` `\@catcodetablestack` Data structures for a stack: a list of free tables in the stack and the stack record itself.

```
\def\@catcodetablelist{}
\def\@catcodetablestack{}
```

`\@catcodetablestackcnt` A count for adding to the list of scratch tables.

```
\newcount\@catcodetablestackcnt
```

`\@pushcatcodetable` `\@pushctbl`

To push a table, first check there is a free one in the pool and if not create one. Then take the top table in the pool and use it to save the current table.

```
\def\@pushcatcodetable{%
  \ifx\@catcodetablelist\empty
    \global\advance\@catcodetablestackcnt by\@ne
    \edef\@tempa{\csname ct@\the\@catcodetablestackcnt\endcsname}%
    \expandafter\newcatcodetable\@tempa
    \xdef\@catcodetablelist{\@tempa}%
  \fi
  \expandafter\@pushctbl\@catcodetablelist\@nil
}
\def\@pushctbl#1#2\@nil{%
  \gdef\@catcodetablelist{#2}%
  \xdef\@catcodetablestack{#1\@catcodetablestack}%
  \savecatcodetable#1%
}
```

`\@popcatcodetable` `\@popctbl`

Much the same in reverse.

```
\def\@popcatcodetable{%
  \if!\@catcodetablestack!%
    \errmessage{Attempt to pop empty catcodetable stack}%
  \else
    \expandafter\@popctbl\@catcodetablestack\@nil
  \fi
}
\def\@popctbl#1#2\@nil{%
  \gdef\@catcodetablestack{#2}%
  \xdef\@catcodetablelist{\@catcodetablelist#1}%
  \catcodetable#1%
}

\catcode`\@\ctstackatcatcode\relax

⟨/package⟩
```