catcodes

—

“Generic” Switching of Category Codes

Uwe Lück[*]

September 20, 2012

Abstract

The catcodes bundle provides small packages for switching category codes, usable both with LaTeX and without. (i) stacklet.sty maintains stacks for “private letters,” needed for plainpkg.tex’s minimal framework for “generic” packages. (ii) actcodes.sty deals with “active characters,” switching their category codes and assigning meanings to “active-character tokens.” (iii) catchdq.sty uses the “double quote” as an active character for simplified access to typographical double quotes.—These packages are “generic” in the sense that they should be usable at least both with LaTeX and Plain TeX, based on plainpkg.tex.

Required Packages: plainpkg, stacklet

Related Packages: catoptions, pcatcode from amsrefs, texapi, csquotes.

Keywords: Macro programming, category codes, private letters, active characters, double quotes

Contents

1 Shared Features of Usage 2

2 stacklet.sty—Private Letters 3

2.1 Package File Header—plainpkg and Legalese 3

2.2 Usage 3

2.2.1 Installing and Calling 3

2.2.2 Commands and Syntax 3

2.3 The Code 4

2.3.1 Name Space 4

2.3.2 Pushing 4

2.3.3 Popping 5

[*]`http://contact-ednotes.sty.de.vu`

2.3.4 No `@` Stack with LaTeX 5
2.3.5 Leaving the Package File 5
2.3.6 VERSION HISTORY 5

3 actcodes.sty—Active Characters **6**
3.1 Package File Header—plainpkg and Legalese 6
3.2 Purpose and Usage 6
3.2.1 Installing and Calling 6
3.2.2 Commands and Syntax 6
3.3 The Code 7
3.3.1 Our Private Letters 7
3.3.2 The Core 7
3.3.3 `\def` and `\let` 7
3.3.4 Switching Back 8
3.3.5 Leaving and Version History 8

4 catchdq.sty—Proper Double Quotes by Toggling **9**
4.1 Package File Header—plainpkg and Legalese 9
4.2 Purpose and Usage 9
4.2.1 Installing and Calling 9
4.2.2 Commands and Syntax 9
4.3 The Code 10
4.3.1 Required 10
4.3.2 The Core: `\catchdq` 10
4.3.3 What Double Quotes Actually Are 10
4.3.4 Switching 11
4.3.5 Leaving and Version History 11

1 Shared Features of Usage

All the packages of the bundle are "plainpkg packages" in the sense of the plainpkg[1] documentation that exhibits details of what is summarized here. Therefore:

- All of them require that TeX finds `plainpkg.tex` as well as `stackrel.sty`.
- In order to load ⟨*catcodes*⟩`.sty` (where ⟨*catcodes*⟩ is `stacklet`, `actcodes`, or `catchdq`), type $\boxed{\texttt{\textbackslash usepackage\{}\langle\textit{catcodes}\rangle\texttt{\}}}$ within a LaTeX document preamble, $\boxed{\texttt{\textbackslash RequirePackage\{}\langle\textit{catcodes}\rangle\texttt{\}}}$ in a "plainpkg package", or $\boxed{\texttt{\textbackslash input}\textvisiblespace\langle\textit{catcodes}\rangle\texttt{.sty}}$ ... or perhaps `\input{`⟨*catcodes*⟩`.sty}`?

[1] `ctan.org/pkg/plainpkg`

2 stacklet.sty—Private Letters

See Section 2.2.2 for the commands provided.

2.1 Package File Header—plainpkg and Legalese

```
                                                 \input plainpkg
\ProvidesPackage{stacklet}[2012/08/27 v0.3 private letters (UL)]
%%
%% Copyright (C) 2012 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% There is NO WARRANTY (actually somewhat experimental).
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

2.2 Usage

2.2.1 Installing and Calling

The file `stacklet.sty` is provided ready, installation only requires putting it somewhere where TeX finds it (which may need updating the filename data base).[2] However, the file `plainpkg.tex` must be installed likewise.

With LaTeX, the file should be loaded by `\RequirePackage{stacklet}` or `\usepackage{stacklet}`.

Without LaTeX, both `\input␣stacklet.sty` and `\input␣plainpkg` load `stacklet.sty`.

2.2.2 Commands and Syntax

stacklet.sty provides

$\boxed{\texttt{\textbackslash PushCatMakeLetter\textbackslash}\langle char\rangle}$ *and* $\boxed{\texttt{\textbackslash PopLetterCat\textbackslash}\langle char\rangle}$

for getting "private letters" and giving them back their previous category code in package files with or without LaTeX. As LaTeX has its own stack for `@`, there are also

$\boxed{\texttt{\textbackslash PushCatMakeLetterAt}}$ *and* $\boxed{\texttt{\textbackslash PopLetterCatAt}}$

that care for `@`'s category code *without* LaTeX only.

[2] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

2.3 The Code

2.3.1 Name Space

Each "private letter" ⟨*char*⟩ gets its own stack, in some name space, determined by $\boxed{\texttt{\\cat\_stack}}$ (`\withcsname` is from plainpkg.tex):

```
\withcsname\xdef cat_stack\endcsname{%
    \noexpand\string \withcsname\noexpand cat_stack\endcsname
        \noexpand\string}
```

I.e., ?`cat_stack` will expand to

?`string` ?`cat_stack` ?`string`

in the notation of the dowith package.

```
% \withcsname\show cat_stack\endcsname
```

2.3.2 Pushing

$\boxed{\texttt{\\PushCatMakeLetter\\}\langle\textit{char}\rangle}$ ...

```
\xdef\PushCatMakeLetter#1{%
  \noexpand\withcsname
    \withcsname\noexpand pushcat_makeletter\endcsname
      \withcsname\noexpand cat_stack\endcsname#1\endcsname#1}
% \show\PushCatMakeLetter
\withcsname\gdef pushcat_makeletter\endcsname#1#2{%
```

`#1` is the stack token, `#2` is the "quoted" character. Pushing ...

```
    \xdef#1{\the\catcode`#2\relax%
```

... the new entry. `\relax` separates entries, braces instead tend to get lost in popping ... If the stack has existed before, its previous content is appended:

```
            \ifx#1\relax \else #1\fi}%
```

I thought of storing `\catcode`s hexadecimally (without braces) using LaTeX's `\hexnumber`, but the latter has so many tokens ... Finally rendering ⟨*char*⟩ a "letter":

```
    \catcode`#211\relax}
```

Now we can use a "private letter stack" for our own package:

```
\PushCatMakeLetter\_
```

2.3.3 Popping

`\PopLetterCat\⟨char⟩` passes \\⟨*char*⟩, the corresponding stack token, and the latter's expansion to `\popcat_`. `\end` serves as argument delimiter for the end of the stack only:

```
\gdef\PopLetterCat#1{%
  \expandafter\expandafter\expandafter
    \popcat_\csname\cat_stack#1\expandafter\endcsname
      \expandafter \end \csname\cat_stack#1\endcsname#1}
```

`\popcat_` parses the expansion, assigns the old category code and and stores the reduced stack:

```
\gdef\popcat_#1\relax#2\end#3#4{\catcode`#4#1\gdef#3{#2}}
```

. . . check existence? TODO

2.3.4 No @ Stack with LaTeX

`\PushCatMakeLetterAt` is like `\PushCatMakeLetter\@` except that it has no effect under LaTeX:

```
\gdef\PushCatMakeLetterAt{\ifltx\else\PushCatMakeLetter\@\fi}
```

`\PopLetterCatAt` by analogy . . .

```
\gdef\PopLetterCatAt{\ifltx\else\PopLetterCat\@\fi}
```

2.3.5 Leaving the Package File

. . . in our new way:

```
\PopLetterCat\_
\endinput
```

2.3.6 VERSION HISTORY

```
v0.1   2012/08/24   started
       2012/08/25   completed
       2012/08/26   extending doc.; \def\withcsname removed
v0.2   2012/08/26   \with_catstack containing \endcsname and with
                    three popping macros replaced by \csname
                    content \cat_stack, cf. memory.tex;
                    restructured
       2012/08/27   \PushCatMakeLetterAt fixed
v0.3   2012/08/27   def.s global

```

3 actcodes.sty—Active Characters

See Section 3.2.2 for the commands provided.

3.1 Package File Header—plainpkg and Legalese

```
                                                  \input plainpkg
\ProvidesPackage{actcodes}[2012/09/19 v0.2 active characters (UL)]
%%
%% Copyright (C) 2012 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% There is NO WARRANTY (actually somewhat experimental).
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

3.2 Purpose and Usage

The package derives from switching category codes in the nicetext and morehype bundles and should improve them.

3.2.1 Installing and Calling

The file `actcodes.sty` is provided ready, installation only requires putting it somewhere where TeX finds it (which may need updating the filename data base).[3] However, the files `plainpkg.tex` and `stacklet.sty` must be installed likewise.

With LaTeX, the file should be loaded by `\RequirePackage{actcodes}` or `\usepackage{actcodes}`.

Without LaTeX, load it by `\input␣actcodes.sty`.

As explained in `plainpgk-doc.pdf`, however, “generic” packages based on plainpkg should load actcodes by `\RequirePackage{actcodes}`.

3.2.2 Commands and Syntax

actcodes.sty provides `\MakeActive`, `\MakeActiveAss`, `\MakeActiveDef`, `\MakeActiveLet`, `\MakeOther`, `\MakeActiveOther` with rather obvious syntax—you find more detailed descriptions mixed with implementation below

[3] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

... TODO —Without LaTeX, the latter's internal $\boxed{\texttt{\textbackslash @gobble}\langle arg\rangle}$ is provided as well.

3.3 The Code

3.3.1 Our Private Letters

```
\PushCatMakeLetterAt
```

3.3.2 The Core

$\boxed{\texttt{\textbackslash MakeActiveAss}\langle \textit{ass-fun}\rangle\texttt{\textbackslash}\langle \textit{char}\rangle\langle \textit{ass-args}\rangle}$ "activates" ⟨*char*⟩ and then applies assignment function ⟨*ass-fun*⟩ with arguments ⟨*ass-args*⟩ to it. The code derives from LaTeX's `\@sverb` and `\do@noligs` and was also discussed on the LATEX-L mailing list September 2010 (Will Robertson; Heiko Oberdiek). The present definition generalizes `\MakeActiveDef` and `\MakeActiveLet` of my earlier packages.

```
\gdef\MakeActiveAss#1#2{%
  \MakeActive#2%
  \begingroup \lccode`\~`#2\relax \lowercase{\endgroup #1~}}
```

I was reluctant to provide $\boxed{\texttt{\textbackslash MakeActive\textbackslash}\langle \textit{char}\rangle}$, but with catchdq.sty, it would be better ...

```
\gdef\MakeActive#1{\catcode`#1\active}
```

... it just "revives" the meaning that the corresponding active-character token last time ...

3.3.3 `\def` and `\let`

$\boxed{\texttt{\textbackslash MakeActiveDef\textbackslash}\langle \textit{char}\rangle\langle \textit{parameters}\rangle\texttt{\{}\langle \textit{replace}\rangle\texttt{\}}}$ has been employed in fifinddo and blog (which is based on fifinddo) so far.

```
\gdef\MakeActiveDef{\MakeActiveAss\def}
```

W.r.t. the definition of this `\MakeActiveDef`, Heiko Oberdiek remarked that it allows *macro parameters*, as opposed to my earlier definition in fifinddo. Without parameters, this kind of macro has been used for conversion of text encodings (atari.fdf, and I thought this was the idea of stringenc ...).

$\boxed{\texttt{\textbackslash MakeActiveLet\textbackslash}\langle \textit{char}\rangle\langle \textit{cmd}\rangle}$ has been provided in niceverb so far. The present package has been made in order to have `\MakeActiveLet` with blog.sty as well, it was too annoying to use `\MakeActiveDef` there so often.

```
\gdef\MakeActiveLet{\MakeActiveAss\let}
```

3.3.4 Switching Back ...

Sometimes, the “active” behaviour of ⟨*char*⟩ is too difficult, and you may want to switch bach to its “simple” way ... This may work by $\boxed{\texttt{\textbackslash MakeOther\textbackslash}\langle\textit{char}\rangle}$ ... with LaTeX, `\MakeOther` just is `\@makeother` ...

```
\ifltx \global\let\MakeOther\@makeother
\else  \gdef\MakeOther#1{\catcode`#112\relax}
\fi
```

But within a macro (or other) argument, you can’t change the `\catcode`. (I lost some time by not realizing that it was within a large argument where I tried to switch the `\catcode`.) Anyway or in certain cases, it may be better to keep a character “active” throughout a document and just to change the *expansion* of the “active-character token.” This can be done with `\MakeActiveLet` and `\MakeActiveDef` in certain cases already. E.g., when the *“blank space”* has been “activated” by `\obeylines`, `\MakeActiveLet\␣\space` “undoes” this half-way, while it does not restore “argument skipping” and “compressing blank spaces.”

When character ⟨*char*⟩ should be “active” for some time, but for certain moments you prefer that it behaves like an “other character”, you can switch to its “other” expansion by $\boxed{\texttt{\textbackslash MakeActiveOther\textbackslash}\langle\textit{char}\rangle}$:

```
\gdef\MakeActiveOther#1{%
    \MakeActiveAss\edef#1{\expandafter\@gobble\string#1}}
```

`\MakeActiveOther` uses LaTeX’s $\boxed{\texttt{\textbackslash @gobble}\langle\textit{arg}\rangle}$, *without* LaTeX, actcodes provides it:

```
\ifltx\else \long\gdef\@gobble#1{} \fi
% \show_ \MakeActiveOther\_ \show_ \expandafter\show_
```

I am *not* providing a version *without* the `\catcode` change, although the latter is superfluous here TODO ...

`niceverb` also provides `\MakeNormal\`⟨*char*⟩, it may migrate to here in the future, and there may be $\boxed{\texttt{\textbackslash MakeActiveNormal\textbackslash}\langle\textit{char}\rangle}$ extending the above `\MakeActiveOther` TODO ...

Also, a *stack* might be used as in stacklet, even to switch *meanings* of active-character tokens ... not sure TODO ...

babel does similar things, but I never have ... TODO

3.3.5 Leaving and Version History

```
\PopLetterCatAt
\endinput
```

VERSION HISTORY

```
v0.1   2012/08/26   started, almost completed
       2012/08/27   completed; realizing \Push...At ..., bug fixes
v0.2   2012/08/28   \global\let, \def -> \gdef
       2012/09/16   \MakeActive
       2012/09/19   doc: stacklet

```

4 catchdq.sty—Proper Double Quotes by Toggling

See Section 4.2.2 for the commands provided. Note that the csquotes provides more comprehensive functionality.

4.1 Package File Header—plainpkg and Legalese

```
                                                \input plainpkg
\ProvidesPackage{catchdq}[2012/09/20 v0.2 simple typographic dqs (UL)]
%%
%% Copyright (C) 2012 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% There is NO WARRANTY (actually somewhat experimental).
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

4.2 Purpose and Usage

4.2.1 Installing and Calling

The file `catchdq.sty` is provided ready, installation only requires putting it somewhere where TEX finds it (which may need updating the filename data base).[4] However, the files `plainpkg.tex` and `stacklet.sty` must be installed likewise.

With LATEX, the file should be loaded by `\RequirePackage{catchdq}` or `\usepackage{catchdq}`.

Without LATEX, load it by `\input␣catchdq.sty`.

As explained in `plainpgk-doc.pdf`, however, "generic" packages based on plainpkg should load catchdq by `\RequirePackage{catchdq}`.

4.2.2 Commands and Syntax

catchdq.sty (indirectly) allows using $\boxed{\texttt{"}\langle \textit{no-dqs}\rangle\texttt{"}}$ for surrounding ⟨*no-dqs*⟩ with typographical quotation marks, using that double quote $\boxed{\texttt{"}}$ as an active character. As rendering that `"` active during defining macros can corrupt the latter, the user (or package writer) must activate that `"` explicitly by $\boxed{\texttt{\textbackslash catchdqs}}$.

[4] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

Further difficulties may arise after `\catchdqs`, various ways to get around them are described in the remaing sections.

4.3 The Code

4.3.1 Required

The package is an application (of ideas of) actcodes.sty:

```
\RequirePackage{actcodes}
```

4.3.2 The Core: `\catchdq`

`\catchdq⟨no-dqs⟩"` will expand to `\dqtd{⟨no-dqs⟩}`, provided the ASCII double quote is an active character:

```
{\MakeActive\"\gdef\catchdq#1"{\dqtd{#1}}}
```

4.3.3 What Double Quotes Actually Are

`\dqtd` in turn is a kind of "variable." blog.sty offered `\endqtd` for English typographical double quotes, `\dedqtd` for German ones, and `\asciidqtd` for "non-typographical" double quotes (as needed for XML attributes). `\asciidq` accesses a single ASCII double quote, `\enldq` a single English typographical left one, `\enrdq` a single English typographical right one. (It may be useful to access them indepentently of each other, in certain complex situations ...) blog.sty, dealing with HTML, of course has different ideas about them TODO.

```
\gdef\asciidq{"}
\gdef\asciidqtd#1{"#1"}
```

We allow loading catchdq *after* another package (such as blog.sty) has chosen meanings for `\endqtd` and the like (difficult TODO)

```
\ifx\enldq \undefined \gdef\enldq{``}                 \fi
\ifx\enrdq \undefined \global\let\enrdq\asciidq      \fi
\ifx\endqtd\undefined \gdef\endqtd#1{\enldq#1\enrdq} \fi
```

Typographical alternatives to `\endqtd` may be obtained from ngerman.sty or so, if you are smart ... (see Section 4.3.4 for how it works):

```
\ifx\dedqtd\undefined \gdef\dedqtd#1{\glqq#1\grqq}   \fi
```

blog.sty, dealing with HTML, had a different idea about `\endqtd` of course. It has also used the mechanism of the langcode package that allows using `\dqtd` and other language-depended constructs with an "implicit" choice according to the "current language code," which should appear soon.

4.3.4 Switching

blog.sty usually does a single switch which gets a new name now: `\catchdqs`.

```
\gdef\catchdqs{\MakeActiveLet\"\catchdq}
```

After this, `"`⟨*no-dqs*⟩`"` will expand to `\dqtd{#1}`. The default expansion for `\dqtd` will be `\endqtd`:

```
\ifx\dqtd\undefined \global\let\dqtd\endqtd \fi
```

Might be done by `\endqs`—when there are alternatives, but blog.sty and langcode.sty do this in a different way … TODO

```
% \gdef\endqs{\let\dqtd\endqtd}
% \ifx\dqtd\undefined \global\endqs \fi
```

Actually, here is a little “Tessst” … and here with „doytshe doppleta anf…“ … This has been achieved by

```
\usepackage{ngerman}␣\originalTeX
```

`\MakeOther\"` may switch off catching mode (—done just before, as niceverb at present doesn't render it verbatim). actcodes suggests a different way to return from the `\catchdqs` state: Let the character active and change its meaning only, let it *expand* to its “other” version—by `\activeasciidqs`? `\MakeActiveOther\"` and `\let"\asciidq` (it works!) or `\MakeActiveLet\"\asciidq` (abbreviate as `\activeasciidqs`?) … In blog.sty, there never was a need for switching back. We must rework interaction with niceverb and can perhaps simplify the latter, … TODO

4.3.5 Leaving and Version History

```
\endinput
```

VERSION HISTORY

```
v0.1   2010/11/13   in texblog.fdf
v0.2   2012/09/17   own file, new ideas ...
       2012/09/19   doc: stacklet
       2012/09/20   \dedqtd conditionally; reworked doc.,
                    tested ngerman.sty

```