1 Introduction

The well known etoolbox package provides a bunch of functions for patching existing commands; in particular \patchcmd, \pretocmd and \apptocmd that do a wonderful job, but suffer from a limitation: if some package has defined

\newcommand{\xyz}[1][x]{-#1!}

where \xyz has an optional argument, then \patchcmd and siblings cannot be used to modify the workings of \xyz. The same happens when a command has been defined with \DeclareRobustCommand.

The reason for this is \TeXnical or, better, \LaTeXnical. When \LaTeX performs the above definition, the expansion of \xyz will be

\@protected@testopt \xyz \xyz \xyz {x}

where \@protected@testopt is a macro that essentially checks whether we are in a “protected” context, so that expansion should not be performed all the way (in moving arguments or write operations), or not; in the former case it issues a protected version of \xyz, while in the latter case it expands the macro \xyz that is a single command (yes, with a backslash in its name) which contains the real definition; a way to access this definition is to issue the command

\expandafter\show\csname\string\xyz\endcsname

which will print in the log file the message

> \xyz=\long macro: [#1]->-#1!

As usual, after -> we see the definition. In order to use \patchcmd to change the exclamation mark into a hyphen one must do

\expandafter\patchcmd\csname\string\xyz\endcsname{!}{-}{1}{1}

(see the documentation of etoolbox for details about the arguments).

A similar situation happens if \xyz has been defined by

---

* This file describes version 0.3a, last revised 2020/03/25.
† E-mail: Enrico DOT Gregorio AT univr DOT it
\DeclareRobustCommand{\xyz}{something}

A \show\xyz would show the cryptic

> \xyz=macro:
   ->\protect \xyz.

and only a close look reveals the clever trick used by the \LaTeX{} team: the \protect is not applied to \xyz, but to the macro \xyz which has a space at the end of its name! And this macro is the one that contains the real definition. Indeed,

\expandafter\show\csname xyz\space\endcsname

produces the message

> \xyz =\long macro:
   ->something.

In this case, in order to apply \patchcmd we must say

\expandafter\patchcmd\csname\string\xyz\space\endcsname{s}{S}{}{}

If the macro with \DeclareRobustCommand is defined to have an optional argument, say

\DeclareRobustCommand{\xyz}[1][x]{-#1!}

one has to combine the two tricks:

\expandafter\patchcmd\csname\string\xyz\space\endcsname{!}{-}{}{}

It’s hard and error prone to remember all these tricks, so this package comes to the rescue.

**Caveat**

This package is still in a preliminary version, but no relevant changes to the interface should be introduced in later versions. A different and more powerful implementation is under testing, see the package *regexpatch*.

## 2 Commands

The commands introduced by this package are

- \xpatchcmd
- \xpretocmd
- \xapptocmd

which have the same syntax as the similar commands provided by *etoolbox* and apply to all kind of commands defined by

- the \LaTeX{} kernel macros \newcommand, \renewcommand, \providecommand, but also \newenvironment and \renewenvironment;
- the \LaTeX{} kernel macro for defining robust commands \DeclareRobustCommand;
• the `etoolbox` macros \texttt{\newrobustcmd}, \texttt{\renewrobustcmd}, \texttt{\providerobustcmd}.

Notice that patching the definition of the environment `foo` requires patching `\foo` or `\endfoo`.

These commands will act as the original ones if the macro to patch is not robust or with optional arguments.

Moreover the package defines

• \texttt{\xpatchbibmacro}
• \texttt{\xpretobibmacro}
• \texttt{\apptobibmacro}

that can be used to patch commands defined with bibliography’s \texttt{\newbibmacro}. Say that we have

\begin{verbatim}
\newbibmacro{foo.bar}{[2]{#1 and #2}
\end{verbatim}

Then, to change `and` into `und`, we can now say

\begin{verbatim}
\xpatchbibmacro{foo.bar}{(and)}{(und)}{}
\end{verbatim}

Patching these macros requires resorting to the very cryptic
\begin{verbatim}
\expandafter\patchcmd\csname abx@macro@\detokenize{foo.bar}\endcsname{and}{und}{}{}
\end{verbatim}

that would become an astonishing
\begin{verbatim}
\expandafter\patchcmd\csname\expandafter\string\csname abx@macro@\detokenize{foo.bar}\endcsname\endcsname{and}{und}{}{}
\end{verbatim}

if the original definition had been with an optional argument, say

\begin{verbatim}
\newbibmacro{foo.bar}{[2][x]{#1 and #2}
\end{verbatim}

For bibliography users there are also

• \texttt{\xpatchbibdriver}
• \texttt{\xpretobibdriver}
• \texttt{\apptobibdriver}

for patching commands defined with \texttt{\DeclareBibliographyDriver}. One could use, for patching the driver `foo`,

\begin{verbatim}
\makeatletter
\patchcmd\blx@bbx@foo\{X\}\{Y\}\{<success>\}\{<failure>\}
\pretocmd\blx@bbx@foo\{P\}
\apptocmd\blx@bbx@foo\{A\}
\makeatother
\end{verbatim}

but having a lighter interface can be handy. Since our macros use \texttt{\pretocmd} and \texttt{\apptocmd} for consistency, remember to always use the \texttt{\{success\}} and \texttt{\{failure\}} arguments also with \texttt{\xpretobibdriver} and \texttt{\apptobibdriver}.

Under the same philosophy, one can use the macros
for the \texttt{biblatex} internal macro defined respectively with
\DeclareshapeFieldFormat, \DeclareshapeNameFormat, \DeclareshapeListFormat, \DeclareshapeIndexFieldFormat, \DeclareshapeIndexNameFormat, \DeclareshapeIndexListFormat.

All the eighteen \texttt{x...format} commands take a first optional argument, with default value $*$, see later on.

Finally, the package defines the commands
\texttt{xshowcmd}
\texttt{xshowbibmacro}
\texttt{xshowbibdriver}
\texttt{xshowfieldformat}
\texttt{xshownameformat}
\texttt{xshowlistformat}
\texttt{xshowindexfieldformat}
\texttt{xshowindexnameformat}
\texttt{xshowindexlistformat}

that are the analog of \texttt{show} to see the “real” definition of a macro, be it defined with optional arguments or as a robust command; the \texttt{xshowbib...} and \texttt{xshow...format} ones are for the corresponding \texttt{biblatex} macros. The last six have an optional first argument (default value $*$).
3 Using the original commands

The original \texttt{\textbackslash patchcmd} has still its use: suppose you want to modify the default for the optional argument passed to a macro: if the original definition is

\texttt{\newcommand\texttt{\textbackslash xyz}{1}[x]{-#1!}}

then one can say

\texttt{\textbackslash patchcmd\texttt{\textbackslash xyz}{\{x\}}\{\{y\}\}{\{\}\}}

because of the way \texttt{\textbackslash xyz} is defined, as shown before.

4 Syntax

\texttt{\textbackslash xpatchcmd}{\langle command\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretocmd}{\langle command\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xapptocmd}{\langle command\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchbibmacro}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretoibibmacro}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xapptoibibmacro}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchfieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretofieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptofieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchnameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretonameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptonameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchlistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretolistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptolistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchindexfieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretoindexfieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptoindexfieldformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchindexnameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretoindexnameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptoindexnameformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpatchindextlistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle search\rangle}{\langle replace\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xpretoindextlistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle prepend\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xaptoindextlistformat}{\langle entrytype\rangle}{\langle name\rangle}{\langle append\rangle}{\langle success\rangle}{\langle failure\rangle}
\texttt{\textbackslash xshowcmd}{\langle command\rangle}
\texttt{\textbackslash xshowbibname}{\langle name\rangle}
\texttt{\textbackslash xshowbibdriver}{\langle name\rangle}
\texttt{\textbackslash xshowfieldformat}{\langle entrytype\rangle}{\langle name\rangle}
Here \emph{\langle\text{name}\rangle} is the command’s name (with the backslash), while \emph{\langle\text{name}\rangle} is the string that appears as the argument to \verb!\newbibmacro!, \verb!\DeclareBibliographyDriver!, \verb!\DeclareFieldFormat!, \verb!\DeclareNameFormat!, \verb!\DeclareListFormat!, \verb!\DeclareIndexFieldFormat!, \verb!\DeclareIndexNameFormat! or \verb!\DeclareIndexListFormat! respectively; \emph{\langle\text{search}\rangle}, \emph{\langle\text{replace}\rangle}, \emph{\langle\text{prepend}\rangle} and \emph{\langle\text{append}\rangle} are the list of tokens that are to be used for the specific tasks; \emph{\langle\text{success}\rangle} and \emph{\langle\text{failure}\rangle} are token lists to be executed if the patching succeeds or fails respectively. I find it useful to use \verb!\ddt! as \emph{\langle\text{failure}\rangle}, so that \LaTeX{} will stop for the undefined control sequence when the patching fails.

All the \verb!\x...ormat! macros have an optional argument that by default is \emph{\langle\text{*}\rangle}. It’s important to remember that patching commands that have \texttt{@}-commands in their name or replacement text must always be performed between \verb!\makeatletter! and \verb!\makeatother!.

## 5 Limitations and warnings

Macros defined in devious ways might trick \verb!\xpatchcmd! and siblings, although many precautions have been taken in order this not to happen. Always check with care.

Remember that one must never use the old trick
\begin{verbatim}
\let\ORIxyx\xyz
\renewcommand{\xyz}{[1][x]{\ORIxyz[#1]?}}
\end{verbatim}

if \verb!\xyz! had been defined with an optional argument. For such things it’s better to use \verb!\xpatchcmd! and friends or employ the \verb!letltxmacro! package by H. Oberdiek, that provides \verb!\LetLtxMacro! for purposes like this one.

Although this package has been written with the experimental \LaTeX{}3 macros, the commands can’t be used to patch commands defined with the \verb!xparse! interface, in general.

If a command appears to have one optional argument at the user level, this doesn’t mean it has been defined with \verb!\newcommand! directly. One should always check the definitions with \verb!\show! and \verb!\xshowcmd! before trying a patch: of course one has to know what a command does, in order to patch it. And, when first testing the patch, it’s best to set \verb!\tracingpatches!.

## 6 History

Version 0.1 First public release.

Version 0.2 Added \verb!\x...ibdriver! macros; fixed a bug for control symbols defined with \verb!\newcommand! and an optional argument.

Version 0.3 Added \verb!\x...ormat! macros (by kind request of the \verb!biblatex! maintainers).
7 The implementation of \texttt{xpatch}

\begin{verbatim}
\ProvidesExplPackage{\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}

A check to make sure that expl3 is not too old
\@ifpackagelater { expl3 } { 2011/10/09 }
{
\PackageError { xpatch } { Support-package-l3kernel-too-old. }
{ Please-install-an-up-to-date-version-of-l3kernel-
using-your-TeX-package-manager-or-from-CTAN.\ \ \ \ \ \Loading-xpatch-will-abort! }
\tex_endinput:D
}
\end{verbatim}

The \texttt{xparse} and \texttt{etoolbox} packages are required.
\begin{verbatim}
\RequirePackage{xparse,etoolbox}
\end{verbatim}

7.1 Utilities, variables and constants

Generate a variant of \texttt{\tl_if_in:NnT} to get the expanded second argument.
\begin{verbatim}
\cs_generate_variant:Nn \tl_if_in:NnT { Nx }
\end{verbatim}

A boolean for the testing of robust commands.
\begin{verbatim}
\bool_new:N \l__xpatch_protect_bool
\end{verbatim}

The constant \texttt{\c_backslash_str} is defined in \texttt{l3str} that's not loaded at the moment, so we save a bit of memory not loading it.
\begin{verbatim}
\cs_if_exist:NF \c_backslash_str
{ \tl_const:Nx \c_backslash_str { \cs_to_str:N \texttt{\char92} } }
\end{verbatim}

A “bizarre” token list that’s quite improbable to find in the replacement text of a macro.
\begin{verbatim}
\tl_const:Nx \c__xpatch_bizarre_tl
{ \tl_to_str:n { **)-(**/**[-[** } }
\end{verbatim}

Internal token lists for storing the various parts of the command to be patched.
\begin{verbatim}
\tl_new:N \l__xpatch_name_tl
\tl_new:N \l__xpatch_repl_tl
\end{verbatim}

7.2 The main functions

The main function takes as first argument one of \texttt{\patchcmd, \pretocmd or \apptocmd}; the second argument is the command we want to patch.

Some technical remarks. Suppose we have the following definitions:
\begin{verbatim}
\DeclareRobustCommand{\xa}[1]{\xa (DeclareRobustCommand-noopt)}
\DeclareRobustCommand{\xb}[1][x]{\xb (DeclareRobustCommand-opt)}
\newcommand{\xc}[1][]\{\xc (newcommand-opt)}
\newrobustcmd{\xad}[1][]\{\xad (newrobustcmd-opt)}
\DeclareRobustCommand{\1}[1][1\ 1}{\1 (DeclareRobustCommand-noopt)}
\DeclareRobustCommand{\2}[1][2\ 2\ 2]{\2 (DeclareRobustCommand-opt)}
\newcommand{\3}[1][]\{\3 (newcommand-opt)}
\newrobustcmd{\4}[1][4\ 4\ 4]{\4 (newrobustcmd-opt)}
\end{verbatim}
Then the first level expansions are, respectively,

\[
\begin{align*}
+\protect\texttt{\protect/xaa_{\texttt{\protect}}/xaa_{\texttt{\protect}}/xaa_{\texttt{\protect}}} + \\
+\protect\texttt{\protect/xab_{\texttt{\protect}}/xab_{\texttt{\protect}}/xab_{\texttt{\protect}}} + \\
+\@protected@testopt\texttt{\protect/xac_{\texttt{\protect}}/xac_{\texttt{\protect}}/xac_{\texttt{\protect}}} + \\
+\texttt{\protect/xac_{\texttt{\protect}}/xac_{\texttt{\protect}}/xac_{\texttt{\protect}}} + \\
+\x@protect\texttt{\protect/xad_{\texttt{\protect}}/xad_{\texttt{\protect}}/xad_{\texttt{\protect}}} + \\
+\texttt{\protect/xad_{\texttt{\protect}}/xad_{\texttt{\protect}}/xad_{\texttt{\protect}}} + \\
+\@protected@testopt\texttt{\protect/1/protest_{\texttt{\protect}}/1_{\texttt{\protect}}/1_{\texttt{\protect}}} + \\
+\texttt{\protect/1/protest_{\texttt{\protect}}/1_{\texttt{\protect}}/1_{\texttt{\protect}}} + \\
+\@protected@testopt\texttt{\protect/2/protest_{\texttt{\protect}}/2_{\texttt{\protect}}/2_{\texttt{\protect}}} + \\
+\texttt{\protect/2/protest_{\texttt{\protect}}/2_{\texttt{\protect}}/2_{\texttt{\protect}}} + \\
+\@protected@testopt\texttt{\protect/3/protest_{\texttt{\protect}}/3_{\texttt{\protect}}/3_{\texttt{\protect}}} + \\
+\texttt{\protect/3/protest_{\texttt{\protect}}/3_{\texttt{\protect}}/3_{\texttt{\protect}}} + \\
+\@testopt\texttt{\protect/4/protest_{\texttt{\protect}}/4_{\texttt{\protect}}/4_{\texttt{\protect}}} + \\
+\texttt{\protect/4/protest_{\texttt{\protect}}/4_{\texttt{\protect}}/4_{\texttt{\protect}}} + \\
\end{align*}
\]

where the + is used to delimit the expansions and show the spaces. Remember that \show always adds a space after a control word, but not after a control symbol such as \1. However, in lines 5 and 6, \1 is not a control symbol any more. So we have to take care of \protect, \x@protect, \@protected@testopt and \@testopt. But it’s not simply sufficient to check for the presence of such a token at the start of the replacement text, or we’ll be confused by macros such as \linebreak, whose replacement text starts with \@testopt. So we’ll check also for the presence of the subsequent tokens, that depend on the macro’s name. We add a perhaps useless “random” string at the beginning, as we’d like to ensure that the matches are exactly at the start of the replacement text.

\begin{verbatim}
\cs_new:Npn \xpatch_main:NN #1 #2
\{
We initialize the boolean to false.
\bool_set_false:N \l__xpatch_protect_bool

First of all we store the command-to-patch name.
\tl_set:Nx \l__xpatch_name_tl \cs_to_str:N #2

We store the replacement text of the command-to-patch, but adding the bizarre token list in front of it which consists of all category 12 characters, just to be sure that the matches are at the beginning.\footnote{This part will be reimplemented as soon as \l3regex stabilizes.}
\tl_set:Nx \l__xpatch_repl_tl \cs_replacement_spec:N #2

We look whether the token list contains the bizarre list followed by \protect and the same name (with two spaces) which happens if \2 is a control sequence defined by \DeclareRobustCommand, so we add a space to the command name.
\tl_if_in:NxT \l__xpatch_repl_tl \l__xpatch_bizarre_tl \cs_to_str:N \c_space_tl
\tl_put_right:Nn \l__xpatch_name_tl \cs_space_tl \c_space_tl

We look whether the token list contains the bizarre list followed by \x@protect which happens if \2 is a control symbol defined by \DeclareRobustCommand, so we add a space to the command name.
\tl_if_in:NxT \l__xpatch_repl_tl \l__xpatch_bizarre_tl
\tl_put_right:Nn \l__xpatch_name_tl \c_space_tl
\end{verbatim}
In both the preceding cases, we have to do another check, so we set a boolean to true.

We look whether the token list contains the bizarre list followed by \保护@testopt which happens if \#2 is a control word with an optional argument (from \newcommand).

\IfIn:NxT \__xpatch_repl_tl
\{ \c__xpatch_bizarre_tl \token_to_str:N \@protected@testopt \c_space_tl \c_backslash_str \l__xpatch_name_tl \c_backslash_str
\}
\{ \bool_set_true:N \l__xpatch_protect_bool
\tl_put_right:Nn \l__xpatch_name_tl \c_space_tl \c_backslash_str \c_backslash_str
\}
\IfIn:NxT \__xpatch_repl_tl
\{ \c__xpatch_bizarre_tl \token_to_str:N \@protected@testopt \c_space_tl \c_backslash_str \c_backslash_str \c_backslash_str \l__xpatch_name_tl \c_backslash_str
\}
\{ \tl_put_left:Nn \l__xpatch_name_tl \c_backslash_str \c_backslash_str
\}

We look whether the token list contains the bizarre list followed by \保护@testopt which happens if \#2 is a control symbol with an optional argument (from \newcommand).

\IfIn:NxT \__xpatch_repl_tl
\{ \c__xpatch_bizarre_tl \token_to_str:N \@protected@testopt \c_space_tl \c_backslash_str \c_backslash_str \l__xpatch_name_tl \c_backslash_str
\}
\{ \tl_put_left:Nn \l__xpatch_name_tl \c_backslash_str \c_backslash_str
\}

We look whether the token list contains the bizarre list followed by \testopt which happens if \#2 is a command with an optional argument (from \newrobustcmd).

\IfIn:NxT \__xpatch_repl_tl
\{ \c__xpatch_bizarre_tl \token_to_str:N \testopt \c_space_tl \c_backslash_str \c_backslash_str \l__xpatch_name_tl \c_backslash_str
\}
\{ \tl_put_left:Nn \l__xpatch_name_tl \c_backslash_str \c_backslash_str
\}

In both the preceding cases, we add a backslash in front of the command’s name.

If the command-to-patch was defined by \DeclareRobustCommand we have to do another test, namely checking whether it has an optional argument and, in this case, adding a backslash in front of the name. We replicate the test for \保护@testopt.

\IfIfNT \l__xpatch_protect_bool
\{ \c__xpatch_bizarre_tl \token_to_str:N \@protected@testopt \c_space_tl \c_backslash_str \l__xpatch_name_tl \c_backslash_str
\}

Finally, we pass the real command-to-patch name to the patching macro.
\begin{verbatim}
\exp_after:wN \cs:w \l__xpatch_name_tl \cs_end:
\end{verbatim}
That’s the last operation!

\section*{7.3 User level commands}

The user level commands.

\begin{verbatim}
\NewDocumentCommand{\xpatchcmd}{\O{*} m}{\xpatch_main:Nc \patchcmd \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\NewDocumentCommand{\xpretocmd}{\O{*} m}{\xpatch_main:Nc \pretocmd \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\NewDocumentCommand{\xapptocmd}{\O{*} m}{\xpatch_main:Nc \apptocmd \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\NewDocumentCommand{\xshowcmd}{\O{*} m}{\xpatch_main:Nc \show \tl_to_str:n {#1} #1}
\end{verbatim}

We generate a variant of \texttt{xpatch_main:N} to accept a macro’s name as its second argument.

\begin{verbatim}
\cs_generate_variant:Nn \xpatch_main:NN { Nc }
\end{verbatim}

Now we can define the patching macros for \texttt{newbibmacro} defined commands. In case one uses a wrong name, it will remain in the hash space, but it shouldn’t be a problem: \texttt{tracingpatches} must be used when testing, and it will warn about an undefined macro or one equivalent to \texttt{relax}.

\begin{verbatim}
\NewDocumentCommand{\xpatchbibmacro}{\O{*} m}{\xpatch_main:Nc \patchbibmacro \tl_to_str:n {#1} #1}
\NewDocumentCommand{\xpretobibmacro}{\O{*} m}{\xpatch_main:Nc \pretobibmacro \tl_to_str:n {#1} #1}
\NewDocumentCommand{\xapptobibmacro}{\O{*} m}{\xpatch_main:Nc \apptobibmacro \tl_to_str:n {#1} #1}
\NewDocumentCommand{\xshowbibmacro}{\O{*} m}{\xpatch_main:Nc \showbibmacro \tl_to_str:n {#1} #1}
\end{verbatim}

The macros for patching commands defined with \texttt{DeclareFieldFormat}; all that holds for the preceding commands is valid also for the following groups of similar commands.

\begin{verbatim}
\NewDocumentCommand{\xpatchfieldformat}{\O{*} m}{\xpatch_main:Nc \patchfieldformat \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\NewDocumentCommand{\xpretofieldformat}{\O{*} m}{\xpatch_main:Nc \pretofieldformat \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\NewDocumentCommand{\xapptofieldformat}{\O{*} m}{\xpatch_main:Nc \apptofieldformat \tl_to_str:n {#1} \tl_to_str:n {#2} #1}
\end{verbatim}
Finally, the patching macros for \texttt{biblatex} drivers that don’t need the overhead of \texttt{\xpatch_main:NN}. 

\begin{verbatim}
  \NewDocumentCommand{\xshowfieldformat}{O{*} m}{\xpatch_main:Nc \show{abx@ffd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpatchnameformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretonameformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptonameformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshownameformat}{O{*} m}{\xpatch_main:Nc \show{abx@ffd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}

\DeclareNameFormat:
  \NewDocumentCommand{\xpatchnameformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretonameformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptonameformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshownameformat}{O{*} m}{\xpatch_main:Nc \show{abx@ffd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}

\DeclareListFormat:
  \NewDocumentCommand{\xpatchlistformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@lfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretolistformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@lfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptolistformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@lfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshowlistformat}{O{*} m}{\xpatch_main:Nc \show{abx@lfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}

\DeclareIndexFieldFormat:
  \NewDocumentCommand{\xpatchindexfieldformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@fid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretoindexfieldformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@fid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptoindexfieldformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@fid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshowindexfieldformat}{O{*} m}{\xpatch_main:Nc \show{abx@fid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}

\DeclareIndexNameFormat:
  \NewDocumentCommand{\xpatchindexnameformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@nid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretoindexnameformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@nid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptoindexnameformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@nid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshowindexnameformat}{O{*} m}{\xpatch_main:Nc \show{abx@nid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}

\DeclareIndexListFormat:
  \NewDocumentCommand{\xpatchindexlistformat}{O{*} m}{\xpatch_main:Nc \patchcmd{abx@lid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xpretoindexlistformat}{O{*} m}{\xpatch_main:Nc \pretocmd{abx@lid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xapptoindexlistformat}{O{*} m}{\xpatch_main:Nc \apptocmd{abx@lid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
  \NewDocumentCommand{\xshowindexlistformat}{O{*} m}{\xpatch_main:Nc \show{abx@lid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2}}}
\end{verbatim}

11
Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

<table>
<thead>
<tr>
<th>Symbols</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>\apptocmd</td>
<td>99, 107, 115, 123, 131, 139, 147, 155, 163</td>
</tr>
<tr>
<td>B</td>
<td>\bool_if:NTF</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>\bool_new:N</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>\bool_set_false:N</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>\bool_set_true:N</td>
<td>36, 46</td>
</tr>
<tr>
<td>C</td>
<td>\cs:w</td>
<td>83, 95</td>
</tr>
<tr>
<td></td>
<td>\cs_end:</td>
<td>83, 95</td>
</tr>
<tr>
<td></td>
<td>\cs_generate_variant:Nn</td>
<td>15, 101</td>
</tr>
<tr>
<td></td>
<td>\cs_if_exist:NTF</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>\cs_new:Npn</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>\cs_replacement_spec:N</td>
<td>28, 82</td>
</tr>
<tr>
<td></td>
<td>\cs_to_str:N</td>
<td>18, 26</td>
</tr>
<tr>
<td>E</td>
<td>\exp_after:wN</td>
<td>82, 95</td>
</tr>
<tr>
<td></td>
<td>\exp_args:Nc</td>
<td>159, 161, 163, 165</td>
</tr>
<tr>
<td></td>
<td>\ExplFileDate</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>\ExplFileDescription</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>\ExplFileName</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>\ExplFileVersion</td>
<td>2</td>
</tr>
</tbody>
</table>

| P | \PackageError | 6 |
| | \patchcmd | 97, 103, 111, 119, 127, 135, 143, 151, 159 |
| | \pretocmd | 98, 105, 113, 121, 129, 137, 145, 153, 161 |
| | \protect | 32 |
| | \ProvidesExplPackage | 1 |
| R | \RequirePackage | 14 |
| S | \show | 100, 109, 117, 125, 133, 141, 149, 157, 165 |
| | \str commands: | 17, 18, 33, 43, 53, 54, 57, 63, 64, 67, 73, 76, 88, 89, 92 |
| T | \TeX{} and \LaTeX{} commands: | 3 |
| | \ifpackagelater | 52, 62, 87 |
| | \@protected@testopt | 72 |
| | \@testopt | 42 |
| | \tex_endinput:D | 12 |
| tl commands: | \c_space_tl | 32, 33, 37, 42, 47, 52, 54, 62, 72, 87, 89 |
| | \tl_const:Nn | 18, 19 |
| | \tl_if_in:NnTF | 15, 29, 39, 49, 59, 69, 84 |
| | \tl_new:N | 21, 22 |
| | \tl_put_left:Nn | 57, 67, 76, 92 |
| | \tl_put_right:Nn | 37, 47 |
\tl_set:Nn 26, 27, 80
\tl_to_str:n 20, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157

\token_to_str:N 32, 42, 52, 62, 72, 87, 88
\xpatchbibdriver 158
\xpatchbibmacro 102
\xpatchcmd 97
\xpatchfieldformat 110
\xpatchindexfieldformat 134
\xpatchindexlistformat 150
\xpatchindexnameformat 142
\xpatchlistformat 126
\xpatchnameformat 118
\xpretobibdriver 160
\xpretobibmacro 104
\xpretocmd 98
\xpretocomplete 112
\xpretoindexfieldformat 136
\xpretoindexlistformat 152
\xpretoindexnameformat 144
\xpretolistformat 128
\xpretobibdriver 164
\xshowbibdriver 108
\xshowcommand 100
\xshowfieldformat 116
\xshowindexfieldformat 140
\xshowindexlistformat 156
\xshowindexnameformat 148
\xshowlistformat 132
\xshownameformat 124

Change History

v0.2
General: Added \x...bibdriver macros; fixed a bug for control symbols defined with \newcommand and an optional argument.

v0.3
General: Added a bunch of biblatex related commands
v0.3a
General: Changed deprecated function

13