The **regexpatch** package*
Replacing etoolbox patching commands

Enrico Gregorio†

Released 2018/05/02

**Important preliminary notice**

This is an experimental version and it might cease to work if the commands in the package l3regex are modified. When that LATEX3 package will be declared stable, this package will replace xpatch and calling \usepackage{regexpatch} will load the main package. Use at own risk.

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 \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!.

*This file describes version 0.2d, last revised 2018/05/02.
†E-mail: Enrico DOT Gregorio AT univr DOT it
As usual, after \rightarrow we see the definition. In order to use \texttt{\patchcmd} to change the exclamation mark into a hyphen one must do

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

(see the documentation of \texttt{etoolbox} for details about the arguments).

A similar situation happens if \texttt{\xyz} has been defined by

\texttt{\DeclareRobustCommand{\xyz}{something}}

A \texttt{\show\xyz} instruction would show the cryptic

\begin{verbatim}
> \xyz=macro:
\rightarrowprotect\xyz .
\end{verbatim}

and only a close look reveals the clever trick used by the L\TeX\ team: the \texttt{\protect} is not applied to \texttt{\xyz}, but to the macro \texttt{\xyz/uni2423} which has a space at the end of its name!

And this macro is the one that contains the real definition. Indeed,

\texttt{\expandafter\show\csname xyz\space\endcsname}

produces the message

\begin{verbatim}
> \xyz =\long macro:
\rightarrowsomething.
\end{verbatim}

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

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

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

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

one has to combine the two tricks:

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

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

The package is now completely independent of \texttt{etoolbox}. It doesn’t feature commands analogous to \texttt{\preto} and \texttt{\appto} that, in the author’s opinion, are a bit dangerous, since somebody might apply them to commands defined with \texttt{\DeclareRobustCommand} or \texttt{\newrobustcmd}, with the obvious problems.

The \texttt{regexpatch} package uses many features of the L\TeX\3 experimental packages, in particular of \texttt{l3regex}. This has a clear advantage: we can have a \texttt{*}-variant of \texttt{\patchcmd} that does a “replace all” which can avoid multiple uses of \texttt{\patchcmd} on the same macro. Moreover there’s a very powerful \texttt{\regexpatchcmd} function that uses regular expression syntax for search and replace which can even patch commands defined under different category code setup.

For example, let’s see how the L\TeX\ kernel defines \texttt{\strip@pt}:
The same result can be obtained by
\begingroup\def\defrem@pt{\endgroup\def\rem@pt##1.##2pt{##1\ifnum##2>\z@.##2\fi}}
\regexpatchcmd{\defrem@pt}{pt}{\cOp\cOt}{}{}
\defrem@pt
\def\strip@pt{\expandafter\rem@pt\the}
Perhaps not so striking, but the pattern seems to be more intuitive; however the package supplies also a function for patching the parameter text of a macro:
\begingroup\def\defrem@pt{\endgroup\def\rem@pt#1.#2pt{#1\ifnum#2>\z@.#2\fi}}
\xpatchparametertext{\rem@pt}{pt}{\cO p \cO t}{}{}
Of course, reading the manual of l3regex is necessary for being able to exploit the full power of \regexpatchcmd or \xpatchparametertext; in this case, \cO p (the space in between is optional) specifies a character ‘p’ with category code ‘other’. Actually neither the \cO escape is necessary, as all letters in a replacement text in the context of regular expressions has category code 12 by default, but clarity is often to be preferred to efficiency.

2 Important notices

If the command to be patched contains ‘@-commands’ in its replacement text, always ensure that the patching code is enclosed between \makeatletter and \makeatother; this is different from what etoolbox requires. It’s recommended to turn on \tracingxpatches when testing a patch, to get maximum information.

Some people like to add informative messages to the (failure) code in the patching commands. Usually I’m lazy and don’t do it; when testing I find it better to trace the patchings or add \ddt to the (failure) code. Adding warnings to the (success) code is annoying for the user.

3 Acknowledgment

This package would not exist without the l3regex package and Bruno Le Floch. Some parts of l3regex were added just because I asked for them while developing the present package. Thanks also to Joseph Wright and all the \LaTeX{}3 team.
4 Commands

The main commands introduced by this package are

- \xpretocmd
- \xapptocmd
- \xpatchcmd
- \regexpatchcmd

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 \newrobustcmd, \renewrobustcmd, \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.

There is also added functionality that etoolbox doesn’t provide (at least easily for the first command):

- \xpatchoptarg
- \xpatchparametertext
- \checkifpatchable

Moreover the package defines

- \xpretobibmacro
- \xapptobibmacro
- \xpatchbibmacro
- \regexpatchbibmacro

that can be used to patch commands defined with biblatex’s \newbibmacro. Say that we have

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

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

\xpatchbibmacro{foo.bar}{and}{und}{}}
Patching these macros with `etoolbox` requires resorting to the very cryptic
\expandafter\patchcmd\csname abx@macro@\detokenize{foo.bar}\endcsname
(and)(und){}\{}
that would become an astonishing
\expandafter\patchcmd\csname\expandafter\string\csname
abx@macro@\detokenize{foo.bar}\endcsname\endcsname
(and)(und){}\{}
if the original definition had been with an optional argument, say
\newbibmacro{foo.bar}[2][x1 and #2]}

For `biblatex` users there are also

- \xpretobibdriver
- \xapptobibdriver
- \xpatchbibdriver
- \regexpatchbibdriver

for patching commands defined with `DeclareBibliographyDriver`. One could use, for
patching the driver `foo`,
\makeatletter
\patchcmd{\blx@bbx@foo}{X}{Y}{success}{failure}
\preto{\blx@bbx@foo}{P}
\appto{\blx@bbx@foo}{A}
\makeatother

but having a lighter interface can be handy. Since our macros use `\pretocmd` and
`\apptocmd` for consistency, remember to always use the `{success}` and `{failure}` ar-
guments also with `\xpretobibdriver` and `\xapptobibdriver`.

Under the same philosophy, one can use the macros

- \xpatchfieldformat,
  `\pretocmd\xapptocmd`,
- \xpatchnameformat,
  `\pretocmd\xapptocmd`,
- \xpatchlistformat,
  `\pretocmd\xapptocmd`,
- \xpatchindexfieldformat,
  `\pretocmd\xapptocmd`,
- \xpatchindexnameformat,
  `\pretocmd\xapptocmd`,

5
for the \texttt{biblatex} internal macro defined respectively with

\texttt{\DeclareFieldFormat, \DeclareNameFormat, \DeclareListFormat,}
\texttt{\DeclareIndexFieldFormat, \DeclareIndexNameFormat, \DeclareIndexListFormat.}

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

Finally, the package defines the commands

\begin{itemize}
  \item \texttt{\xshowcmd}
  \item \texttt{\xshowbibmacro}
  \item \texttt{\xshowbibdriver}
  \item \texttt{\xshowfieldformat}
  \item \texttt{\xshownameformat}
  \item \texttt{\xshowlistformat}
  \item \texttt{\xshowindexfieldformat}
  \item \texttt{\xshowindexnameformat}
  \item \texttt{\xshowindexlistformat}
  \item \texttt{\tracingxpatches}
\end{itemize}

The first three 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{bib} ones are for the corresponding \texttt{biblatex} macros. The last one takes an optional argument for activating and deactivating the tracing system. So

\texttt{\tracingxpatches}

will activate it (it’s equivalent to \texttt{\tracingxpatches[1]}), while

\texttt{\tracingxpatches[0]}

will stop issuing messages.

\section{Syntax}

Here is the formal syntax of the commands.

\begin{verbatim}
\xpatchcmd{⟨command⟩}{⟨search⟩}{⟨replace⟩}{⟨success⟩}{⟨failure⟩}
\xpretocmd{⟨command⟩}{⟨prepend⟩}{⟨success⟩}{⟨failure⟩}
\xapptocmd{⟨command⟩}{⟨append⟩}{⟨success⟩}{⟨failure⟩}
\xpatchbibmacro{⟨name⟩}{⟨search⟩}{⟨replace⟩}{⟨success⟩}{⟨failure⟩}
\xpretobibmacro{⟨name⟩}{⟨prepend⟩}{⟨success⟩}{⟨failure⟩}
\xapptobibmacro{⟨name⟩}{⟨append⟩}{⟨success⟩}{⟨failure⟩}
\end{verbatim}
Here ⟨command⟩ is the command’s name (with the backslash), while ⟨name⟩ is the string that appears as the argument to \newbibmacro, \DeclareBibliographyDriver, \DeclareFieldFormat, \DeclareListFormat, \DeclareIndexFieldFormat, \DeclareIndexNameFormat or \DeclareIndexListFormat respectively; ⟨search⟩, ⟨replace⟩, ⟨prepend⟩ and ⟨append⟩ are the list of tokens that are to be used for the specific tasks; ⟨success⟩ and ⟨failure⟩ are token lists to be executed if the patching succeeds or fails respectively. I find it useful to use \vddt as ⟨failure⟩, so that \TeX will stop for the undefined control sequence when the patching fails.

All the \texttt{\ldots fmt} macros have an optional argument that by default is *.

In the commands whose name contains the string regex, both ⟨search⟩ and ⟨replace⟩ are understood to represent regular expressions. Check with the documentation of \texttt{\l3regex} for details.

The *-variants of the patch type commands means that the replacement is performed on all matches. With \texttt{\xshowcmd*\foo} one gets all information on \texttt{\foo}, as if the tracing system were activated, including the default optional argument, if existent. So it’s best to use it before trying \texttt{\xpatchoptarg} (and all the other commands, of course).

A curiosity about optional arguments: if one defines

\newcommand{\foo}[1][\{bar\}]{-#1-}
then the braces around \textit{bar} are stripped off. So with
\newcommand{\foo}[1]{\textit{bar}}{-#1-}
all text following the call of \foo without an optional argument would be set in italics; one needs two sets of braces, in this case. However,
\xpatchoptarg{\foo}{\textit{bar}}
would not strip the braces.

It’s important to remember that patching commands that have @ in their name must always be performed between \makeatletter and \makeatother.

6 Examples

From http://tex.stackexchange.com/a/42894: the series of successive patches for changing the three occurrences of \texttt{\mathcode} in @addligto into \texttt{\Umathcodenum} can become
\xpatchcmd*{@addligto}{\mathcode}{\Umathcodenum}{}{}

while the code
\expandafter\patchcmd\csname mathligsoff \endcsname
{\mathcode}{\Umathcodenum}{()}
needed without regexpatch can become
\xpatchcmd{\mathligsoff}{\mathcode}{\Umathcodenum}{}{}

Another one: changing the space reserved for the theorem number in the ‘List of theorems’ provided by ntheorem could be obtained with etoolbox’s \patchcmd by
\patchcmd{\thm@thmline}{2.3em}{5em}{}{}
\patchcmd{\thm@thmline@name}{2.3em}{5em}{}{}
\patchcmd{\thm@thmline@noname}{2.3em}{5em}{}{}

if the hyperref option is not used, but a long series of patches would be needed with the option, as \texttt{2.3em} appears three times in each macro. With regexpatch one can do independently of the option:
\xpatchcmd*{\thm@thmline}{2.3em}{5em}{}{}
\xpatchcmd*{\thm@thmline@name}{2.3em}{5em}{}{}
\xpatchcmd*{\thm@thmline@noname}{2.3em}{5em}{}{}

A user asked how to patch the \rubric environment in the ‘CurVe’ class in order to avoid the repetition of the rubric’s title on continuation pages. The environment is based on longtable and the task is to remove the \texttt{\endhead} material, which is delimited by \texttt{\endfirsthead} and \texttt{\endhead}. Instead of
\patchcmd{\rubric}
{\endfirsthead}@rubrichead{#1@continuedname}\*[rubricspace]\endhead
{\endfirsthead}{()}

one can more simply exploit regular expressions:
\makeatletter % the replacement text has @-commands
\renewcommand{\rubric}{
  \c{endfirsthead} .* \c{endhead} }
\makeatother

Assume you want to insert a patch in the argument of a command; with the tra-
ditional method this is possible provided the patch text doesn’t contain #. Here’s an
example

\makeatletter
\usepackage{etoolbox} % for \ifdef
\ifdef{\@old@part}
  {
    \renewcommand{\@old@part}{\$} % regex representing the end
    {\c{gdef}\c{cont@name@part}\cB\{\cP\#2\cE\}} % replacement
    {()}%
  }
  {
    \renewcommand{\@part}{\$}
    {\c{gdef}\c{cont@name@part}\cB\{\cP\#2\cE\}}
    {()}%
  }
\makeatother

We want to add \gdef{\cont@name@part}{#2} at the end of the replacement text, distin-
guishing when hyperref is loaded or not. So we patch the command by doing just what’s
requested. The example is a bit contrived, as using \ifdef\else instead of the argument
form wrapper would allow the traditional \apptocmd. However, other applications may
be foreseen.

A problem raised on comp.text.tex in 2008 was to extract the number from the
name of the file being typeset; the name was of the form lecture15.tex and the question
was how to define a macro \lecturenumber that acted on \jobname to do its work. The
obvious

\def\\lecturenumber{\expandafter\\extractnumber\\jobname;}
\def\\extractnumber lecture#1;{#1}

doesn’t work because the characters produced by \jobname all have category code 12
(spaces have 10, as usual). A nifty solution was provided by David Kastrup:

\begingroup
  \escapechar=-1
  \expandafter\\extractnumber\string\lecture#1;{#1}
\endgroup

Now with \xpatchparametertext one can do

\def\\lecturenumber{\expandafter\\extractnumber\\jobname;}
\def\\extractnumber lecture#1;{#1}
\xpatchparametertext\\extractnumber\\lecture\{lecture\}{lecture}{()}
recalling that the substitution performed by \texttt{l3regex} uses category code 12 characters by default. The command can be generalized to accept any (fixed) prefix:

\begin{verbatim}
def\setupfilenumber#1{
def\filenumber{\expandafter\extractnumber\jobname;}
def\extractnumber#1##1;{##1}\
xpatchparameter\extractnumber{#1}{#1}{}{}\}
\setupfilenumber{lecture}
\end{verbatim}

where the prefix is passed to \texttt{\setupfilenumber} and the macro to use is \texttt{\filenumber}.

A proper \texttt{\LaTeX}3 definition might be

\begin{verbatim}
\NewDocumentCommand{\setupfilenumber}{ m }{
  \group_begin:
  \cs_set:Npx \filenumber_aux: {
    \group_end:
    \cs_set:Npn \exp_not:N \filenumber_extract:w {
      \tl_to_str:n { #1 } ####1 ; { ####1 }
    }
  }
  \filenumber_aux:
}
\NewDocumentCommand{\filenumber} {}{
  \exp_after:wN \filenumber_extract:w \c_job_name_tl ;
}\end{verbatim}

\section{The implementation of regexpatch}

The usual starting stuff.

\begin{verbatim}
\ProvidesExplPackage{\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}
\@ifpackagelater { expl3 } { 2012/01/19 } { }
\PackageError { regexpatch } { Support-package-l3kernel-too-old. } {
  Please-install-an-up-to-date-version-of-l3kernel-
  using-your-TeX-package-manager-or-from-CTAN.\ \ \ \Loading-regexpatch-will-abort!
}
\tex_endinput:D
\RequirePackage{sparse}
\end{verbatim}

\subsection{Variables}

We define a bunch of variables: some booleans and token lists. The first tells us when the macro to patch has been defined by \texttt{\DeclareRobustCommand}, the second if it has an optional argument, the third if it’s patchable, that is it can be reconstructed from its
decomposition under the current category code regime. The last boolean is used for the tracing system: if true, messages about patching are issued.

\bool_new:N \l_xpatch_protect_bool
\bool_new:N \l_xpatch_optional_bool
\bool_new:N \l_xpatch_patchable_bool
\bool_new:N \l_xpatch_tracing_bool

The token list variables contain various items regarding the macro to patch: the name, the first level replacement text (we distinguish it from the ‘real’ replacement text), the prefixes, the argument spec and the ‘real’ replacement text.

\tl_new:N \l_xpatch_name_tl
\tl_new:N \l_xpatch_repl_tl
\tl_new:N \l_xpatch_prefix_tl
\tl_new:N \l_xpatch_arg_tl
\tl_new:N \l_xpatch_replacement_tl
\tl_new:N \l_xpatch_type_tl % for debugging messages

A variant for checking a regex match so that we can give the second argument as a token list.

\cs_generate_variant:Nn \regex_match:nnT {nV}

7.2 Functions

The function \xpatch_main_check:N is responsible for telling us what kind of macro we’re patching. Only one of the first four tests can be true; if none is, the macro is not ‘special’ and can be patched without doing anything particular to get its ‘real name’. The check consists in matching with a suitable regex at the start of the replacement text (which is in detokenized form). If the macro passes one of the first two tests, it can still have an optional argument, so a supplementary test is needed.

Some technical remarks. Suppose we have the following definitions:

\DeclareRobustCommand{\xaa}[1]{\xaa (DeclareRobustCommand-noopt)}
\DeclareRobustCommand{\xab}[1][x]{\xab (DeclareRobustCommand-opt)}
\newcommand{\xac}[1][\]{\xac (newcommand-opt)}
\newrobustcmd{\xad}[1][\]{\xad (newrobustcmd-opt)}
\DeclareRobustCommand{\1}[1]{\1 (DeclareRobustCommand-noopt)}
\DeclareRobustCommand{\2}[1][\]{\2 (DeclareRobustCommand-opt)}
\newcommand{\3}[1][\]{\3 (newcommand-opt)}
\newrobustcmd{\4}[1][\]{\4 (newrobustcmd-opt)}

Then the first level expansions are, respectively,

%+\protect\xaar\xaar\xaar+
%+\protect\xab\xab\xab\xab+
%+@protected@testopt\xcd\xcd\xcd\xcd{}
%+@testopt\xad\xad\xad\xad{}
%+\@protect\xar\xar\xar\xar+
%+\xar\xar\xar\xar+
%+\@protect\prot\prot\prot\prot{}
%+\prot\prot\prot\prot{}
%+\@protect\prot\prot\prot\prot{}
%+\prot\prot\prot\prot{}
%+\@protect\prot\prot\prot\prot{}
%+\prot\prot\prot\prot{}
%+

11
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, \&\_ 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. If the macro is recognized to have an optional argument, its default value is stored in \tl_xpatch_repl_tl (that we wouldn’t use any more) to be shown by \xshowcmd* or when the tracing system is active: we throw away everything except what’s contained between the final pair of braces.

\cs_new_protected:Npn \xpatch_main_check:N #1
{\bool_set_false:N \l_xpatch_protect_bool
\bool_set_false:N \l_xpatch_optional_bool
\tl_set:Nx \l_xpatch_name_tl { \cs_to_str:N #1 }
\tl_set:Nx \l_xpatch_repl_tl { \token_get_replacement_spec:N #1 }
\tl_clear:N \l_xpatch_type_tl
\regex_match:nVT % \DeclareRobustCommand<control word>
{^\protect\ \\u{l_xpatch_name_tl}\ \ }
\l_xpatch_repl_tl
{\bool_set_true:N \l_xpatch_protect_bool
\tl_put_right:Nx \l_xpatch_name_tl { \c_space_tl }
\tl_set:Nn \l_xpatch_type_tl { DRCw }
}
\regex_match:nVT % \DeclareRobustCommand<control symbol>
{^\x@protect\ \\u{l_xpatch_name_tl}\ }
\l_xpatch_repl_tl
{\bool_set_true:N \l_xpatch_protect_bool
\tl_put_right:Nx \l_xpatch_name_tl { \c_space_tl }
\tl_set:Nn \l_xpatch_type_tl { DRCs }
}
\regex_match:nVT % \newcommand<control word> with opt arg
{^\@\protected@\testopt\ \\\u{l_xpatch_name_tl}\\}
\l_xpatch_repl_tl
{\bool_set_true:N \l_xpatch_optional_bool
\tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
\tl_set:Nn \l_xpatch_type_tl { ncw+o }
}
\regex_match:nVT % \newcommand<control symbol> with opt arg
{^\@\protected@\testopt\ \\\u{l_xpatch_name_tl}\\\\}
\l_xpatch_repl_tl
{\bool_set_true:N \l_xpatch_optional_bool
\tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
\tl_set:Nn \l_xpatch_type_tl { ncs+o }
}
\regex_match:nVT % \newrobustcmd<any cs> with opt arg
{^\@\testopt\ \\\u{l_xpatch_name_tl}}
\l_xpatch_repl_tl

12
We use the information gathered with \xpatch_main_check:N to perform the patch; the macro to patch is \#2, the function to execute is \#1; in case the macro’s name is misspelled, the following arguments will be ignored because they have already been absorbed. The main function is \xpatch_main_four:NNnnnn, where the four refers to the number of braced arguments for the patch and regexpatch type macros; we define also a three function for preto and apppto macros, and a zero function for the show macros. We also define the variants taking a name as their second argument.

\cs_new_protected:Npn \xpatch_main_four:NNnnnn \#1 \#2 \#3 \#4 \#5 \#6
\begin{verbatim}
\cs_if_exist:NTF \#2
{ \xpatch_main_check:N \#2
  \bool_if:NT \l_xpatch_tracing_bool
  { \xpatch_message_cstype:N \#2 }
  \exp_after:wN \#1 \cs:w \l_xpatch_name_tl \cs_end: {\#3}{\#4}{\#5}{\#6}
}
\end{verbatim}

\cs_new_protected:Npn \xpatch_main_three:NNnnn \#1 \#2 \#3 \#4 \#5
\begin{verbatim}
\cs_if_exist:NTF \#2
{ \xpatch_main_check:N \#2
  \bool_if:NT \l_xpatch_tracing Bool
  { \xpatch_message_cstype:N \#2 }
  \exp_after:wN \#1 \cs:w \l_xpatch_name_tl \cs_end: {\#3}{\#4}{\#5}{}
}
\end{verbatim}

\cs_new_protected:Npn \xpatch_main_two:NNnn \#1 \#2 \#3 \#4 \#5
\begin{verbatim}
\cs_if_exist:NTF \#2
{ \xpatch_main_check:N \#2
  \bool_if:NT \l_xpatch_tracing_Bool
  { \xpatch_message_cstype:N \#2 }
  \exp_after:wN \#1 \cs:w \l_xpatch_name_tl \cs_end: {\#3}{\#4}{\#5}{}
}
\end{verbatim}
Now we define the patching functions. We get all the parts in which a macro can be split: prefixes, parameter text and replacement text; the name is already available. The token lists \l_xpatch_X_tl will contain the prefix or parameter text or replacement text of \#1 first in ‘detokenized’ and then in ‘tokenized’ form.

After possible modifications to the replacement text, we can call \xpatch_rebuild:N to redo the definition of \#1; we can also use it for checking if \#1 is patchable. Of course we need to use \tex_def:D at this point. Apologies to the developers of \LaTeX3 that recommend never using :D functions.

To check if \#1 is patchable, we rebuild it as \xpatch_tmpa:w and look whether \#1 and \xpatch_tmpa:w are the same. This is always the first thing to do, so we put \xpatch_get_all:N here; \#1 is the macro to patch.
Defining the internal versions of \xpretocmd and \xapptocmd is easy: we check if the
command is patchable and, if so, we prepend or append the second argument to the
replacement text and rebuild the macro, then we execute the \textit{success} code. If the patch
isn’t possible we just execute the \textit{failure} code.

\begin{verbatim}
cs_new_protected:Npn \xpatch_pretocmd:Nnnn #1 #2 #3 #4
  {\
    \xpatch_check_patchable:N #1
    \bool_if:NTF \l_xpatch_patchable_bool
      {\tl_put_left:Nn \l_xpatch_replacement_tl \{ \ #2 \}
       \xpatch_rebuild:N #1
       \ #3
      }
      {\ #4
      }
  }
cs_new_protected:Npn \xpatch_apptocmd:Nnnn #1 #2 #3 #4
  {\
    \xpatch_check_patchable:N #1
    \bool_if:NTF \l_xpatch_patchable_bool
      {\tl_put_right:Nn \l_xpatch_replacement_tl \{ \ #2 \}
       \xpatch_rebuild:N #1
       \ #3
      }
      {\ #4
      }
  }
\end{verbatim}

Substituting tokens in the replacement text is a bit harder, but not conceptually dif-
ferent. First the internal version of \regexpatchcmd(*): check if #1 is patchable, do the
replacement if possible; beware that characters in the replacement string are of category
12 by default. We use \texttt{\regex_replace_all:mmNTF} and \texttt{\regex_replace_once:mmNTF}
in order to pass correctly the success or failure arguments.

\cs_new_protected:Npn \xpatch_regexpatchcmd_all:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \regex_replace_all:nnNTF { #2 } { #3 } \l_xpatch_replacement_tl
\{ \xpatch_rebuild:N #1 #4 \}
\{ #5 \}
\}
\{ #5 \}
\}

\cs_new_protected:Npn \xpatch_regexpatchcmd_once:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \regex_replace_once:nnNTF { #2 } { #3 } \l_xpatch_replacement_tl
\{ \xpatch_rebuild:N #1 #4 \}
\{ #5 \}
\}
\{ #5 \}
\}

\cs_new_protected:Npn \xpatch_patchcmd_once:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \tl_set:Nn \l_tmpa_tl { #2 }
\tl_set:Nn \l_tmpb_tl { #3 }
\regex_replace_once:nnNTF \{ \u{l_tmpa_tl} \} \{ \u{l_tmpb_tl} \}
\l_xpatch_replacement_tl
\{ \xpatch_rebuild:N #1 #4 \}
\{ #5 \}
\}
\{ #5 \}
\}

\cs_new_protected:Npn \xpatch_patchcmd_all:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \tl_set:Nn \l_tmpa_tl { #2 }
\}

Thanks to the features of l3regex, we can also implement directly the analog of \patchcmd, but also with a ‘replace all’ version.

\cs_new_protected:Npn \xpatch_patchcmd_all:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \tl_set:Nn \l_tmpa_tl { #2 }
\tl_set:Nn \l_tmpb_tl { #3 }
\regex_replace_once:nnNTF \{ \l_tmpa_tl \} \{ \l_tmpb_tl \}
\l_xpatch_replacement_tl
\{ \xpatch_rebuild:N #1 #4 \}
\{ #5 \}
\}
\{ #5 \}
\}

\cs_new_protected:Npn \xpatch_patchcmd_all:Nnnnn #1 #2 #3 #4 #5
\{ \xpatch_check_patchable:N #1 \bool_if:NTF \l_xpatch_patchable_bool
\{ \tl_set:Nn \l_tmpa_tl { #2 }
\}

16
\tl_set:Nn \l_tmpb_tl { #3 }
\regex_replace_all:nnNTF { \u{l_tmpa_tl} } { \u{l_tmpb_tl} } \l_xpatch_replacement_tl \xpatch_replacement_tl \xpatch_rebuild:N #1 #4 \l_xpatch_replacement_tl \xpatch_rebuild:N #1 #5
\cs_new_protected:Npn \xpatch_message:n #1 \bool_if:NT \l_xpatch_tracing_bool \msg_term:n { xpatch~message \ #1 }
\cs_new:Npn \xpatch_message_cstype:N #1 \str_case:onF { \l_xpatch_type_tl } { DRCw } { \token_to_str:N #1~is~a~control~word~defined~with~\token_to_str:N \DeclareRobustCommand }
{ DRCw+o } { \token_to_str:N #1~is~a~control~word~defined~with~\token_to_str:N \DeclareRobustCommand~and~a~default~optional~argument~\l_xpatch_repl_tl }
{ DRCs } { \token_to_str:N #1~is~a~control~symbol~defined~with~\token_to_str:N \DeclareRobustCommand }
{ DRCs+o } { \token_to_str:N #1~is~a~control~symbol~defined~with~\token_to_str:N \DeclareRobustCommand~and~a~default~optional~argument~\l_xpatch_repl_tl }
Now the tracing system.
\cs_new_protected:Npn \xpatch_message:n #1 \bool_if:NT \l_xpatch_tracing_bool \msg_term:n { xpatch~message \ #1 }
\cs_new:Npn \xpatch_message_cstype:N #1 \str_case:onF { \l_xpatch_type_tl } { DRCw } { \token_to_str:N #1~is~a~control~word~defined~with~\token_to_str:N \DeclareRobustCommand }
{ DRCw+o } { \token_to_str:N #1~is~a~control~word~defined~with~\token_to_str:N \DeclareRobustCommand~and~a~default~optional~argument~\l_xpatch_repl_tl }
{ DRCs } { \token_to_str:N #1~is~a~control~symbol~defined~with~\token_to_str:N \DeclareRobustCommand }
{ DRCs+o } { \token_to_str:N #1~is~a~control~symbol~defined~with~\token_to_str:N \DeclareRobustCommand~and~a~default~optional~argument~\l_xpatch_repl_tl }
}
7.3 The user level functions

Here are the functions for patching usual macros; the *-variants for \xpatchcmd and \regexpatchcmd do a ‘replace all’. All arguments are declared ‘long’ with +m because we may need \par in them.

\NewDocumentCommand{\xshowcmd}{s+{m}}{
\IfBooleanT{#1}{
\group_begin:
\bool_set_true:N \l_xpatch_tracing_bool
}
\xpatch_main_zero:NN \cs_show:N #2
\IfBooleanT{#1}{
\group_end:
}
}

\NewDocumentCommand{\xpretocmd}{+m+{m}+{m}+{m}}{
\xpatch_main_three:NNnnn \xpatch_pretocmd:Nnnn #1 {#2} {#3} {#4}
}
The functions for patching \cite{biblatex} related macros that are given by name and we'll use the already defined variants.

\NewDocumentCommand{\xshowbibmacro}{s+m}{
\IfBooleanT{#1}{
\group_begin:n:
  \bool_set_true:N \l_xpatch_tracing_bool
\group_end:n:}
\xpatch_main_zero:Nc \cs_show:N { abx@macro@ \tl_to_str:n {#2} }
\IfBooleanT{#1}{
  \group_end:n:}
}

\NewDocumentCommand{\xpretobibmacro}{+m+m+m+m}{
\xpatch_main_three:Ncnnn \xpatch_pretocmd:Nnnn
  \tl_to_str:n {#1} } {#2} {#3} {#4}
}

\NewDocumentCommand{\xapptobibmacro}{+m+m+m+m}{
\xpatch_main_three:Ncnnn \xpatch_apptocmd:Nnnn
  \tl_to_str:n {#1} } {#2} {#3} {#4}
}

\NewDocumentCommand{\xpatchbibmacro}{s+m+m+m+m+m}{
\IfBooleanTF{#1}{
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
  \tl_to_str:n {#2} } {#3} {#4} {#5} {#6}
}{
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
  \tl_to_str:n {#2} } {#3} {#4} {#5} {#6}
}

\NewDocumentCommand{\regexpatchbibmacro}{s+m+m+m+m+m}{
\IfBooleanTF{#1}{
  \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_all:Nnnnn
  \tl_to_str:n {#2} } {#3} {#4} {#5} {#6}
}{
  \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_once:Nnnnn
  \tl_to_str:n {#2} } {#3} {#4} {#5} {#6}
}
\IfBooleanTF{#1}
{
  \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_all:Nnnnn
  \{ \abx@macro\tl_to_str:n\{#2\}\{#3\}\{#4\}\{#5\}\{#6\}
  \}
\}

\IfBooleanTF{#1}
{
  \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_once:Nnnnn
  \{ \abx@macro\tl_to_str:n\{#2\}\{#3\}\{#4\}\{#5\}\{#6\}
  \}
\}

\NewDocumentCommand{\xshowbibdriver} { s +m }
{
  \IfBooleanT{#1}
  {
    \group_begin:
    \bool_set_true:N \l_xpatch_tracing_bool
  }
  \xpatch_main_zero:Nc \cs_show:N \{ \blx@bbx@#2 \}
  \IfBooleanT{#1}
  {
    \group_end:
  }
}
\NewDocumentCommand{\xpretobibdriver} { +m +m +m +m }
{ \exp_args:Nc \xpatch_pretocmd:Nnnnn \{\blx@bbx@#1\} \{#2\}\{#3\}\{#4\} }
\NewDocumentCommand{\xapptobibdriver} { +m +m +m +m }
{ \exp_args:Nc \xpatch_apptocmd:Nnnnn \{\blx@bbx@#1\} \{#2\}\{#3\}\{#4\} }
\NewDocumentCommand{\xpatchbibdriver} { s +m +m +m +m +m }
{ \IfBooleanTF{#1}
  {
    \exp_args:Nc \xpatch_patchcmd_all:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }
  { \exp_args:Nc \xpatch_patchcmd_once:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }
}
\NewDocumentCommand{\regexpatchbibdriver} { s +m +m +m +m +m }
{ \IfBooleanTF{#1}
  {
    \exp_args:Nc \xpatch_regexpatchcmd_all:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }
  { \exp_args:Nc \xpatch_regexpatchcmd_once:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }
}

Other \texttt{bibtex} related macros, added by request of the maintainers.
\NewDocumentCommand{\xshowfieldformat} { s O{*} +m }
{
  \IfBooleanT{#1}
  {
    \group_begin:
    \bool_set_true:N \l_xpatch_tracing_bool
  }
  \xpatch_main_zero:Nc \cs_show:N \{ \abx@ffd0\tl_to_str:n\{#2\} \tl_to_str:n\{#3\} \}
  \IfBooleanT{#1}
  {
    \group_end:
  }
}
\NewDocumentCommand{\xpretofieldformat} { s O{*} +m +m +m +m +m }
{ \exp_args:Nc \xpatch_pretocmd:Nnnnn \{\blx@bbx@#1\} \{#2\}\{#3\}\{#4\}\{#5\}\{#6\} }
\NewDocumentCommand{\regexpretofieldformat} { s O{*} +m +m +m +m +m }
{ \exp_args:Nc \xpatch_regexpatchcmd_all:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }
\NewDocumentCommand{\regexapptofieldformat} { s O{*} +m +m +m +m +m }
{ \exp_args:Nc \xpatch_regexpatchcmd_once:Nnnnn \{\blx@bbx@#2\} \{#3\}\{#4\}\{#5\}\{#6\} }

{20}
\IfBooleanTF{#1}
  \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
  \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
{ abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}

\xpatchfieldformat{ s O{*} +m +m +m +m }
\NewDocumentCommand\xpatchfieldformat { s O{*} +m +m +m +m +m }
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
{ abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}

\regexpatchfieldformat{ s O{*} +m +m +m +m +m }
\NewDocumentCommand\regexpatchfieldformat { s O{*} +m +m +m +m +m }
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
  \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
{ abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}

\xshownameformat{ s O{*} +m }
\NewDocumentCommand\xshownameformat { s O{*} +m }
  \group_begin:
  \bool_set_true:N \l_xpatch_tracing_bool
\NewDocumentCommand{\xshowlistformat}{s O{*} +m}{\IfBooleanT{#1}{\group_begin:\bool_set_true:N \l_xpatch_tracing_bool} \xpatch_main_zero:Nc \cs_show:N { abx@lfd@ \tl_to_str:n {#1} \tl_to_str:n {#2} } \IfBooleanT{#1}{\group_end:}}
\NewDocumentCommand{\xpretolistformat}{s O{*} +m +m +m +m}{\IfBooleanTF{#1}{\xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}}{\xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}}}
\NewDocumentCommand{\xapptolistformat}{s O{*} +m +m +m +m}{\IfBooleanTF{#1}{\xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}}{\xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}}}
\NewDocumentCommand{\xpatchlistformat}{s O{*} +m +m +m +m +m}{\IfBooleanTF{#1}{\xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}}{\xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}}}
\NewDocumentCommand{\regexpatchlistformat}{s O{*} +m +m +m +m +m}{\IfBooleanTF{#1}{\xpatch_main_five:Ncnnnnn \xpatch_patchcmd_all:Nnnnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}{#8}}{\xpatch_main_five:Ncnnnnn \xpatch_patchcmd_once:Nnnnnn { abx@lfd@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}{#8}}}
\NewDocumentCommand{\xshowindexfieldformat}{s O{*} +m}{
  \IfBooleanT{#1}{
    \group_begin:
    \bool_set_true:N \l_xpatch_tracing_bool
  }
  \xpatch_main_zero:Nc \cs_show:N { abx@fid@ \tl_to_str:n {#1} \tl_to_str:n {#2} }
  \IfBooleanT{#1}{
    \group_end:
  }
}
\NewDocumentCommand{\xpretoindexfieldformat}{s O{*} +m +m +m +m}{
  \IfBooleanTF{#1}{
    \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}
  }{
    \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}
  }
}
\NewDocumentCommand{\xapptoindexfieldformat}{s O{*} +m +m +m +m}{
  \IfBooleanTF{#1}{
    \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}
  }{
    \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}
  }
}
\NewDocumentCommand{\xpatchindexfieldformat}{s O{*} +m +m +m +m +m}{
  \IfBooleanTF{#1}{
    \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
  }{
    \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
    { abx@fid@ \tl_to_str:n {#2} \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
  }
}
\IfBooleanTF{#1}{
  \group_begin:
  \bool_set_true:N \l_xpatch_tracing_bool
}{
\xpatch_main_zero:Nc \cs_show:N { abx@fid@ \tl_to_str:n {#1} \tl_to_str:n {#2} }
\IfBooleanT{#1}{
  \group_end:
}
\newcommand{\xpatchindexnameformat}[1]{\IfBooleanTF{#1}
\startxpatchfour:ncnnnn \xpatch_patchcmd_all:Nnnnn
\endwith\xpatchfour:ncnnnn
\xpatchnameformat:nnnnnnnnnnn
\xpatch_patchcmd:Nnnnnnnnnn
\xpatch_patchcmd:once:Nnnnnnnnnnnn
\xpatch_patchcmd:once:Nnnnnnnnnnnn
\endxpatchindexnameformat}

\newcommand{\regexpatchindexnameformat}[1]{\IfBooleanTF{#1}
\startxpatchfour:ncnnnn \xpatch_regexpatchcmd_all:Nnnnn
\endwith\xpatchfour:ncnnnn
\xpatchnameformat:nnnnnnnnnnn
\xpatch_regexpatchcmd:Nnnnnnnnnnnn
\xpatch_regexpatchcmd:once:Nnnnnnnnnnnn
\xpatch_regexpatchcmd:once:Nnnnnnnnnnnn
\endregexpatchindexnameformat}

\newcommand{\xshowindexlistformat}[1]{\IfBooleanTF{#1}
\startxpatchthree:nc
\xpatch_pretocmd_all:Nnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\endxshowindexlistformat}

\newcommand{\xpretoindexlistformat}[1]{\IfBooleanTF{#1}
\startxpatchthree:nc
\xpatch_pretocmd_all:Nnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\endxpretoindexlistformat}

\newcommand{\xapptoindexlistformat}[1]{\IfBooleanTF{#1}
\startxpatchthree:nc
\xpatch_pretocmd_all:Nnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\xpatch_pretocmd:once:Nnnnnnnnnnnn
\endxapptoindexlistformat}
A macro to check if the macro is patchable. It just prints a message on the terminal and in the log file.
\NewDocumentCommand{\checkpatchable}{ +m }
\group_begin:
\bool_set_true:N \l_xpatch_tracing_bool
\xpatch_check_patchable:N #1
\group_end:

The last user level command: a macro for changing the optional argument in a macro that has one.
\cs_generate_variant:Nn \xpatch_all:N \cs_generate_variant:Nn \xpatch_rebuild:N \NewDocumentCommand{\xpatchoptarg}{ +m +m }
\group_begin:
\bool_if:NTF \l_xpatch_optional_bool
\xpatch_check_patchable:N #1
\group_end:
We have a macro with optional argument; so we strip off the first backslash from the name and proceed.

\tl_set:Nx \l_xpatch_name_tl \{ \tl_tail:V \l_xpatch_name_tl \}

Gather the prefix (it is \texttt{protected} when \texttt{#1} has been defined with \texttt{\newrobustcmd}).

\tl_set:Nf \l_xpatch_prefix_tl \{ \token_get_prefix_spec:N \texttt{#1} \}
\tl_clear:N \l_xpatch_prefix_tl
\tl_set_rescan:Nnx \l_xpatch_prefix_tl \{ \} \l_xpatch_prefix_tl

Get the replacement text in tokenized form: the control sequences have spaces in their names, so we can’t rely on \texttt{\token_get_replacement_spec:N} because the spaces would be lost.

\tl_set_eq:Nc \l_xpatch_replacement_tl \{ \l_xpatch_name_tl \}

Now we have to change the last item in the token list: we just store the new optional argument in a token list variable and do a regex substitution, based on the fact that the replacement text consists of control sequences, an open brace, the optional argument and a closed brace, so we anchor at the end of the token list.

\tl_set:Nn \l_tmpa_tl \{ \{ \texttt{#2} \} \}
\regex_replace_once:nnN { \cB. .* \cE. \Z } { \u{l_tmpa_tl} } \l_xpatch_replacement_tl

Now we rebuild the control sequence.

\xpatch_rebuild:c \{ \l_xpatch_name_tl \}

If the macro hasn’t an optional argument we issue a message.

{ \group_begin: \bool_set_true:N \l_xpatch_tracing_bool \xpatch_message:n \{ \token_to_str:N \texttt{#1}’ has no optional argument- or it has been defined with ‘xparse’ and operating- on such commands is (still) not supported \} \group_end: }

Just one more thing: enabling or disabling the tracing system.

\NewDocumentCommand{\tracingxpatches}{ O{1} }{ \int_compare:nTF \{ \texttt{#1} > 0 \} \{ \bool_set_true:N \l_xpatch_tracing_bool \} \{ \bool_set_false:N \l_xpatch_tracing_bool \} }

One more thing: patching the parameter text!

\NewDocumentCommand{\xpatchparametertext}{ +m +m +m +m +m }{ \xpatch_check_patchable:N \texttt{#1} \bool_if:NTF \l_xpatch_patchable_bool \{ \regex_replace_once:nN \{ \texttt{#2} \} \{ \texttt{#3} \} \l_xpatch_arg_tl \xpatch_rebuild:N \texttt{#1} }

28
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.

Symbols
\(\ldots\) 9, 34, 42, 50, 58, 66, 79, 105, 160, 265
\{ \ldots \} 89
\} \ldots 89

Numbers
\(\ldots\) 89
\(\ldots\) 34, 42, 50, 58, 66, 79

B
bool commands:
\bool_if:NTF \ldots 73, 87, 98, 176, 189, 202, 215, 228, 246, 263, 822, 855
\bool_new:N 15, 16, 17, 18
\bool_set_false:N 28, 29, 157, 166, 850
\bool_set_true:N 37, 45, 53, 61, 69, 82, 150, 339, 368, 415, 444, 506, 568, 630, 692, 754, 813, 836, 849

C
\cB 830
\cE 830
\checkpatchable 810
\cs commands:
\cs:w \ldots 77, 100
\cs_end: \ldots 77, 100
\cs_generate_variant:Nn \ldots 25, 119, 120, 121, 817, 818
\cs_if_eq:NNTF 148
\cs_if_exist:NTF 95, 144
\cs_new:Npn 268
\cs_new_protected:Npn \ldots 26, 93, 111, 115, 122, 131, 142, 173, 186, 199, 212, 225, 243, 261
\cs_show:N \ldots 341, 370, 417, 446, 508, 570, 632, 694, 756
\cs_to_str:N 30

D
\DeclareRobustCommand \ldots 33, 41, 78, 276, 283, 291, 298

E
exp commands:
\exp_after:wN 76, 100
\exp_args:Nc 424, 426, 430, 431, 436, 437
\exp_not:N 137
\exp_not:n 135, 138, 139
\ExplFileVersion 2
\ExplFileName 2
\ExplFileVersion 2

G
group commands:
\group_begin: \ldots 338, 367, 414, 443, 505, 567, 629, 691, 753, 812, 835
\group_end: \ldots 344, 373, 420, 449, 511, 573, 635, 697, 759, 815, 843

I
\int_compare:nTF 848

M
\msg_term:n 103, 265

N
\newcommand \ldots 49, 57, 306, 314
\NewDocumentCommand \ldots 334, 347, 349, 351, 357, 363,
P
\PackageError ........................................ 6
\ProvidesExplPackage .................................. 1

R
regex commands:
\regex_match:nnTF .................................... 25, 33, 41, 49, 57, 78
\regex_replace_all:nnNTF .......................... 204, 250
\regex_replace_once:nnN ......................... 89, 830, 857
\regex_replace_once:nnNTF ..................... 217, 232
\regexpatchbibdriver ............................... 433
\regexpatchbibmacro ................................. 398
\regexpatchcmd ....................................... 351
\regexpatchindexfieldformat ..................... 488
\regexpatchindexfieldformat .................... 674
\regexpatchindexlistformat ....................... 798
\regexpatchindexnameformat ....................... 736
\regexpatchlistformat ................................ 612
\regexpatchnameformat ................................ 550
\RequirePackage ...................................... 14

S
str commands:
\c_backslash_str ................................... 54, 62, 70, 83
\str_case:nnTF ....................................... 270

T
\ifx and \if\text{T\text{X}}\fi commands:
\ifpackageglater ...................................... 3
tex commands:
\tex_def:D ........................................... 136
\tex_endinput:D ...................................... 12
tl commands:
\c_space_tl .......................................... 38, 46
\tl_clear:N .......................................... 32, 826
\tl_new:N ............................................ 19, 20, 21, 22, 23, 24
\tl_put_left:Nn .................................... 54, 62, 70, 83, 178
\tl_put_right:Nn .................................... 38, 46, 84, 191
\tl_set:Nn ............................................ 30,
31, 39, 47, 55, 63, 71, 75, 124, 126,
128, 230, 231, 248, 249, 824, 825, 829
\tl_set_eq:NN ......................................... 828
\tl_set_rescan:Nn .................................. 125, 127, 129, 827
\tl_tail:n ............................................. 824
\tl_to_str:n .......................................... 370, 379, 384, 391,
395, 403, 407, 446, 457, 461, 469,
473, 481, 485, 493, 497, 508, 519,
523, 531, 535, 543, 547, 555, 559,
570, 581, 585, 593, 597, 605, 609,
617, 621, 632, 643, 647, 655, 659,
667, 671, 679, 683, 694, 705, 709,
717, 721, 729, 733, 741, 745, 756,
767, 771, 779, 783, 791, 795, 803, 807
\l_tmpa_tl ........................................... 230, 248, 829
\l_tmpb_tl ........................................... 231, 249
token commands:
\token_get_arg_spec:N ................................ 126
\token_get_prefix_spec:N .... 124, 825
\token_get_replacement_spec:N ..................... 31, 76, 128
\token_to_str:N ...................................... 106, 153, 160, 169, 275,
276, 282, 283, 290, 291, 297, 298,
305, 306, 313, 314, 321, 322, 330, 839
\tracingxpatches ..................................... 846

U
\u ......................................................... 34, 42,
50, 58, 60, 79, 233, 234, 251, 252, 830
use commands:
\use:n .................................................. 133

X
\xapptobibdriver ...................................... 425
\xapptobibmacro ........................................ 381
\xapptocmd ............................................ 349
\xapptofieldformat .................................. 464
\xapptoindexfieldformat ............................ 650
\xapptoindexlistformat ................................ 774
\xapptoindexnameformat ................................ 712
\xapptolistformat ...................................... 588
\xaptonameformat ...................................... 526
xpatch commands:
\xpatch_apptocmd:Nnnn ............................ 186, 350, 383, 426
\xpatch_apptocmd_all:Nnnn ..................... 468, 530, 592, 654, 716, 778
\xpatch_apptocmd_once:Nnnn .................... 472, 534, 596, 658, 720, 782
\_xpatch_arg_tl ..................................... 142,
175, 188, 201, 214, 227, 245, 814, 854
\xpatch_check_patchable:N ...................... 122, 146, 817
\xpatch_main_check:N .............................. 26, 97, 821
\xpatch_main_four:Nnnn ............................ 93, 113, 117, 121, 354, 355, 360,
361, 390, 394, 402, 406, 480, 484,
492, 496, 542, 546, 554, 558, 604,
608, 616, 620, 666, 670, 678, 682,
728, 732, 740, 744, 790, 794, 802, 806

30
Change History

v0.1
General: First public release .............. 1

v0.1a
General: Fixed bug in success/failure code (missing final n) .............. 18
Fixed bugs in success/failure code with regex...nnNTF .............. 16
v0.1b
General: Fixed bugs in the ‘bibmacro’ functions ........................................ 1
Polished code and documentation .................................................. 1
v0.1c
General: Replaced \msg_term:x with \msg_term:n ................................. 1
General: Fix for removed function ................................................. 1
Replaced obsolete command \prg_case_str:onn ............................... 1
General: Fixed wrong function name in \xpatchbibmacro .................. 20
v0.2
General: Additional biblatex related macros ............................... 1, 20
v0.2a
General: Fixed silly mistake ...................................................... 1, 20
v0.2b
General: Fix for removed function ................................................. 1
v0.2c
General: Fixed wrong function name in \xpatchbibmacro .................. 20
v0.2d
General: Fixed silly typo ......................................................... 1