

The `regexpatch` package*

Replacing `etoolbox` patching commands

Enrico Gregorio[†]

Released 2020/10/06

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 {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 0.2e, last revised 2020/10/06.

[†]E-mail: Enrico DOT Gregorio AT univr DOT it

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{!}{-}{}{}
```

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

A similar situation happens if `\xyz` has been defined by

```
\DeclareRobustCommand{\xyz}{something}
```

A `\show\xyz` instruction 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 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 of these tricks, so this package comes to the rescue.

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

The `regexpatch` package uses many features of the \LaTeX 3 experimental packages, in particular of `l3regex`. This has a clear advantage: we can have a `*`-variant of `\xpatchcmd` that does a "replace all" which can avoid multiple uses of `\patchcmd` on the same macro. Moreover there's a very powerful `\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 \LaTeX kernel defines `\strip@pt`:

```

\begingroup
  \catcode'P=12
  \catcode'T=12
  \lowercase{
    \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}
  \expandafter\endgroup\x
\def\strip@pt{\expandafter\rem@pt\the}

```

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}{\c0p\c0t}{\c0}
\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:

```

\def\rem@pt#1.##2pt{##1\ifnum##2>\z@.##2\fi}
\xpatchparametertext{\rem@pt}{pt}{\c0 p \c0 t}{\c0}

```

Of course, reading the manual of `l3regex` is necessary for being able to exploit the full power of `\regexpatchcmd` or `\xpatchparametertext`; in this case, ‘`\c0 p`’ (the space in between is optional) specifies a character ‘p’ with category code ‘other’. Actually neither the `\c0` 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 L^AT_EX3 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 L^AT_EX kernel macros `\newcommand`, `\renewcommand`, `\providecommand`, but also `\newenvironment` and `\renewenvironment`;
- the L^AT_EX kernel macro for defining robust commands `\DeclareRobustCommand`;
- the `etoolbox` macros `\newrobustcmd`, `\renewrobustcmd`, `\providrobustcmd`.

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][x]{#1 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}` arguments also with `\xpretobibdriver` and `\xapptobibdriver`.

Under the same philosophy, one can use the macros

- \xpatchfieldformat,
 \xpretofieldformat,
 \xapptofieldformat,
- \xpatchnameformat,
 \xpretonameformat,
 \xapptonameformat,
- \xpatchlistformat,
 \xpretonameformat,
 \xapptonameformat,
- \xpatchindexfieldformat,
 \xpretoindexfieldformat,
 \xapptoindexfieldformat,
- \xpatchindexnameformat,
 \xpretoindexnameformat,
 \xapptoindexnameformat,

- `\xpatchindexlistformat`,
- `\xpretoindexlistformat`,
- `\xapptoindexlistformat`,

for the biblatex internal macro defined respectively with

```
\DeclareFieldFormat, \DeclareNameFormat, \DeclareListFormat,
\DeclareIndexFieldFormat, \DeclareIndexNameFormat, \DeclareIndexListFormat.
```

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

Finally, the package defines the commands

- `\xshowcmd`
- `\xshowbibmacro`
- `\xshowbibdriver`
- `\xshowfieldformat`
- `\xshownameformat`
- `\xshowlistformat`
- `\xshowindexfieldformat`
- `\xshowindexnameformat`
- `\xshowindexlistformat`
- `\tracingxpatches`

The first three are the analog of `\show` to see the “real” definition of a macro, be it defined with optional arguments or as a robust command; the `bib` ones are for the corresponding biblatex macros. The last one takes an optional argument for activating and deactivating the tracing system. So

```
\tracingxpatches
```

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

```
\tracingxpatches[0]
```

will stop issuing messages.

5 Syntax

Here is the formal syntax of the commands.

```
\xpatchcmd{<command>}{<search>}{<replace>}{<success>}{<failure>}
\xpreto cmd{<command>}{<prepend>}{<success>}{<failure>}
\xapptocmd{<command>}{<append>}{<success>}{<failure>}
\xpatchbibmacro{<name>}{<search>}{<replace>}{<success>}{<failure>}
\xpreto bibmacro{<name>}{<prepend>}{<success>}{<failure>}
\xapptobibmacro{<name>}{<append>}{<success>}{<failure>}
```

```

\patchbibdriver{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretobibdriver{<name>}{<prepend>}{<success>}{<failure>}
\apptobibdriver{<name>}{<append>}{<success>}{<failure>}

\patchfieldformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretofieldformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptofieldformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\patchnameformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretonameformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptonameformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\patchlistformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretolistformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptolistformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\patchindexfieldformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretoindexfieldformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptoindexfieldformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\patchindexnameformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretoindexnameformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptoindexnameformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\patchindexlistformat[<entrytype>]{<name>}{<search>}{<replace>}{<success>}{<failure>}
\pretoindexlistformat[<entrytype>]{<name>}{<prepend>}{<success>}{<failure>}
\apptoindexlistformat[<entrytype>]{<name>}{<append>}{<success>}{<failure>}

\showcmd[*]{<command>}
\showbibname{<name>}
\showbibdriver{<name>}

\patchoptarg{<name>}{<replace>}
\patchparametertext{<name>}{<search>}{<replace>}{<success>}{<failure>}
\checkifpatchable{<name>}
\tracingxpatches[<number>]

```

Here $\langle command \rangle$ is the command's name (with the backslash), while $\langle name \rangle$ is the string that appears as the argument to `\newbibmacro`, `\DeclareBibliographyDriver`, `\DeclareFieldFormat`, `\DeclareNameFormat`, `\DeclareListFormat`, `\DeclareIndexFieldFormat`, `\DeclareIndexNameFormat` or `\DeclareIndexListFormat` respectively; $\langle search \rangle$, $\langle replace \rangle$, $\langle prepend \rangle$ and $\langle append \rangle$ are the list of tokens that are to be used for the specific tasks; $\langle success \rangle$ and $\langle failure \rangle$ are token lists to be executed if the patching succeeds or fails respectively. I find it useful to use `\ddt` as $\langle failure \rangle$, so that T_EX will stop for the undefined control sequence when the patching fails.

All the `\x...format` macros have an optional argument that by default is `*`.

In the commands whose name contains the string `regex`, both $\langle search \rangle$ and $\langle replace \rangle$ are understood to represent regular expressions. Check with the documentation of `l3regex` for details.

The `*`-variants of the `patch` type commands means that the replacement is performed on *all* matches. With `\showcmd* $\langle foo \rangle$` one gets all information on $\langle foo \rangle$, as if the tracing system were activated, including the default optional argument, if existent. So it's best to use it before trying `\patchoptarg` (and all the other commands, of course).

A curiosity about optional arguments: if one defines

```
\newcommand{\foo}[1][\bar]{-#1-}
```

then the braces around `bar` are stripped off. So with

```
\newcommand{\foo}[1][{\itshape 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}{\itshape 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 `\mathcode` in `\@addligto` into `\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 `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 `\endhead` material, which is delimited by `\endfirsthead` and `\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
\regexpatchcmd{\rubric}
  { \c{endfirsthead} .* \c{endhead} }
  { \c{endfirsthead} }{}{}
\makeatother

```

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

```

\makeatletter
\usepackage{etoolbox} % for \ifdef
\ifdef{\H@old@part}
  {
    \regexpatchcmd{\H@old@part}
      {$} % regex representing the end
      {\c{gdef}\c{cont@name@part}\cB\{\cP\#2\cE\}} % replacement
      {}{}%
  }
  {
    \regexpatchcmd{\@part}
      {$}
      {\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, distinguishing 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 `\ifdefined` 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\endgroup
\expandafter\def\expandafter\extractnumber\string\lecture#1;{#1}

```

Now with `\xpatchparametertext` one can do

```

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

```

recalling that the substitution performed by `l3regex` uses category code 12 characters by default. The command can be generalized to accept any (fixed) prefix:

```
\def\setupfilename#1{%
  \def\filename{\expandafter\extractnumber\jobname;}%
  \def\extractnumber#1##1;{##1}%
  \xpatchparametertext\extractnumber{#1}{#1}{-}{-}
\setupfilename{lecture}
```

where the prefix is passed to `\setupfilename` and the macro to use is `\filename`.

A proper L^AT_EX3 definition might be

```
\NewDocumentCommand{\setupfilename}{ m }
{
  \group_begin:
  \cs_set:Npx \filename_aux:
  {
    \group_end:
    \cs_set:Npn \exp_not:N \filename_extract:w
      \tl_to_str:n { #1 } #####1 ; { #####1 }
  }
  \filename_aux:
}
\NewDocumentCommand{\filename} {}
{
  \exp_after:wN \filename_extract:w \c_job_name_tl ;
}
```

7 The implementation of `regexpatch`

The usual starting stuff.

```
1 \ProvidesExplPackage
2   {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}
3 \@ifpackagelater { expl3 } { 2012/01/19 }
4   { }
5   {
6     \PackageError { regexpatch } { Support~package~l3kernel~too~old. }
7     {
8       Please~install~an~up~to~date~version~of~l3kernel~
9       using~your~TeX~package~manager~or~from~CTAN.\. \. \.
10      Loading~regexpatch~will~abort!
11    }
12    \tex_endinput:D
13  }
14 \RequirePackage{xparse}
```

7.1 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 `\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.

```

15 \bool_new:N \l_xpatch_protect_bool
16 \bool_new:N \l_xpatch_optional_bool
17 \bool_new:N \l_xpatch_patchable_bool
18 \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.

```

19 \tl_new:N \l_xpatch_name_tl
20 \tl_new:N \l_xpatch_repl_tl
21 \tl_new:N \l_xpatch_prefix_tl
22 \tl_new:N \l_xpatch_arg_tl
23 \tl_new:N \l_xpatch_replacement_tl
24 \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.

```

25 \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_\xaa_\u+
%+\protect_\xab_\u+
%+\@protected@testopt_\xac_\u\{xac_\u\}+
%+\@testopt_\xad_\u\{xad_\u\}+
%+\x@protect_\1\protect_\1_\u+
%+\x@protect_\2\protect_\2_\u+
%+\@protected@testopt_\3\3_\u\{3_\u\}+
%+\@testopt_\4_\u\{4_\u\}+
%

```

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. 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.

```

26 \cs_new_protected:Npn \xpatch_main_check:N #1
27 {
28   \bool_set_false:N \l_xpatch_protect_bool
29   \bool_set_false:N \l_xpatch_optional_bool
30   \tl_set:Nx \l_xpatch_name_tl { \cs_to_str:N #1 }
31   \tl_set:Nx \l_xpatch_repl_tl { \token_get_replacement_spec:N #1 }
32   \tl_clear:N \l_xpatch_type_tl
33   \regex_match:nVT % \DeclareRobustCommand<control word>
34     {^\\protect\ \\u{l_xpatch_name_tl}\ }
35     \l_xpatch_repl_tl
36     {
37       \bool_set_true:N \l_xpatch_protect_bool
38       \tl_put_right:Nx \l_xpatch_name_tl { \c_space_tl }
39       \tl_set:Nn \l_xpatch_type_tl { DRCw }
40     }
41   \regex_match:nVT % \DeclareRobustCommand<control symbol>
42     {^\\x@protect\ \\u{l_xpatch_name_tl}\\}
43     \l_xpatch_repl_tl
44     {
45       \bool_set_true:N \l_xpatch_protect_bool
46       \tl_put_right:Nx \l_xpatch_name_tl { \c_space_tl }
47       \tl_set:Nn \l_xpatch_type_tl { DRCs }
48     }
49   \regex_match:nVT % \newcommand<control word> with opt arg
50     {^\\@protected@testopt\ \\u{l_xpatch_name_tl}\\}
51     \l_xpatch_repl_tl
52     {
53       \bool_set_true:N \l_xpatch_optional_bool
54       \tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
55       \tl_set:Nn \l_xpatch_type_tl { ncw+o }
56     }
57   \regex_match:nVT % \newcommand<control symbol> with opt arg
58     {^\\@protected@testopt\ \\u{l_xpatch_name_tl}\\}
59     \l_xpatch_repl_tl
60     {
61       \bool_set_true:N \l_xpatch_optional_bool
62       \tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
63       \tl_set:Nn \l_xpatch_type_tl { ncs+o }
64     }
65   \regex_match:nVT % \newrobustcmd<any cs> with opt arg
66     {^\\@testopt\ \\u{l_xpatch_name_tl}}
67     \l_xpatch_repl_tl

```

```

68     {
69       \bool_set_true:N \l_xpatch_optional_bool
70       \tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
71       \tl_set:Nn \l_xpatch_type_tl { nrc+o }
72     }
73 \bool_if:NT \l_xpatch_protect_bool
74 {
75   \tl_set:Nx \l_xpatch_repl_tl
76     { \exp_after:wN \token_get_replacement_spec:N
77       \cs:w \l_xpatch_name_tl \cs_end: }
78   \regex_match:nVT % \DeclareRobustCommand<any cs> with opt arg
79     {^\\@protected@testopt\ \\u{l_xpatch_name_tl}\ \\}
80   \l_xpatch_repl_tl
81   {
82     \bool_set_true:N \l_xpatch_optional_bool
83     \tl_put_left:Nx \l_xpatch_name_tl { \c_backslash_str }
84     \tl_put_right:Nn \l_xpatch_type_tl { +o }
85   }
86 }
87 \bool_if:NT \l_xpatch_optional_bool
88 {
89   \regex_replace_once:nnN { .*? \{ (.*?) \} \Z } { \1 }
90   \l_xpatch_repl_tl
91 }
92 }

```

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 `appto` macros, and a `zero` function for the `show` macros. We also define the variants taking a name as their second argument.

```

93 \cs_new_protected:Npn \xpatch_main_four:NNnnnn #1 #2 #3 #4 #5 #6
94 {
95   \cs_if_exist:NTF #2
96   {
97     \xpatch_main_check:N #2
98     \bool_if:NT \l_xpatch_tracing_bool
99     { \xpatch_message_cstype:N #2 }
100    \exp_after:wN #1 \cs:w \l_xpatch_name_tl \cs_end: {#3}{#4}{#5}{#6}
101  }
102  {
103    \iow_term:n
104    {
105      xpatch-message: ~
106      '\token_to_str:N #2'~is~undefined;~
107      I'll~ignore~the~request.
108    }
109  }
110 }
111 \cs_new_protected:Npn \xpatch_main_three:NNnnn #1 #2 #3 #4 #5
112 {
113   \xpatch_main_four:NNnnnn #1 #2 { #3 } { #4 } { #5 } { }

```

```

114 }
115 \cs_new_protected:Npn \xpatch_main_zero:NN #1 #2
116 {
117   \xpatch_main_four:NNnnnn #1 #2 { } { } { } { }
118 }
119 \cs_generate_variant:Nn \xpatch_main_zero:NN {Nc}
120 \cs_generate_variant:Nn \xpatch_main_three:NNnnn {Nc}
121 \cs_generate_variant:Nn \xpatch_main_four:NNnnnn {Nc}

```

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.

```

122 \cs_new_protected:Npn \xpatch_get_all:N #1
123 {
124   \tl_set:Nf \l_xpatch_prefix_tl { \token_get_prefix_spec:N #1 }
125   \tl_set_rescan:Nnx \l_xpatch_prefix_tl { } \l_xpatch_prefix_tl
126   \tl_set:Nf \l_xpatch_arg_tl { \token_get_arg_spec:N #1 }
127   \tl_set_rescan:Nnx \l_xpatch_arg_tl { } \l_xpatch_arg_tl
128   \tl_set:Nf \l_xpatch_replacement_tl { \token_get_replacement_spec:N #1 }
129   \tl_set_rescan:Nnx \l_xpatch_replacement_tl { } \l_xpatch_replacement_tl
130 }

```

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 L^AT_EX3 that recommend never using `:D` functions.

```

131 \cs_new_protected:Npn \xpatch_rebuild:N #1
132 {
133   \use:x
134   {
135     \exp_not:V \l_xpatch_prefix_tl
136     \tex_def:D % unavoidable
137     \exp_not:N #1
138     \exp_not:V \l_xpatch_arg_tl
139     { \exp_not:V \l_xpatch_replacement_tl }
140   }
141 }

```

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.

```

142 \cs_new_protected:Npn \xpatch_check_patchable:N #1
143 {
144   \cs_if_exist:NTF #1
145   {
146     \xpatch_get_all:N #1
147     \xpatch_rebuild:N \xpatch_tmpa:w
148     \cs_if_eq:NNTF #1 \xpatch_tmpa:w
149     {
150       \bool_set_true:N \l_xpatch_patchable_bool
151       \xpatch_message:n
152       {
153         Macro ‘\token_to_str:N #1’~is~patchable
154       }
155     }
156   }

```

```

155     }
156     {
157     \bool_set_false:N \l_xpatch_patchable_bool
158     \xpatch_message:n
159     {
160     Macro '\token_to_str:N #1'~is~NOT~patchable\\
161     (Check~if~it~contains~'@'~commands)
162     }
163     }
164   }
165   {
166   \bool_set_false:N \l_xpatch_patchable_bool
167   \xpatch_message:n
168   {
169   Macro '\token_to_str:N #1'~doesn't~exist.
170   }
171   }
172 }

```

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 *success* code. If the patch isn't possible we just execute the *failure* code.

```

173 \cs_new_protected:Npn \xpatch_pretocmd:Nnnn #1 #2 #3 #4
174 {
175   \xpatch_check_patchable:N #1
176   \bool_if:NTF \l_xpatch_patchable_bool
177   {
178     \tl_put_left:Nn \l_xpatch_replacement_tl { #2 }
179     \xpatch_rebuild:N #1
180     #3
181   }
182   {
183     #4
184   }
185 }
186 \cs_new_protected:Npn \xpatch_apptocmd:Nnnn #1 #2 #3 #4
187 {
188   \xpatch_check_patchable:N #1
189   \bool_if:NTF \l_xpatch_patchable_bool
190   {
191     \tl_put_right:Nn \l_xpatch_replacement_tl { #2 }
192     \xpatch_rebuild:N #1
193     #3
194   }
195   {
196     #4
197   }
198 }

```

Substituting tokens in the replacement text is a bit harder, but not conceptually different. 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 `\regex_replace_all:nnNTF` and `\regex_replace_once:nnNTF`

in order to pass correctly the success or failure arguments.

```
199 \cs_new_protected:Npn \xpatch_regexpatchcmd_all:Nnnnn #1 #2 #3 #4 #5
200 {
201   \xpatch_check_patchable:N #1
202   \bool_if:NTF \l_xpatch_patchable_bool
203     {
204       \regex_replace_all:nnNTF { #2 } { #3 } \l_xpatch_replacement_tl
205       { \xpatch_rebuild:N #1 #4 }
206       { #5 }
207     }
208     {
209       #5
210     }
211 }
212 \cs_new_protected:Npn \xpatch_regexpatchcmd_once:Nnnnn #1 #2 #3 #4 #5
213 {
214   \xpatch_check_patchable:N #1
215   \bool_if:NTF \l_xpatch_patchable_bool
216     {
217       \regex_replace_once:nnNTF { #2 } { #3 } \l_xpatch_replacement_tl
218       { \xpatch_rebuild:N #1 #4 }
219       { #5 }
220     }
221     {
222       #5
223     }
224 }
```

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

```
225 \cs_new_protected:Npn \xpatch_patchcmd_once:Nnnnn #1 #2 #3 #4 #5
226 {
227   \xpatch_check_patchable:N #1
228   \bool_if:NTF \l_xpatch_patchable_bool
229     {
230       \tl_set:Nn \l_tmpa_tl { #2 }
231       \tl_set:Nn \l_tmpb_tl { #3 }
232       \regex_replace_once:nnNTF
233         { \u{1_tmpa_tl} }
234         { \u{1_tmpb_tl} }
235         \l_xpatch_replacement_tl
236         { \xpatch_rebuild:N #1 #4 }
237         { #5 }
238     }
239     {
240       #5
241     }
242 }
243 \cs_new_protected:Npn \xpatch_patchcmd_all:Nnnnn #1 #2 #3 #4 #5
244 {
245   \xpatch_check_patchable:N #1
246   \bool_if:NTF \l_xpatch_patchable_bool
247     {
248       \tl_set:Nn \l_tmpa_tl { #2 }
```



```

249     \tl_set:Nn \l_tmpb_tl { #3 }
250     \regex_replace_all:nnNTF
251       { \u{l_tmpa_tl} }
252       { \u{l_tmpb_tl} }
253     \l_xpatch_replacement_tl
254     { \xpatch_rebuild:N #1 #4 }
255     { #5 }
256   }
257   {
258     #5
259   }
260 }

```

Now the tracing system.

```

261 \cs_new_protected:Npn \xpatch_message:n #1
262 {
263   \bool_if:NT \l_xpatch_tracing_bool
264   {
265     \iow_term:n { xpatch-message: ~ #1 }
266   }
267 }
268 \cs_new:Npn \xpatch_message_cstype:N #1
269 {
270   \str_case:onF { \l_xpatch_type_tl }
271   {
272     { DRCw } {
273       \xpatch_message:n
274       {
275         '\token_to_str:N #1'~is~a~control~word~defined~
276         with~'\token_to_str:N \DeclareRobustCommand
277       }
278     }
279     { DRCw+o } {
280       \xpatch_message:n
281       {
282         '\token_to_str:N #1'~is~a~control~word~defined~
283         with~'\token_to_str:N \DeclareRobustCommand'~
284         and~a~default~optional~argument~'\l_xpatch_repl_tl'
285       }
286     }
287     { DRCs } {
288       \xpatch_message:n
289       {
290         '\token_to_str:N #1'~is~a~control~symbol~defined~
291         with~'\token_to_str:N \DeclareRobustCommand'
292       }
293     }
294     { DRCs+o } {
295       \xpatch_message:n
296       {
297         '\token_to_str:N #1'~is~a~control~symbol~defined~
298         with~'\token_to_str:N \DeclareRobustCommand'~
299         and~a~default~optional~argument~'\l_xpatch_repl_tl'
300       }
301     }

```

```

302 { ncw+o } {
303     \xpatch_message:n
304     {
305         '\token_to_str:N #1'~is-a~control~word~defined~
306         with~'\token_to_str:N \newcommand'~
307         and-a~default~optional~argument~'\l_xpatch_repl_tl'
308     }
309 }
310 { ncs+o } {
311     \xpatch_message:n
312     {
313         '\token_to_str:N #1'~is-a~control~symbol~defined~
314         with~'\token_to_str:N \newcommand'~
315         and-a~default~optional~argument~'\l_xpatch_repl_tl'
316     }
317 }
318 { nrc+o } {
319     \xpatch_message:n
320     {
321         '\token_to_str:N #1'~is-a~control~sequence~defined~
322         with~'\token_to_str:N \newrobustcmd'~
323         and-a~default~optional~argument~'\l_xpatch_repl_tl'
324     }
325 }
326 }
327 {
328     \xpatch_message:n
329     {
330         '\token_to_str:N #1'~is~not~especially~defined
331     }
332 }
333 }

```

7.3 The user level functions

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

```

334 \NewDocumentCommand{\xshowcmd} { s +m }
335 {
336     \IfBooleanT{#1}
337     {
338         \group_begin:
339         \bool_set_true:N \l_xpatch_tracing_bool
340     }
341     \xpatch_main_zero:NN \cs_show:N #2
342     \IfBooleanT{#1}
343     {
344         \group_end:
345     }
346 }
347 \NewDocumentCommand{\xpretocmd}{ +m +m +m +m }
348 { \xpatch_main_three:NNnnn \xpatch_pretocmd:Nnnn #1 {#2} {#3} {#4} }

```

```

349 \NewDocumentCommand{\xapptocmd}{ +m +m +m +m }
350 { \xpatch_main_three:NNnnn \xpatch_apptocmd:Nnnn #1 {#2} {#3} {#4} }
351 \NewDocumentCommand{\regxpathcmd}{ s +m +m +m +m }
352 {
353   \IfBooleanTF{#1}
354   { \xpatch_main_four:NNnnnn \xpatch_regxpathcmd_all:Nnnnn #2 {#3}{#4}{#5}{#6} }
355   { \xpatch_main_four:NNnnnn \xpatch_regxpathcmd_once:Nnnnn #2 {#3}{#4}{#5}{#6} }
356 }
357 \NewDocumentCommand{\xpatchcmd}{ s +m +m +m +m }
358 {
359   \IfBooleanTF{#1}
360   { \xpatch_main_four:NNnnnn \xpatch_patchcmd_all:Nnnnn #2 {#3}{#4}{#5}{#6} }
361   { \xpatch_main_four:NNnnnn \xpatch_patchcmd_once:Nnnnn #2 {#3}{#4}{#5}{#6} }
362 }

```

The functions for patching biblatex related macros that are given by name and we'll use the already defined variants.

```

363 \NewDocumentCommand{\xshowbibmacro} { s +m }
364 {
365   \IfBooleanT{#1}
366   {
367     \group_begin:
368     \bool_set_true:N \l_xpatch_tracing_bool
369   }
370   \xpatch_main_zero:Nc \cs_show:N { abx@macro@ \tl_to_str:n {#2} }
371   \IfBooleanT{#1}
372   {
373     \group_end:
374   }
375 }
376 \NewDocumentCommand{\xpretobibmacro} { +m +m +m +m }
377 {
378   \xpatch_main_three:Ncnnn \xpatch_pretocmd:Nnnn
379   { abx@macro@ \tl_to_str:n {#1} } {#2}{#3}{#4}
380 }
381 \NewDocumentCommand{\xapptobibmacro} { +m +m +m +m }
382 {
383   \xpatch_main_three:Ncnnn \xpatch_apptocmd:Nnnn
384   { abx@macro@ \tl_to_str:n {#1} } {#2}{#3}{#4}
385 }
386 \NewDocumentCommand{\xpatchbibmacro} { s +m +m +m +m }
387 {
388   \IfBooleanTF{#1}
389   {
390     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
391     { abx@macro@ \tl_to_str:n {#2} } {#3}{#4}{#5}{#6}
392   }
393   {
394     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
395     { abx@macro@ \tl_to_str:n {#2} } {#3}{#4}{#5}{#6}
396   }
397 }
398 \NewDocumentCommand{\regxpathbibmacro} { s +m +m +m +m }
399 {

```

```

400 \IfBooleanTF{#1}
401 {
402   \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_all:Nnnnn
403   { abx@macro@ \tl_to_str:n {#2} } {#3}{#4}{#5}{#6}
404 }
405 {
406   \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_once:Nnnnn
407   { abx@macro@ \tl_to_str:n {#2} } {#3}{#4}{#5}{#6}
408 }
409 }
410 \NewDocumentCommand{\xshowbibdriver} { s +m }
411 {
412   \IfBooleanT{#1}
413   {
414     \group_begin:
415     \bool_set_true:N \l_xpatch_tracing_bool
416   }
417   \xpatch_main_zero:Nc \cs_show:N { blx@bbx@#2 }
418   \IfBooleanT{#1}
419   {
420     \group_end:
421   }
422 }
423 \NewDocumentCommand{\xpretobibdriver} { +m +m +m +m }
424 { \exp_args:Nc \xpatch_pretocmd:Nnnn {blx@bbx@#1} {#2}{#3}{#4} }
425 \NewDocumentCommand{\xapptobibdriver} { +m +m +m +m }
426 { \exp_args:Nc \xpatch_apptocmd:Nnnn {blx@bbx@#1} {#2}{#3}{#4} }
427 \NewDocumentCommand{\xpatchbibdriver} { s +m +m +m +m }
428 {
429   \IfBooleanTF{#1}
430   { \exp_args:Nc \xpatch_patchcmd_all:Nnnnn {blx@bbx@#2} {#3}{#4}{#5}{#6} }
431   { \exp_args:Nc \xpatch_patchcmd_once:Nnnnn {blx@bbx@#2} {#3}{#4}{#5}{#6} }
432 }
433 \NewDocumentCommand{\regexpatchbibdriver} { s +m +m +m +m }
434 {
435   \IfBooleanTF{#1}
436   { \exp_args:Nc \xpatch_regexpatchcmd_all:Nnnnn {blx@bbx@#2} {#3}{#4}{#5}{#6} }
437   { \exp_args:Nc \xpatch_regexpatchcmd_once:Nnnnn {blx@bbx@#2} {#3}{#4}{#5}{#6} }
438 }

```

Other biblatex related macros, added by request of the maintainers.

```

439 \NewDocumentCommand{\xshowfieldformat} { s O{*} +m }
440 {
441   \IfBooleanT{#1}
442   {
443     \group_begin:
444     \bool_set_true:N \l_xpatch_tracing_bool
445   }
446   \xpatch_main_zero:Nc \cs_show:N { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} }
447   \IfBooleanT{#1}
448   {
449     \group_end:
450   }
451 }
452 \NewDocumentCommand{\xpretofieldformat} { s O{*} +m +m +m +m }

```

```

453 {
454   \IfBooleanTF{#1}
455   {
456     \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
457     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
458   }
459   {
460     \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
461     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
462   }
463 }
464 \NewDocumentCommand{\xapptofieldformat} { s O{*} +m +m +m +m }
465 {
466   \IfBooleanTF{#1}
467   {
468     \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
469     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
470   }
471   {
472     \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
473     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
474   }
475 }
476 \NewDocumentCommand{\xpatchfieldformat} { s O{*} +m +m +m +m +m }
477 {
478   \IfBooleanTF{#1}
479   {
480     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
481     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
482   }
483   {
484     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
485     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
486   }
487 }
488 \NewDocumentCommand{\regexpatchfieldformat} { s O{*} +m +m +m +m +m }
489 {
490   \IfBooleanTF{#1}
491   {
492     \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_all:Nnnnn
493     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
494   }
495   {
496     \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_once:Nnnnn
497     { abx@ffd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
498   }
499 }
500
501 \NewDocumentCommand{\xshownameformat} { s O{*} +m }
502 {
503   \IfBooleanT{#1}
504   {
505     \group_begin:
506     \bool_set_true:N \l_xpatch_tracing_bool

```

```

507 }
508 \xpatch_main_zero:Nc \cs_show:N { abx@nfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2} }
509 \IfBooleanTF{#1}
510 {
511   \group_end:
512 }
513 }
514 \NewDocumentCommand{\xpretonameformat} { s O{*} +m +m +m +m }
515 {
516   \IfBooleanTF{#1}
517   {
518     \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
519     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
520   }
521   {
522     \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
523     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
524   }
525 }
526 \NewDocumentCommand{\xapptonameformat} { s O{*} +m +m +m +m }
527 {
528   \IfBooleanTF{#1}
529   {
530     \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
531     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
532   }
533   {
534     \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
535     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
536   }
537 }
538 \NewDocumentCommand{\xpatchnameformat} { s O{*} +m +m +m +m +m }
539 {
540   \IfBooleanTF{#1}
541   {
542     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
543     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
544   }
545   {
546     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
547     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
548   }
549 }
550 \NewDocumentCommand{\regxpathnameformat} { s O{*} +m +m +m +m +m }
551 {
552   \IfBooleanTF{#1}
553   {
554     \xpatch_main_four:Ncnnnn \xpatch_regxpathcmd_all:Nnnnn
555     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
556   }
557   {
558     \xpatch_main_four:Ncnnnn \xpatch_regxpathcmd_once:Nnnnn
559     { abx@nfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
560   }

```

```

561 }
562
563 \NewDocumentCommand{\xshowlistformat} { s O{*} +m }
564 {
565   \IfBooleanT{#1}
566   {
567     \group_begin:
568     \bool_set_true:N \l_xpatch_tracing_bool
569   }
570   \xpatch_main_zero:Nc \cs_show:N { abx@lfd@ \tl_to_str:n {#1} @ \tl_to_str:n {#2} }
571   \IfBooleanT{#1}
572   {
573     \group_end:
574   }
575 }
576 \NewDocumentCommand{\xpretolistformat} { s O{*} +m +m +m +m }
577 {
578   \IfBooleanTF{#1}
579   {
580     \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
581     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
582   }
583   {
584     \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
585     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
586   }
587 }
588 \NewDocumentCommand{\xapptolistformat} { s O{*} +m +m +m +m }
589 {
590   \IfBooleanTF{#1}
591   {
592     \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
593     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
594   }
595   {
596     \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
597     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
598   }
599 }
600 \NewDocumentCommand{\xpatchlistformat} { s O{*} +m +m +m +m +m }
601 {
602   \IfBooleanTF{#1}
603   {
604     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
605     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
606   }
607   {
608     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
609     { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
610   }
611 }
612 \NewDocumentCommand{\regexpatchlistformat} { s O{*} +m +m +m +m +m }
613 {
614   \IfBooleanTF{#1}

```

```

615 {
616   \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_all:Nnnnn
617   { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
618 }
619 {
620   \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_once:Nnnnn
621   { abx@lfd@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
622 }
623 }
624
625 \NewDocumentCommand{\xshowindexfieldformat} { s O{*} +m }
626 {
627   \IfBooleanT{#1}
628   {
629     \group_begin:
630     \bool_set_true:N \l_xpatch_tracing_bool
631   }
632   \xpatch_main_zero:Nc \cs_show:N { abx@fid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2} }
633   \IfBooleanT{#1}
634   {
635     \group_end:
636   }
637 }
638 \NewDocumentCommand{\xpretaindexfieldformat} { s O{*} +m +m +m +m }
639 {
640   \IfBooleanTF{#1}
641   {
642     \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
643     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
644   }
645   {
646     \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
647     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
648   }
649 }
650 \NewDocumentCommand{\xapptaindexfieldformat} { s O{*} +m +m +m +m }
651 {
652   \IfBooleanTF{#1}
653   {
654     \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
655     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
656   }
657   {
658     \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
659     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
660   }
661 }
662 \NewDocumentCommand{\xpatchindexfieldformat} { s O{*} +m +m +m +m +m }
663 {
664   \IfBooleanTF{#1}
665   {
666     \xpatch_main_four:Ncnynn \xpatch_patchcmd_all:Nnnnn
667     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
668   }

```



```

669 {
670   \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
671   { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
672 }
673 }
674 \NewDocumentCommand{\regxpathchindexfieldformat} { s O{*} +m +m +m +m +m }
675 {
676   \IfBooleanTF{#1}
677   {
678     \xpatch_main_four:Ncnnnn \xpatch_regxpathchcmd_all:Nnnnn
679     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
680   }
681   {
682     \xpatch_main_four:Ncnnnn \xpatch_regxpathchcmd_once:Nnnnn
683     { abx@fid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
684   }
685 }
686 }
687 \NewDocumentCommand{\xshowindexnameformat} { s O{*} +m }
688 {
689   \IfBooleanT{#1}
690   {
691     \group_begin:
692     \bool_set_true:N \l_xpatch_tracing_bool
693   }
694   \xpatch_main_zero:Nc \cs_show:N { abx@nid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2} }
695   \IfBooleanT{#1}
696   {
697     \group_end:
698   }
699 }
700 \NewDocumentCommand{\xpretointindexnameformat} { s O{*} +m +m +m +m }
701 {
702   \IfBooleanTF{#1}
703   {
704     \xpatch_main_three:Ncnnn \xpatch_pretocmd_all:Nnnn
705     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
706   }
707   {
708     \xpatch_main_three:Ncnnn \xpatch_pretocmd_once:Nnnn
709     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
710   }
711 }
712 \NewDocumentCommand{\xapptointindexnameformat} { s O{*} +m +m +m +m }
713 {
714   \IfBooleanTF{#1}
715   {
716     \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
717     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
718   }
719   {
720     \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
721     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
722   }

```

```

723 }
724 \NewDocumentCommand{\xpatchindexnameformat} { s O{*} +m +m +m +m +m }
725 {
726   \IfBooleanTF{#1}
727   {
728     \xpatch_main_four:Ncnynn \xpatch_patchcmd_all:Nnnnn
729     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
730   }
731   {
732     \xpatch_main_four:Ncnynn \xpatch_patchcmd_once:Nnnnn
733     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
734   }
735 }
736 \NewDocumentCommand{\regexpatchindexnameformat} { s O{*} +m +m +m +m +m }
737 {
738   \IfBooleanTF{#1}
739   {
740     \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_all:Nnnnn
741     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
742   }
743   {
744     \xpatch_main_four:Ncnynn \xpatch_regexpatchcmd_once:Nnnnn
745     { abx@nid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
746   }
747 }
748
749 \NewDocumentCommand{\xshowindexlistformat} { s O{*} +m }
750 {
751   \IfBooleanT{#1}
752   {
753     \group_begin:
754     \bool_set_true:N \l_xpatch_tracing_bool
755   }
756   \xpatch_main_zero:Nc \cs_show:N { abx@lid@ \tl_to_str:n {#1} @ \tl_to_str:n {#2} }
757   \IfBooleanT{#1}
758   {
759     \group_end:
760   }
761 }
762 \NewDocumentCommand{\xpretaindexlistformat} { s O{*} +m +m +m +m }
763 {
764   \IfBooleanTF{#1}
765   {
766     \xpatch_main_three:Ncnynn \xpatch_pretocmd_all:Nnnn
767     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
768   }
769   {
770     \xpatch_main_three:Ncnynn \xpatch_pretocmd_once:Nnnn
771     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
772   }
773 }
774 \NewDocumentCommand{\xapptoindexlistformat} { s O{*} +m +m +m +m }
775 {
776   \IfBooleanTF{#1}

```

```

777 {
778   \xpatch_main_three:Ncnnn \xpatch_apptocmd_all:Nnnn
779   { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
780 }
781 {
782   \xpatch_main_three:Ncnnn \xpatch_apptocmd_once:Nnnn
783   { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}
784 }
785 }
786 \NewDocumentCommand{\xpatchindexlistformat} { s O{*} +m +m +m +m +m }
787 {
788   \IfBooleanTF{#1}
789   {
790     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_all:Nnnnn
791     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
792   }
793   {
794     \xpatch_main_four:Ncnnnn \xpatch_patchcmd_once:Nnnnn
795     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
796   }
797 }
798 \NewDocumentCommand{\regexpatchindexlistformat} { s O{*} +m +m +m +m +m }
799 {
800   \IfBooleanTF{#1}
801   {
802     \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_all:Nnnnn
803     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
804   }
805   {
806     \xpatch_main_four:Ncnnnn \xpatch_regexpatchcmd_once:Nnnnn
807     { abx@lid@ \tl_to_str:n {#2} @ \tl_to_str:n {#3} } {#4}{#5}{#6}{#7}
808   }
809 }

```

A macro to check if the macro is patchable. It just prints a message on the terminal and in the log file.

```

810 \NewDocumentCommand{\checkpatchable}{ +m }
811 {
812   \group_begin:
813   \bool_set_true:N \l_xpatch_tracing_bool
814   \xpatch_check_patchable:N #1
815   \group_end:
816 }

```

The last user level command: a macro for changing the optional argument in a macro that has one.

```

817 \cs_generate_variant:Nn \xpatch_get_all:N {c}
818 \cs_generate_variant:Nn \xpatch_rebuild:N {c}
819 \NewDocumentCommand{\xpatchoptarg}{ +m +m }
820 {
821   \xpatch_main_check:N #1
822   \bool_if:NTF \l_xpatch_optional_bool
823   {

```

We have a macro with optional argument; so we strip off the first backslash from the name and proceed.

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

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

```
825 \tl_set:Nf \l_xpatch_prefix_tl { \token_get_prefix_spec:N #1 }
826 \tl_clear:N \l_xpatch_prefix_tl
827 \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 `\token_get_replacement_spec:N` because the spaces would be lost.

```
828 \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.

```
829 \tl_set:Nn \l_tmpa_tl { { #2 } }
830 \regex_replace_once:nnN { \cB. .* \cE. \Z} { \u{l_tmpa_tl} }
831 \l_xpatch_replacement_tl
```

Now we rebuild the control sequence.

```
832 \xpatch_rebuild:c { \l_xpatch_name_tl }
833 }
```

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

```
834 {
835 \group_begin:
836 \bool_set_true:N \l_xpatch_tracing_bool
837 \xpatch_message:n
838 {
839 Macro~'\token_to_str:N #1'~ has~no~optional~argument~
840 or~it~has~been~defined~with~'xparse'~and~operating~
841 on~such~commands~is~(still)~not~supported
842 }
843 \group_end:
844 }
845 }
```

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

```
846 \NewDocumentCommand{\tracingxpatches}{ 0{1} }
847 {
848 \int_compare:nTF { #1 > 0 }
849 { \bool_set_true:N \l_xpatch_tracing_bool }
850 { \bool_set_false:N \l_xpatch_tracing_bool }
851 }
```

One more thing: patching the parameter text!

```
852 \NewDocumentCommand{\xpatchparametertext}{ +m +m +m +m +m }
853 {
854 \xpatch_check_patchable:N #1
855 \bool_if:NTF \l_xpatch_patchable_bool
856 {
857 \regex_replace_once:nnN { #2 } { #3 } \l_xpatch_arg_tl
858 \xpatch_rebuild:N #1
```

```

859     #4
860   }
861   {
862     #5
863   }
864 }

```

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		D	
<code>\</code>	9, 34, 42, 50, 58, 66, 79, 105, 160, 265	<code>\DeclareRobustCommand</code>	33, 41, 78, 276, 283, 291, 298
<code>\{</code>	89		
<code>\}</code>	89		
Numbers		E	
<code>\1</code>	89	exp commands:	
<code>\sqcup</code>	34, 42, 50, 58, 66, 79	<code>\exp_after:wN</code>	76, 100
B		<code>\exp_args:Nc</code>	424, 426, 430, 431, 436, 437
bool commands:		<code>\exp_not:N</code>	137
<code>\bool_if:NTF</code>	73, 87, 98, 176, 189, 202, 215, 228, 246, 263, 822, 855	<code>\exp_not:n</code>	135, 138, 139
<code>\bool_new:N</code>	15, 16, 17, 18	<code>\ExplFileDate</code>	2
<code>\bool_set_false:N</code>	28, 29, 157, 166, 850	<code>\ExplFileDescription</code>	2
<code>\bool_set_true:N</code>	37, 45, 53, 61, 69, 82, 150, 339, 368, 415, 444, 506, 568, 630, 692, 754, 813, 836, 849	<code>\ExplFileName</code>	2
C		<code>\ExplFileVersion</code>	2
<code>\cB</code>	830	G	
<code>\cE</code>	830	group commands:	
<code>\checkpatchable</code>	810	<code>\group_begin:</code>	338, 367, 414, 443, 505, 567, 629, 691, 753, 812, 835
cs commands:		<code>\group_end:</code>	344, 373, 420, 449, 511, 573, 635, 697, 759, 815, 843
<code>\cs:w</code>	77, 100	I	
<code>\cs_end:</code>	77, 100	<code>\IfBooleanT</code>	336, 342, 365, 371, 412, 418, 441, 447, 503, 509, 565, 571, 627, 633, 689, 695, 751, 757
<code>\cs_generate_variant:Nn</code>	25, 119, 120, 121, 817, 818	<code>\IfBooleanTF</code>	353, 359, 388, 400, 429, 435, 454, 466, 478, 490, 516, 528, 540, 552, 578, 590, 602, 614, 640, 652, 664, 676, 702, 714, 726, 738, 764, 776, 788, 800
<code>\cs_if_eq:NNTF</code>	148	int commands:	
<code>\cs_if_exist:NTF</code>	95, 144	<code>\int_compare:nTF</code>	848
<code>\cs_new:Npn</code>	268	iow commands:	
<code>\cs_new_protected:Npn</code>	26, 93, 111, 115, 122, 131, 142, 173, 186, 199, 212, 225, 243, 261	<code>\iow_term:n</code>	103, 265
<code>\cs_show:N</code>	341, 370, 417, 446, 508, 570, 632, 694, 756	N	
<code>\cs_to_str:N</code>	30	<code>\newcommand</code>	49, 57, 306, 314
		<code>\NewDocumentCommand</code>	334, 347, 349, 351, 357, 363,

376, 381, 386, 398, 410, 423, 425, 427, 433, 439, 452, 464, 476, 488, 501, 514, 526, 538, 550, 563, 576, 588, 600, 612, 625, 638, 650, 662, 674, 687, 700, 712, 724, 736, 749, 762, 774, 786, 798, 810, 819, 846, 852	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
\newrobustcmd 65, 322	\l_tmpa_tl 230, 248, 829 \l_tmpb_tl 231, 249
P	token commands:
\PackageError 6	\token_get_arg_spec:N 126
\ProvidesExplPackage 1	\token_get_prefix_spec:N ... 124, 825
	\token_get_replacement_spec:N 31, 76, 128
R	\token_to_str:N 106, 153, 160, 169, 275, 276, 282, 283, 290, 291, 297, 298, 305, 306, 313, 314, 321, 322, 330, 839
regex commands:	\tracingxpatches 846
\regex_match:nnTF 25, 33, 41, 49, 57, 65, 78	
\regex_replace_all:nnNTF ... 204, 250	
\regex_replace_once:nnN . 89, 830, 857	
\regex_replace_once:nnNTF .. 217, 232	
\regexprpatchbibdriver 433	
\regexprpatchbibmacro 398	
\regexprpatchcmd 351	
\regexprpatchfieldformat 488	
\regexprpatchindexfieldformat 674	
\regexprpatchindexlistformat 798	
\regexprpatchindexnameformat 736	
\regexprpatchlistformat 612	
\regexprpatchnameformat 550	
\RequirePackage 14	
S	U
str commands:	\u 34, 42, 50, 58, 66, 79, 233, 234, 251, 252, 830
\c_backslash_str 54, 62, 70, 83	use commands:
\str_case:nnTF 270	\use:n 133
T	X
T _E X and L ^A T _E X 2 _ε commands:	\xapptobibdriver 425
\@ifpackagelater 3	\xapptobibmacro 381
tex commands:	\xapptocmd 349
\tex_def:D 136	\xapptofieldformat 464
\tex_endinput:D 12	\xapptoindexfieldformat 650
tl commands:	\xapptoindexlistformat 774
\c_space_tl 38, 46	\xapptoindexnameformat 712
\tl_clear:N 32, 826	\xapptolistformat 588
\tl_new:N 19, 20, 21, 22, 23, 24	\xapptonameformat 526
\tl_put_left:Nn .. 54, 62, 70, 83, 178	xpatch commands:
\tl_put_right:Nn 38, 46, 84, 191	\xpatch_apptocmd:Nnnn 186, 350, 383, 426
\tl_set:Nn 30, 31, 39, 47, 55, 63, 71, 75, 124, 126, 128, 230, 231, 248, 249, 824, 825, 829	\xpatch_apptocmd_all:Nnnn 468, 530, 592, 654, 716, 778
\tl_set_eq:NN 828	\xpatch_apptocmd_once:Nnnn 472, 534, 596, 658, 720, 782
\tl_set_rescan:Nnn . 125, 127, 129, 827	\l_xpatch_arg_tl 22, 126, 127, 138, 857
\tl_tail:n 824	\xpatch_check_patchable:N .. 142, 175, 188, 201, 214, 227, 245, 814, 854
\tl_to_str:n ... 370, 379, 384, 391, 395, 403, 407, 446, 457, 461, 469,	\xpatch_get_all:N 122, 146, 817
	\xpatch_main_check:N ... 26, 97, 821
	\xpatch_main_four:NNnnnn 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

v0.1b		v0.2a	
General: Fixed bugs in the ‘bibmacro’		General: Fixed silly mistake	1, 20
functions	1	v0.2b	
v0.1c		General: Fix for removed function	1
General: Replaced <code>\msg_term:x</code> with		v0.2c	
<code>\msg_term:n</code>	1	General: Fixed wrong function name	
Replaced obsolete command		in <code>\xpatchbibmacro</code>	20
<code>\prg_case_str:onm</code>	1	Removed loading of <code>l3regex</code>	1
v0.2		v0.2d	
General: Additional biblatex related		General: Fixed silly typo	1
macros	1, 20	v0.2e	
Polished code and documentation . . .	1	General: Replaced deprecated	
		commands	1