The `patchcmd` package

Michael J. Downes*

Version 1.05, 2016/06/03

1 Introduction

The `patchcmd` package provides a command `\patchcommand` that can be used to add material at the beginning and/or end of the replacement text of an existing macro. It works for macros with any number of normal arguments (0–9), including macros that were defined with `\DeclareRobustCommand`. However, it does not work for macros that use `\futurelet`, for example ones defined to have starred forms or optional arguments. In addition, it does not work for macros that have delimited arguments.

This package is the result of some discussion initiated by Peter Wilson in the newsgroup `comp.text.tex` in June 2000 and continued with Peter and Heiko Oberdiek.

```
Subject: Re: Adding stuff at end of a macro
References:
        <3ojiks8in1f5s575eta5rg7ncoc98mpi8f@4ax.com>
        <39492C50.A962193B@boeing.com>
        <pcitvabqk2.fsf@thor.ams.org>
```

2 Implementation

Standard declaration of package name and date.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{patchcmd}[2016/06/03 v1.05]
```

For debugging.

```
%%\def\wrs{\immediate\write\sixt@@n}
```

```
\newcommand{\patchcommand}[1]{%
  \expandafter\patchcmd@a\meaning#1??->@\@nil#1%
}
```

We begin by looking at the meaning string of the given token. We will issue an error message if the token is not a control sequence, if it is an undefined control sequence, or if it is a known control sequence but not a macro.

```
\long\def\patchcmd@a#1#2#3->#4#5\@nil#6{%
```

*Michael died in 2003

```
%%  \wrs{\string#6: [#1] [#2] [#3]->[#4]}%
  \ifx @#4\relax \patchcmdError#6#1%
    \expandafter\@gobbletwo % discard the other two arguments
  \else
    \if l#2\toks@{\patchcmd@e{}#6}% l in this position means \long
    \else \toks@{\patchcmd@e*#6}%   not \long
    \fi
```

Now `\patchcmd@b` will do further analysis to determine what kinds of arguments the macro takes. If it takes n normal arguments, the digit n (0–9) will be added to `\toks@`.

```
    \patchcmd@b #3@#4#5 ? ? ? \@nil#6%
    \expandafter\the\expandafter\toks@
  \fi
}
```

We handle robust commands by scanning the first two control words in the macro's definition to see if the second one is followed by two spaces. If it is, then arg 7 will be empty. In that case we compare the csname preceding the two spaces to the original argument of `\patchcommand`. If they are equal it is nearly certain that this is a robust command in normal LaTeX form. In that case we call `\patchcommand` quasi-recursively on the protected control sequence with the extra space at the end of its name.

```
\def\patchcmd@b#1:#2@#3#4 #5#6 #7 #8\@nil#9{%
%%  \wrs{[#1] [#2] [#3] [#4] [#5] [#6] ARG7=[#7] [#8]}%
  \if \ifx @#7@\expandafter
      \ifx\csname #6\endcsname#9T\else F\fi\else F\fi T%
    \toks@\expandafter{\expandafter\patchcommand\csname #6 \endcsname}%
  \else
    \ifx @#2@% No arguments
      \toks@\expandafter{\the\toks@ 0}%
    \else
      \patchcmd@c 0#2{\string##}0%
    \fi
  \fi
}
```

The task of `\patchcmd@c` is to iterate over `#D` pairs, where D is a digit in the range 1–9, until reaching the last one. Whenever we find such a pair, we carry the digit forward for the next recursive call; but we use digit 0 as a marker to terminate the recursion. If any other character interrupts the `#D` pattern, it means that this macro has some kind of delimited argument.

```
\def\patchcmd@c#1#2#3{%
  \if\string###2%      % yes it's a # token
    \ifodd 0#31 % and it's followed by a number
      \if 0#3\patchcmd@d#1\fi % number=0? then we're done
    \else \patchcmd@d D% # not a number: must be a delimited arg
    \fi
  \else \patchcmd@d D% not a # token: must be a delmited arg
  \fi
```

```
    \patchcmd@c#3%
  }

  \def\patchcmd@d#1{%
    \if D#1%
      \PackageError{patchcmd}{Cannot change a macro that has
        delimited arguments}\@ehd
    \else
      \toks@\expandafter{\the\toks@ #1}%
    \fi
    \begingroup
    \aftergroup\@gobble
    \let\patchcmd@c\endgroup
  }

  \def\patchcmd@e#1#2#3#4#5{%
    \begingroup
    \edef\@##1{%
      \@temptokena\noexpand\expandafter{%
        \noexpand#2%
          \ifnum#3>0 {####1}\ifnum#3>1 {####2}\ifnum#3>2 {####3}%
          \ifnum#3>3 {####4}\ifnum#3>4 {####5}\ifnum#3>5 {####6}%
          \ifnum#3>6 {####7}\ifnum#3>7 {####8}\ifnum#3>8 {####9}%
          \fi\fi\fi\fi\fi\fi\fi\fi\fi
        ##1%
      }%
    }
    \@{#5}%
    \edef\@##1{\endgroup
      \noexpand\renewcommand#1\noexpand#2\ifcase#3 \else [#3]\fi
      {##1\the\@temptokena}}%
    \@{#4}%
  %%  \show#2%
  }
```

Two possible error messages. Optimized. Second arg is the first letter from the meaning string; it will be "u" if and only if the control sequence is undefined.

```
  \long\def\patchcmdError#1#2{%
    \begingroup
    \toks@{Not redefinable}%
    \ifcat\relax\noexpand#1% Is it a control sequence?
      \begingroup
      \let#1=?\ifx ?\relax % Is it "\relax"?
        \endgroup % accept current value of \toks@
      \else \endgroup
```

This is equivalent to the `\@ifundefined` test (true if arg 2 is either undefined or its meaning is `\relax`).

```
        \if\ifx\relax#1u\else #2\fi u%
          \toks@{Not defined}%
        \fi
```

```
    \fi
  \fi
```

Apply `\string` preemptively to arg 1 to prevent catastrophic failure if it happens to be `\par` (`\PackageError` isn't defined as `\long` in older versions of LaTeX).

```
  \edef\@{\endgroup
    \noexpand\PackageError{patchcmd}{%
      \the\toks@: \string#1}\noexpand\@ehd}%
  \@
}
```

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

```
\endinput
```