Abstract

Compsci is useful whenever writing about programming, but especially when writing about TeX and especially when used as a supplement to the \texttt{ltxdoc} class to document \LaTeX{} macros in a literate programming style with \texttt{dtx} files.

Contents

<table>
<thead>
<tr>
<th>I</th>
<th>Discussion</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Textual elements</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Referring to commands</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Miscellaneous</td>
<td>5</td>
</tr>
<tr>
<td>II</td>
<td>Implementation</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Version control</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Requirements and context</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Hyperlatex option</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Textual elements</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Referring to commands</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>Miscellaneous</td>
<td>15</td>
</tr>
<tr>
<td>III</td>
<td>Configuration</td>
<td>17</td>
</tr>
</tbody>
</table>
Part I
Discussion

1 Textual elements

\env \{\textit{environment name}\} typesets the name of a \LaTeX{} environment. For example, the \texttt{enumerate} environment.

\textit{To do:} Add \texttt{\bib} how about?

\bst \{\textit{bibstyle name}\} typesets the name of a \BibTeX{} bibliography style. For example, the \texttt{achicago} bibstyle.

\package \{\textit{package name}\} typesets the name of a \LaTeX{} package. For example, the \texttt{compsci} package.

\class \{\textit{class name}\} typesets the name of a \LaTeX{} class. For example, the \texttt{letter} class.

\file \{\textit{file name}\} typesets the name of a disk file. For example, the file \texttt{Makefile.in}.

\textit{To do:} Should I use url.sty’s \texttt{\path} command for this?

\ext \{\textit{extension}\} typesets the name of a disk file extension. For example, the extension \texttt{dvi}. The argument should not include the dot that separates the root name from the extension. The dot can be inserted by the macro if you want it.

\caveat \{\textit{warning text}\} typesets warning paragraphs. You can also enclose the \textit{\texttt{warning text}} in a \texttt{warning} environment. Using the \texttt{warning} environment is necessary if the \textit{\texttt{warning text}} is to include |...|s, a \texttt{codeexample} environment or other “verbatim” constructs.

\textit{Warning:} This is an example \texttt{\caveat} or \texttt{warning} environment.

\todo \{(\textit{text})\} command and the \texttt{todoenv} environment are analogous to \texttt{\caveat} and \texttt{warning}, but they typeset paragraphs of a task “to do”.

\textit{To do:} This is an example \texttt{\todo} or \texttt{todoenv} environment.

\code \{(\textit{verbatim text})\} and \{(\texttt{verbatim text})\} typeset short pieces of code \texttt{verbatim}. For example, \{(\texttt{#\%$\^}\}) looks like \texttt{#\%$\^}. Instead of matching curly braces, any two identical characters, even a space, may be used to contain the \textit{\texttt{verbatim}} after the \texttt{\code} command.

\textit{To do:} Implement \texttt{\code*}. Like \texttt{\verb*}, \texttt{\code*} marks spaces explicitly. This is not an example todo, this is a real one for the \texttt{compsci} package!

\texttt{\codeexample} \{(\textit{text})\} typesets \{(\textit{text})\} in a normal way, but indicates by boxing that the contents are an example of how some piece of code actually appears. For example, this looks like this. These example environments may be used to present a piece of source code and what it looks like when typeset. The \texttt{\codeexample} environment typesets its contents like \texttt{\code} except they are set off in their own indented block element instead of being boxed.

\textit{This is an example of the \texttt{\codeexample} environment.}

\texttt{\codeexample} and \texttt{\codeexample*} environments are similar analogs of \texttt{\code} and \texttt{\code*}. The \texttt{\codeexample} environment is like the \texttt{\codeexample} en-
vironment except that \( \{, \} \) have their usual meanings. Inside the argument, \( \backslash \)
produces a typeset backslash.

This is a \texttt{\textbackslash env\{codeexample\}} environment: \( \backslash \{ \int e^x \, dx \} \) {}{{{!#@$%^&\}
There is a percent sign in the source after the $.

This is a \texttt{\textbackslash env\{codeexample*\}} environment: \( \backslash \{ \int e^x \, dx \} \) {}{{{!#@$%^&\}
Someday, I would like to get rid of the leading spaces in a line.

This is a codeexamplex environment: \( \int e^x \, dx \)
The earlier examples had three unmatched open-braces which had to be
closed in this environment. Also the backslash had to be doubled. Percent signs get ignored---a bug.

\textbf{Warning:} The codeexamplex environment ideally should ignore \%s in
the first column and print all others. The best I can do for the moment is to ignore
all \%s.

The bothexample and bothexample* environments typeset their contents two
times, side by side. The left side is set like either a codeexample or codeexample*
environment, and the right side is set like a typesetexample environment.

I find setting a line width of 65 characters in your editor helps keep this kind
of example from bleeding to the right.

This is one part of a \texttt{\textbackslash env\{bothexample\}} environment:
\( \backslash \{ \int e^x \, dx \} \) \fbox{jub-jub} \par
The contents must not contain any \LaTeX{} errors.

There is an annoying \% plus three explicit spaces at the beginning of
the first part. They are a bug! Also notice also the missing percent
sign after the backslash after ‘‘annoying’’---another bug that affects
the verbatim part only.

LOOKS LIKE:

\begin{quote}
This is one part of a bothexample environment: \( \int e^x \, dx \) jub-jub
The contents must not contain any \LaTeX{} errors.
There is an annoying \% plus three explicit spaces at the beginning of
the first part. They are a bug! Also notice also the missing percent
sign after the backslash after “annoying”---another bug that affects
the verbatim part only.
\end{quote}

The splitexample environment takes an argument which is used on the right
side (the typeset side) of an element that looks like bothexample. The contents
of the environment are set on the left side. This is a way to cheat, making the
right side something other than the strict typesetting of what’s on the left.

\textbf{Warning:} The splitexample and bothexample environments execute ex-
ample code. The example code is executed in a group, but be careful with global
assignments, and with assignments that might screw up the example environment
itself before the group ends. Notice that \texttt{\textbackslash setcounter} is \texttt{global}.

The bothexample and splitexample environments use an auxiliary file with
extension \texttt{vrb}, but a second pass with \LaTeX{} is not necessary.
To do: Implement \texttt{splitexample} and \texttt{splitexample*}.

To do: Get the \texttt{bothexample} sub-environments side by side! Right now they are sequential. I recommend keeping line length inside this environment less than 40, however, in case the side-by-side presentation is ever possible. I have not followed this advice myself.

To do: Implement \texttt{codeexample*x*}.

Warning: The \texttt{bothexample} environment is working nicely now, but I have not even touched the other complicated verbatim-like environments, so they might be broken.

\url\email\path\file\option\program\lips\book

The \texttt{compsci} package uses the \texttt{url} package by Donald Arseneau to handle the typesetting of URLs, email addresses, and filesystem paths. See that package for details of how line breaks are handled within these elements.

\option\program\lips\book

The \texttt{compsci} package uses the \texttt{lips} package, so you can use the \texttt{lips} command for text ellipses. My love is like a red red rose. . . .

The \texttt{compsci} package uses the \texttt{titles} package, so several commands like \texttt{book} are available.

2  Referring to commands

\cs \cmd

\cs\cmd \texttt{(command sequence)} typesets a command sequence, such as the one that starts this sentence. \texttt{\cmd} does the same thing. For example, \texttt{\cs\foo} looks like \texttt{\foo}. When writing, e.g., \texttt{\cs\foo} the following spacing is automatically handled. The result doesn’t look different from using the \texttt{|...|} or \texttt{\code} syntax, but the markup allows the parsing of the macro name itself for some future purpose such as indexing.

Warning: You cannot say \texttt{\cs\foo@bar} when \texttt{@} is not a letter (category code 11), you must say \texttt{\cs{\foo@bar}} or \texttt{\cname{foo@bar}} instead.

Nor can you say \texttt{\cs\par}. It trips on \texttt{\text@command}. Nor can you say \texttt{\cs\iffoo}. Use \texttt{\cname{iffoo}} instead.

\cname\oarg

\cname\oarg \texttt{(command name)} is like \texttt{\cs} but its argument is evaluated and pre-fixed with a backslash character. For example, writing \texttt{\cname{@tfor}} gives you \texttt{\@tfor}.

\marg\oarg

\marg\oarg \texttt{(mandatory argument)} typesets a mandatory argument, and \texttt{\oarg} \texttt{(optional argument)} typesets an optional argument. For example, These macros are defined with \texttt{\meta}, so you can use \texttt{\} for a typeset backslash.

\cs\GobbleMOM\marg{first arg}\oarg{optional second arg}\marg{third arg}

LOOKS LIKE:
\begin{quote}
\texttt{\GobbleMOM \{(first arg)\{(optional second arg)\{(third arg)\}}}
\end{quote}

\meta

\meta \texttt{(text)} refers to a syntactic placeholder. You would use this for example to talk about one of the arguments of a command you’ve introduced, like \texttt{(text)} above, which can contain normal text but is broken across lines only at spaces. Within the argument \texttt{\} is a typeset backslash.
3 Miscellaneous

\newprogram  The compsci package uses the abbrevs package to define the Program category of abbreviation. Use \newprogram{⟨\csname⟩}{⟨initial expansion⟩}{⟨subsequent expansion⟩} to define a new abbreviation {⟨\csname⟩} for a program.

\ProcessDTXFile  \ProcessDTXFile can be used by a documentation driver file of the kind you might be inclined to make using the doc package. Its purpose is to be able to load a package up to a certain point so that just descriptive information can be read without reading in the whole file. \ProcessDTXFile{⟨filename⟩} defines the variable \JustLoadInformation and inputs the file ⟨filename⟩. To be useful, the file must have a statement that branches on whether \JustLoadInformation is defined. For an example of its use, see the present package.

\MaybeBibliography  Sorry, you will need to look at the source below.

\AddToCheckSum  \AddToCheckSum{⟨number⟩} simply adds ⟨number⟩ globally to the checksum counter implemented in the doc package. This macro finds applications when the sources of a package are spread over one or more files such as configuration files or files concatenated by a Make process. For an example of its use, see the present package.

\BibTeX  The Bib\TeX logo is provided if it isn’t already defined.
Part II
Implementation

4 Version control

These definitions must be the first ones in the file.
1 \def\fileinfo{macros for writing about macros (Frankenstein’s conscience)}
2 \def\DoXPackageS {}  
3 \def\fileversion{v1.2}
4 \def\filedate{2001/08/31}
5 \def\docdate{2001/08/31}
6 \edef\PPOptArg {%
7 \filedate\space \fileversion\space \fileinfo
8 %}

If we’re loading this file from a \ProcessDTXFile command (see the compsci package), then \JustLoadDInformation will be defined; otherwise we assume it is not (that’s why the FunkY Name).

If we’re loading from \ProcessDTXFile, we want to load the packages listed in \DoXPackageS (needed to typeset the documentation for this file) and then bail out. Otherwise, we’re using this file in a normal way as a package, so do nothing. \DoXPackageS, if there are any, are declared in the dtx file, and, if you’re reading the typeset documentation of this package, would appear just above. (It’s OK to call \usepackage with an empty argument or \relax, by the way.)

\makeatletter% A special comment to help create bst files. Don’t change!
\@ifundefined{JusTLoaDInformatioN} {%
  % ELSE (we know the compsci package is already loaded, too)
  \undefinedCS\JustLoadDInformation
  \SaveDoXVarS
  \eExpand\csname DoXPackageS\endcsname\In {}\use\csname in case it’s undefined
  \usepackage{#1}%
%}
\restoreDoXVarS
\makeatother
\endinput
%
% A special comment to help create bst files. Don’t change!

Now we check for \LaTeX{}2e and declare the \LaTeX{}X package.

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{compsci}[\PPOptArg]

5 Requirements and context

\RequirePackage{abbrevs,alltt,lips,moredefs,relsize,shortvrb,%
slemph,titles,url,verbatim}

6 Hyperlatex option

Handle hyperlatex option here.

To do: Should this inherit globally?
Long URLs are so common and screw up LaTeX in the middle of a paragraph that a good scheme seems to be to relegate them to footnotes. Make option/switch for footnotes and inline.

Actually, when it’s a footnote, there’s almost never going to be a linebreak, unless the url is longer than the textwidth of the footnote...so \texttt{} is probably going to be fine.

Could also use \marginpar{\smaller\url{#1}}—definitely would have to use \url not \texttt there.

We undefine \code and \file immediately because we want compsci’s versions not hyperlatex’s.

\texttt{To do: Alternative ways to handle things}
We use the `url` package to implement \texttt{code} and the `shortvrb` package to implement \texttt{...}. There are no linebreaks in either. See the respective packages for considerations of robustness.

\textbf{To do:} summarize robustness considerations here.

What I’m doing here is defining \texttt{codeexample} and \texttt{codeexample*} to be just like the `doc` package’s \texttt{verbatim} and \texttt{verbatim*}. That is, they ignore \% characters at the beginning of lines.
These are defined in the `doc` package, which may be loaded.

To do: use `verbatim-write trick`

to implement the `bothexample` environments we write out the contents of the environment and read them in twice, typesetting them differently each time. The
verbatimwrite environment comes from the moreverb package. Reading the text twice seems unavoidable because we want to read them with different catcodes each time.

To do: Figure whether the tab stuff here is going is a help or a hindrance.
The first \begingroup...\endgroup I think is necessary to contain the verbatim activity. The second is necessary so that activity in the example environment does not effect anything outside it. Of course global activity will, so be careful! The \minipage effect these second group. [Hmm, I don’t see a “second” group......]

\newboolean{@cs@using@doc@}
\newenvironment{bothexample} {%
  @ifundefined{ver@doc.sty} {
    @cs@using@doc@false
  }{% ELSE
    @cs@using@doc@true
  }
\begingroup
  \cs@verbatim@write{\jobname.vrb}
}{%
  \end@cs@verbatim@write
  \cs@verbatim@input{\jobname.vrb}
\endgroup
\noindent{\sffamily\larger LOOKS LIKE:}
\par
\noindent{fbox{%
  \minipage{\textwidth}
    \MakePercentComment
    \input{\jobname.vrb}
  \endminipage
}\par
\medskip
}

We can’t use #1 in the end-environment part, so we write it out. Again we have the problem of \%s.

\Warning: This is broken!
Abstract markup of terms and objects.
The `hyperlatex` option will change `\email` at the end of this package.

8 Referring to commands

Here I redefine `\meta` to include a `\normalfont` right at the beginning inside a group, since the command was working strangely when `\ttfamily` was in effect. Also, I redefine `\` to be a typeset backslash.

These are simply from the `doc` package.

13
These use \meta and are therefore slightly different than their counterparts in \ltxdoc. A tiny bit of breakability is added at the beginning.

\marg

\oarg

\cname

\cmd

\cs

For referring to a \LaTeX{} macro. The code is a mix from the \ltxdoc class and the \abbrevs package. There is no need to check for \nocorr commands inside the argument so we skip \text@command. We might conceivably be in a typewriter slanted, so we do check for italic corrections. We don’t handle the case of calling this in math mode (see the definition of \DeclareTextFontCommand in the kernel).

We handle the cases of \cmd{\texttt{\foo}} and \texttt{\foo} differently. The latter needs logic to determine the following space, whereas the former does not. They both need italic corrections.
9 Miscellaneous

See the version control section above and all throughout the Frankenstein bundle for examples of when you might want to use these commands.

We make sure it's OK to use \def multiple times. You might need to call packages with options.

\newcommand*{\ProcessDTXFile}[1]{% 
  \def\filename {#1}\% 
  \def\JusTLoaDInformatioN {}\% 
  \UndefineCSinitelyHavECitationS \% not necessary in my usage, but safe 
  \input{#1}\% 
  \UndefineCS\JusTLoaDInformatioN % not necessary in my usage, but safe 
}

\newcommand*{\IfCitations}[1]{% 
  \@ifundefined{initelyHavECitationS} {\% 
  #1\relax \% 
  }{% \% ELSE \% 
  #1\relax \% 
  }% 
}

\newcommand*{\IfJustLoadInformation}[1]{% 
  \@ifundefined{JusTLoaDInformatioN} {\% 
  #1\relax \% 
  }{% \% ELSE \% 
  #1\relax \% 
  }% 
}

\newcommand*{\SaveDoXVarS}{% 
  \SaveCS\fileinfo \SaveCS\fileversion \SaveCS\filedate \SaveCS\docdate \SaveCS\DoXPackageS \% 
  \UndefineCS\initelyHavECitationS \% 
  \RestorECitationS \% 
  \IfCitations {\% 
  \let\RestorECitationS\relax \% 
  }% 
}

\newcommand*{\RestoreDoXVarS}{% 
  \RestoreCS\fileinfo \RestoreCS\fileversion \RestoreCS\filedate \RestoreCS\docdate \RestoreCS\DoXPackageS \% 
  \RestoreCS\DoXPackageS \% 
  \RestorECitationS \% 
  }% 

\newcommand*{\AddToCheckSum}[1]{% 
  \global\advance\check@sum #1\% 
}

\newprogram \TMFontProgram \TMHookProgram \TMResetProgram

Provide for program abbreviations.

\NewAbbrevCategory{Program}% 
\NewUserAbbrevDefiner{\newprogram}{Program}% 
\def\TMFontProgram \%
\BibTeX  The \TeX\ logo.
\providecommand{\BibTeX}{\rmfamily B\kern-.05em\textsc{i\kern-.025emb}\kern-.08emT\kern-.1667em\raisebox{-.7ex}{E}\kern-.125emX}
Other things are deferred to a configuration file.

```latex
\InputIfFileExists{compsci.cfg}{}
```

The contents of the distributed configuration file are below.

```latex
\def\fileinfo{Compsci package configuration}
\def\fileversion{v1.2}
\def\filedate{2001/08/31}
\def\docdate{2001/08/31}
\ProvidesFile{compsci.cfg}
```

Various program names.

```latex
\newprogram{Frankenstein}{Frankenstein}
\newlet{monster}{Frankenstein}

To do: Hmm, \LaTeX\ kernel uses \verb|\DeclareRobustCommand|. What does \TeX\ kernel do?
```

```latex
\newcommand{\LaTeX}{}
\newabbrev{CTAN}{CTAN}
\newprogram{kpathsea}{kpathsea}
\newprogram{GNU Emacs}{GNU Emacs}
\newprogram{AUC-\TeX}{AUC-\TeX}
\newprogram{$\cal NTS$}{$\cal NTS$}
\newprogram{Makeindex}{Makeindex}
\newprogram{e-\TeX}{e-\TeX}
\newprogram{\LaTeX{}3}{\LaTeX{}3}
\newprogram{IDVI}{IDVI}
```
Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>\LaTeX</th>
<th>\alltt</th>
<th>\AtEndOfPackage</th>
<th>\auctex</th>
<th>\codeexample*</th>
<th>\codeexample**</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\v</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@beginpenalty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@eschack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@cs@first@line!false</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@cs@first@line!true</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@cs@using!doc!false</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@cs@using!doc!true</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@docverbatim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@doendpe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@eha</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@eschack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@flushglue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@ifnextchar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@undefined</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@makeother</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@oververbatim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@tempsvafalse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@tempsvatrue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@totallengthmargin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@verbatim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@unskipspaces</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@verbatim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\\</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\closein</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\closeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\cmd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\code*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\codeexample</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\A

\active \begin{tabular}{ll}
159, 173, 319, 320 & 173 \\
\AddToCheckSum & 5, 375 \\
\advance & 415 \\
\aftergroup & 364, 371 \\
\end{tabular}\edef & 6

\D

\DeclareOption & 25

\DeclareRobustCommand & 354

\def \begin{tabular}{ll}
1–5, 34, 45, 56, & 82 \\
59, 60, 64, 67, & 60 \\
68, 78, 121, & 82 \\
140, 160, 178, & 59 \\
188, 304, 305, & 59 \\
311, 312, 322, & 160 \\
330, 335, 375, & 160 \\
379, 380, 401, 419, & 160 \\
\end{tabular}

\edef & 6

\edef & 6

\edef & 6

\edef & 6