cprotectinside — Use cprotect arbitrarily deeply nested *

user202729

Released 2022/06/25

Abstract

Extends on `cprotect` package to allow users to use verbatim-like commands inside arbitrary parameters.

*This file describes version v0.0.0, last revised 2022/06/25.

1 Usage

1.1 Main function

\cprotectinside `\cprotectinside` ⟨*delimiter*⟩ ⟨*content*⟩

Execute ⟨*content*⟩ that possibly contains verbatim content.

This is a bit hard to explain. To give an example:

Take an example from https://tex.stackexchange.com/q/24574, you want to write `\textbf{using \verb|-i| as}`. Unfortunately, `\textbf` does not accept `\verb` in input.

Then you need to

- Wrap the whole code inside `\cprotectinside{!}{ ... }`
- Wrap each `\verb` command (and its arguments) inside `!...!`.

The resulting code would be:

```
\cprotectinside{!}{\textbf{\using !\verb|-i|! as}}}
```

The first argument `!` is the delimiter, it can be changed as long as it's changed correspondingly in the ⟨*content*⟩ part.

In this simple case, it's possible to simply use `cprotect`. Nevertheless this package is useful in some cases, for example...

- Value of keyval argument:

```
\cprotectinside{*}{
\begin{lstlisting}[language=Perl,
        caption={text *\verb+myverb+* some more text}]
code code
code code
\end{lstlisting}
}
```

- Pseudo-environments (contrived example):

```
\cprotectinside{*}{
        \begin{align}
                1 &= 2+\text{text *\verb+text+* text}
        \end{align}
}
```

1.2 Common issues

- If you get the error message

  > LaTeX cmd Error: Verbatim-like command '`\cprotectinside`' illegal in argument.

You can't use a literal TAB character inside ⟨*content*⟩ because of a known bug: https://tex.stackexchange.com/q/508001

(when the tab character is at the begin of a line – which is the most common case, it will give no error message but they will be removed from the code. Which might matter for `verbatim` environments, for example.)

- Note that the following code will not work as desired

```
\cprotectinside{!}{
        some content
        %}
        some other content
}
```

because of the unbalanced brace in the comment. It's possible to use characters different from braces to delimit the second argument

```
\cprotectinside{!}|
        some content
        %}
        some other content
|
```

or use the workarounds described in the documentation of `cprotect` package.

- Contents that is intended to be passed as "text" to the outer command must not be `cprotect`ed.

  As a general rule of thumb, if the content inside can be replaced with a `\includegraphics` with no compilation error, it can be `cprotect`ed.

1.3 Implementation note

The working of the code is similar to how `cprotect` package works. Described in more details in https://tex.stackexchange.com/q/622512 (post by the package author).

In particular, given the code `\textbf{using \verb|-i| as}` the command might transform it to become `\textbf{using \cpiContentAi/ as}` then execute the resulting code. Where `\cpiContentAi/` is defined to be something similar to `\input{inner.tex}`, with the (imaginary) file `inner.tex` has the content `\verb|-i|\empty`.

The string to be appended inside each inner chunk defaults to `\empty`, used to remove the space generated at the end of each file (similar in purpose to `^^E^^L` used by `cprotect` package), but it can be configured:

`\cprotectinsideAppend` The content to be appended to every inner macro. Should already be detokenized. Defaults to the detokenized string `\empty`.

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.