The texassert package*

Hanson Char
hanson.char@gmail.com

November 16, 2024

Abstract

An assertion library for unit testing in plain TeX.

1 Introduction

This package emerged from a desire to explore `l3build` and literate programming. It provides a collection of Plain TEX macros that I originally used for unit testing, now converted into a `.dtx` file, allowing for regeneration of the original source files from the literate code.

All `.tex` files in this package are written in Plain TeX, offering a simple mechanism for performing assertions in unit testing Plain TEX macros. I hope you find it useful. *Profitez!*

2 Usage Examples

This section assumes you already have the `texassert` package installed via, for instance, running `l3build install` under the project's root folder.

2.1 Length Assertions

To unit test the `\lengthof` macro in this library, for example, we can create a file `length-tests.tex` with:

```
% Import the necessary macros
\input import \import{lengthof} \import{assert}

% Length of an empty string is zero
\lengthof{} \asserteq\the\length=0

% Length of ‘0’ is one
\lengthof{0} \asserteq\the\length=1

% Length of ‘12.3456’ is seven
\lengthof{12.3456} \asserteq\the\length=7
```

*This document corresponds to texassert v0.0.2, dated 2024/11/07.

```
% Summary of the assertions made so far
\assertionsummary
\bye
```

Compile it with a TEX engine, e.g. `pdftex length-tests.tex`, we get an output file `length-tests.pdf` with:

Assertion Summary: 3/3 assertions passed i.e. 0/3 assertions failed.

2.2 Number scanning

The following demonstrates how TEX scans and expands the input tokens when a number is encountered. First, create a file e.g. `number-scanning.tex` with:

```
% Import the necessary macros
\input import \import{assert}

\count1=1
\count2=2
\count12=912

% Notice how \string\count2 gets absorbed to become
% the number of the first count!
\count3=\count1\the\count2
\asserteq\the\count3={912}

% Several ways to get around the issue.
\count3=\count1 \the\count2
\asserteq\the\count3=1

\count3=\count1\relax\the\count2
\asserteq\the\count3=1

\count3=\count1{}\the\count2
\asserteq\the\count3=1

% Summary of the assertions made so far
\assertionsummary
\bye
```

Compile it with a TEX engine, e.g. `pdftex number-scanning.tex`, we get an output file `number-scanning.pdf` with:

2 2 2
Assertion Summary: 4/4 assertions passed i.e. 0/4 assertions failed.

2.3 More Examples

Many more examples can be found and easily extracted from the `*.lvt` files of the regresssion test suite. I encourage the motivated readers to take a look. Go check out the repository and run them via `l3build check`!

3 Source Repository

https://github.com/hansonchar/texassert

4 Useful Resources

Not so much related to the library provided by this package per se, but some commands and external resources which I found directly useful or necesssary for the purpose of *constructing* this package per se.

1. Examples in the `l3build` repository. The `simple-tree` example in particular.
2. `texdoc l3build` – information directly related to `l3build`.
3. `texdoc doc` – the `doc` package used by `l3build` implicitly.
4. `texdoc docstrip` – the `docstrip` package used by `l3build` implicitly.
5. `texdoc source2e` – information related to various macros that are or can be used in a `.dtx` file.
6. `texdoc dtxtut` – Scott Pakin. *How to Package Your LaTeX Package.* January 21, 2024. (I had lots of *Aha!* moments in reading this.)
7. Michel Gossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison Wesley, Reading, Massachusetts, October 1, 1994. ISBN 0-201-54199-8.
8. David Salomon. *The Advanced TeXbook.* Springer-Verlag New York, 1995. ISBN 0-387-94556-3.

5 Implementation

`import.tex` Contains `\import`.

`\import` `{`⟨*filename*⟩`}`. Used to prevent the same file from being `\input` more than once.

```
\def\import#1{%
  \expandafter\ifx\csname import:#1\endcsname\relax
    \input #1
    \expandafter\gdef\csname import:#1\endcsname{}%
  \fi
}
```

`common.tex` Contains common code and configuration used in this library.

```
\showboxdepth=\maxdimen \showboxbreadth=\maxdimen

\newtoks\result \newtoks\tokstemp
\newcount\n
\newcount\integer

\def\true{\let\bool=\iftrue}
\def\false{\let\bool=\iffalse}
```

`\debug` `{`⟨*message*⟩`}`. Writes a line of debug message immediately to the terminal and the log file when debugging is enabled (via `\debugtrue` which is the default).

```
\newif\ifdebug
\debugtrue
\def\debug#1{\ifdebug \immediate\write16{[DEBUG] #1}\fi}
```

`\ifEmpty` `[`⟨*parameter*⟩`]\then`. Checks if the given parameter delimited by `\then`, when fully expanded, is empty. No parameter is treated as empty.

```
\newif\ifempty
\def\checkifempty#1{{\expandafter\def\expandafter\input\expandafter{#1}%
  \global\ifx\input\empty \emptytrue\else\emptyfalse\fi}}

% Assigning \iffalse to \then and use as a parameter delimiter
% is critical in making the if-macros skippable.
% Source: https://tug.org/TUGboat/tb45-1/tb139wermuth-isint.pdf
\let\then=\iffalse
\def\ifEmpty#1\then{%
  \checkifempty{#1}\ifempty
}
```

`\ifUndefined` `{`⟨*cs token*⟩`}\then`. Checks if the given control sequence delimited by `\then` is undefined.

```
\long\def\ifUndefined#1\then{{%
  \edef\x{\meaning#1}%
  \let\e=\escapechar \escapechar=-1
  \edef\y{\string\undefined}\escapechar=\e
  \def\true{\iftrue}\def\false{\iffalse}%
  \def\next{\expandafter\expandafter\expandafter
    \aftergroup\ifx\x\y\true\else\false\fi}\next}}
```

`\ifDefined` `{`⟨*cs token*⟩`}\then`. Checks if the given control sequence delimited by `\then` is defined.

```
\long\def\not#1#2\then{#1#2\then \false \else \true \fi \bool}
\long\def\ifDefined#1\then{\ifUndefined#1\then
  \false \else \true\fi \bool}
```

`lengthof.tex` Contains the code used to find out the length of a given string.

`\lengthof` `{`⟨*string*⟩`}` Computes the length of the given string parameter when fully expanded.

```
\newcount\length
\def\lengthof#1{{\edef\x{#1}\global\length=0
  \expandafter\lengthofA\x\end
}}
\def\lengthofA#1{\ifx#1\end\else
  \global\advance\length by1
  \expandafter\lengthofA\fi}
```

`checkeq.tex` Contains the code used to check if two given strings are equal.

`\checkeq` `{`⟨*string*⟩`}{`⟨*string*⟩`}`. Used to check if two given string parameters, when fully expanded, are equal. Assume no space in the strings.

```
\input import \import{lengthof}

\newif\ifeq
\chardef\temp=\catcode`@\catcode`@=11

\global\eqtrue
% Assume no spaces
\def\checkeq#1#2{{%
  \edef\checkeq@fstparam{#1}%
```

```
  \edef\checkeq@sndparam{#2}%
  \lengthof\checkeq@fstparam \edef\lena{\number\length}%
  \lengthof\checkeq@sndparam \edef\lenb{\number\length}%
  \ifx\lena\lenb
    \ifnum\length=0
      \global\eqtrue \let\next=\relax
    \else
      \expandafter\expandafter\expandafter
        \def\expandafter\expandafter\expandafter
          \next\expandafter\expandafter\expandafter
            {\expandafter\expandafter\expandafter
              \checkeqA\expandafter\checkeq@fstparam
                \expandafter\eot\checkeq@sndparam\eot}%
    \fi
  \else
    \global\eqfalse \let\next=\relax
  \fi
  \next
}}
\def\checkeqA#1#2\eot#3#4\eot{%
  \if#1#3{}% the trailing `{}%' is necessary to avoid
    \ifx\relax#2\relax % extra spaces
      \global\eqtrue \let\next=\relax
    \else
      \def\next{\checkeqA#2\eot#4\eot}%
    \fi
  \else
    \global\eqfalse \let\next=\relax
  \fi
  \next
}

\catcode`@=\temp % restore the original catcode for @
```

`assert.tex` Contains the code used for assertion purposes.

```
\input import \import{checkeq} \import{common}

\ifDefined\ProvidesPackage\then
  \ProvidesPackage{texassert}
\fi

\newcount\countassertions
\newcount\countassertionspassed
\newcount\countassertionsfailed
\newif\ifassertmessageonly
\chardef\temp=\catcode`@\catcode`@=11

\let\assertDone=\iffalse
\def\unexpected{\toks0={unexpected!}}
\def\expected{\toks0={expected}}
\def\assert{\asserteq\the\toks0={expected}}
\def\assertTrue#1\assertDone{#1\then \expected
  \else \unexpected\fi \assert}
\def\assertFalse#1\assertDone{#1\then \unexpected
```

```
  \else \expected\fi \assert}

\def\resetassertions{%
  \countassertions=0
  \countassertionspassed=0
  \countassertionsfailed=0
}
```

`\asserteq` `[`⟨*string*⟩`]={`⟨*string*⟩`}` Asserts that the two given strings, when fully expanded, are equal, taking catcode into account. The first string, if not specified, is treated as an empty string.

```
\def\asserteq#1=#2{{%
  \global\advance\countassertions by1
  \edef\assert@a{#1}%
  % \message{assert@a: [\meaning\assert@a]}%
  \edef\assert@b{#2}%
  % \message{assert@b: [\meaning\assert@b]}%
  \ifx\assert@a\assert@b\relax\relax
    \global\advance\countassertionspassed by1
  \else
    \global\advance\countassertionsfailed by1
    \message{...}%
    \def\errmsg{*** assertion (\the\countassertions) failure:
      `#1' not equal `#2' ***}%
    \message{\errmsg}%
    \ifassertmessageonly\else
      \medbreak
      \indent\indent{\errmsg}%
      \medbreak\fi
  \fi
}}
```

`\asserteqnocat` `[`⟨*string*⟩`]={`⟨*string*⟩`}` Asserts that the two given strings, when fully expanded, are equal, disregarding any catcode differences. The first string, if not specified, is treated as an empty string.

```
\def\asserteqnocat#1=#2{{%
  \global\advance\countassertions by1
  \edef\assert@a{#1}%
  % \message{assert@a: [\meaning\assert@a]}%
  \edef\assert@b{#2}%
  % \message{assert@b: [\meaning\assert@b]}%
  \checkeq\assert@a\assert@b
  \ifeq
    \global\advance\countassertionspassed by1
  \else
    \global\advance\countassertionsfailed by1
    \message{...}%
    \def\errmsg{*** assertion (\the\countassertions) failure:
      `#1' not equal `#2' ***}%
    \message{\errmsg}%
    \ifassertmessageonly\else
      \medbreak
      \indent\indent{\errmsg}%
      \medbreak\fi
```

```
  \fi
}}
```

`\assertneq` `[`⟨*string*⟩`]={`⟨*string*⟩`}`. Asserts that the two given strings, when fully expanded, are not equal, taking catcode into account. The first string, if not specified, is treated as an empty string.

```
\def\assertneq#1=#2{{%
  \global\advance\countassertions by1
  \edef\assert@a{#1}%
  % \message{assert@a: [\meaning\assert@a]}%
  \edef\assert@b{#2}%
  % \message{assert@b: [\meaning\assert@b]}%
  \ifx\assert@a\assert@b\relax\relax
    \global\advance\countassertionsfailed by1
      \message{...}%
      \def\errmsg{*** assertion (\the\countassertions) failure:
        `#1' equal `#2' ***}%
      \message{\errmsg}%
      \ifassertmessageonly\else
        \medbreak
        \indent\indent{\errmsg}%
        \medbreak\fi
  \else
    \global\advance\countassertionspassed by1
  \fi
}}
```

`\assertionsummary` Typesets a summary of the assertions made, then resets to a state as if no assertion has been made.

```
\def\assertionsummary{{%
  \def\sp{ }%
  \def\summary{%
    Assertion Summary:
      \the\countassertionspassed/\the\countassertions\sp
      assertions passed i.e.
    \the\countassertionsfailed/\the\countassertions\sp
      assertions failed.}%
  \message{\summary}%
  \ifassertmessageonly\else
    \medbreak
    \summary
  \fi}\resetassertions}

\catcode`@=\temp % restore the original catcode for @
```

`texassert.sty` Used for packaging purposes.

```
\input{assert}
```

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.

A

\advance 44, 115, 121, 123, 135, 142, 144, 156, 162, 172
\aftergroup 35
\assert . . . 103, 105, 107
\assert.tex 88
\assert@a . 116, 117, 120, 136, 137, 140, 157, 158, 161
\assert@b . 118, 119, 120, 138, 139, 140, 159, 160, 161
\assertDone 100, 104, 106
\asserteq 103, 114
\asserteqnocat 134
\assertFalse 106
\assertionsummary . 175
\assertneq 155
\assertTrue 104

B

\bool 13, 14, 36, 38

C

\catcode . 49, 87, 98, 189
\chardef 49, 98
\checkeq 46, 140
\checkeq.tex 46
\checkeq@fstparam 54, 56, 66
\checkeq@sndparam 55, 57, 67
\checkeqA . . . 66, 74, 79
\checkifempty . . . 19, 27
\common.tex 7
\countassertions 94, 110, 115, 125, 135, 146, 156, 164, 179, 181
\countassertionsfailed 96, 112, 123, 144, 162, 181
\countassertionspassed 95, 111, 121, 142, 172, 179
\csname 2, 4

D

\debug 15
\debugtrue 16
\def 1, 13, 14, 17, 19, 26, 29, 33, 34, 36, 37, 40, 43, 53, 63, 74, 79, 101, 102, 103, 104, 106, 109, 114, 125, 134, 146, 155, 164, 175, 176, 177

E

\e 31, 32
\edef 30, 32, 40, 54, 55, 56, 57, 116, 118, 136, 138, 157, 159
\else 20, 35, 36, 38, 43, 61, 69, 78, 81, 105, 107, 122, 128, 143, 149, 167, 171, 184
\empty 20
\emptyfalse 20
\emptytrue 20
\end 41, 43
\endcsname 2, 4
\eot 67, 74, 79
\eqfalse 70, 82
\eqtrue 51, 60, 77
\errmsg . . . 125, 127, 130, 146, 148, 151, 164, 166, 169
\escapechar 31, 32
\expandafter 2, 4, 19, 34, 41, 45, 62, 63, 64, 65, 66, 67
\expected . 102, 104, 107

F

\false . 14, 33, 35, 36, 38
\fi 5, 17, 20, 35, 36, 38, 45, 68, 71, 80, 83, 92, 105, 107, 131, 132, 152, 153, 170, 173, 187

G

\gdef 4
\global 20, 40, 44, 51, 60, 70, 77, 82, 115, 121, 123, 135, 142, 144, 156, 162, 172

I

\if 75
\ifassertmessageonly 97, 128, 149, 167, 184
\ifdebug 15, 17
\ifDefined 36, 90
\ifEmpty 18
\ifempty 18, 27
\ifeq 48, 141
\iffalse 14, 22, 25, 33, 100
\ifnum 59
\iftrue 13, 33
\ifUndefined 29, 37
\ifx 2, 20, 35, 43, 58, 76, 120, 161
\immediate 17
\import 1, 46, 88
\import.tex 1
\indent . . . 130, 151, 169
\input 3, 19, 20, 46, 88, 190
\integer 11

L

\lena 56, 58
\lenb 57, 58
\length 39, 40, 44, 56, 57, 59
\lengthof . . . 39, 56, 57
\lengthof.tex 39
\lengthofA . . 41, 43, 45
\let . . 13, 14, 25, 31, 60, 70, 77, 82, 100
\long 29, 36, 37

M

\maxdimen 7

\meaning 30, 117, 119, 137, 139, 158, 160
\medbreak 129, 131, 150, 152, 168, 170, 185
\message 117, 119, 124, 127, 137, 139, 145, 148, 158, 160, 163, 166, 183

N

\n 10
\newcount 10, 11, 39, 94, 95, 96
\newif . . . 15, 18, 48, 97
\newtoks 9
\next 34, 35, 60, 64, 70, 72, 77, 79, 82, 84
\not 36
\number 56, 57

P

\ProvidesPackage 90, 91

R

\relax 2, 60, 70, 76, 77, 82, 120, 161
\resetassertions 109, 187
\result 9

S

\showboxbreadth 7
\showboxdepth 7
\sp 176, 179, 181
\string 32
\summary . . 177, 183, 186

T

\temp 49, 87, 98, 189
\texassert.sty 190
\the 103, 125, 146, 164, 179, 181
\then . 22, 25, 26, 29, 36, 37, 90, 104, 106
\toks 101, 102, 103
\tokstemp 9
\true . . 13, 33, 35, 36, 38

U

\undefined 32
\unexpected 101, 105, 106

W

\write 17

X

\x 30, 35, 40, 41

Y

\y 32, 35

Change History

v0.0.1 – 2024-11-05

General: Initial version 1

v0.0.2 – 2024-11-07

General: Migrate source files to `texassert.dtx` 1