

The `texassert` package*

Hanson Char
`hanson.char@gmail.com`

November 12, 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, for instance, via `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 T_EX 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 T_EX 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 T_EX 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 necessary 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 L^AT_EX Package*. January 21, 2024. (I had lots of *Aha!* moments in reading this.)
7. Michel Gossens, Frank Mittelbach, and Alexander Samarin. *The L^AT_EX Companion*. Addison Wesley, Reading, Massachusetts, October 1, 1994. ISBN 0-201-54199-8.

5 Implementation

import.tex Contains `\import`.

```
\import {⟨filename⟩}. Used to prevent the same file from being \input more than once.  
1 \def\import#1{%  
2   \expandafter\ifx\csname import:#1\endcsname\relax  
3     \input #1  
4     \expandafter\gdef\csname import:#1\endcsname{%  
5       \fi  
6   }
```

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

```
7 \showboxdepth=\maxdimen \showboxbreadth=\maxdimen  
8  
9 \newtoks\result \newtoks\tokstemp  
10 \newcount\n  
11 \newcount\integer  
12  
13 \def\true{\let\bool=\iftrue}  
14 \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).

```
15 \newif\ifdebug  
16 \debugtrue  
17 \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.

```

18 \newif\ifempty
19 \def\checkifempty#1{\expandafter\def\expandafter\input\expandafter{#1}%
20   \global\ifx\input\empty \emptytrue\else\emptyfalse\fi}}
21
22 % Assigning \iffalse to \then and use as a parameter delimiter
23 % is critical in making the if-macros skippable.
24 % Source: https://tug.org/TUGboat/tb45-1/tb139wormuth-isint.pdf
25 \let\then=\iffalse
26 \def\ifEmpty#1\then{%
27   \checkifempty{#1}\ifempty
28 }
```

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

```

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

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

```

36 \long\def\not#1#2\then{#1#2\then \false \else \true \fi \bool}
37 \long\def\ifDefined#1\then{\ifUndefined#1\then
38   \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.

```

39 \input import \import{common}
40
41 \newcount\length
42 \chardef\temp=\catcode'\catcode'\@=11
43
44 \def\lengthof#1{\length=0
45   \bgroup
46     \edef\lengthof@input{#1}%
47     \ifEmpty\lengthof@input\then
48       \let\next=\relax
49     \else
50       \def\next{\expandafter\lengthofA\lengthof@input\eof}%
51     \fi
52   \next
53 \egroup
54 }
55 \def\lengthofA#1#2\eof{\global\advance\length by1
56   \ifEmpty#2\then
57     \let\next=\relax
58   \else
59     \def\next{\lengthofA#2\eof}%
```

```

60 \fi
61 \next
62 }
63
64 \catcode'\@=\temp % restore the original catcode for @

```

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

\checkeq $\{\langle string \rangle\}\{\langle string \rangle\}$. Used to check if two given string parameters, when fully expanded, are equal. Assume no space in the strings.

```

65 \input import \import{lengthof}
66
67 \newif\ifeq
68 \chardef\temp=\catcode'\@ \catcode'\@=11
69
70 \global\eqtrue
71 % Assume no spaces
72 \def\checkeq#1#2{%
73   \edef\checkeq@fstparam{#1}%
74   \edef\checkeq@sndparam{#2}%
75   \lengthof\checkeq@fstparam \edef\lena{\number\length}%
76   \lengthof\checkeq@sndparam \edef\lenb{\number\length}%
77   \ifx\lena\lenb
78     \ifnum\length=0
79       \global\eqtrue \let\next=\relax
80     \else
81       \expandafter\expandafter\expandafter
82         \def\expandafter\expandafter\expandafter
83           \next\expandafter\expandafter\expandafter
84             {\expandafter\expandafter\expandafter
85               \checkeqA\expandafter\checkeq@fstparam
86               \expandafter\eot\checkeq@sndparam\eot}%
87     \fi
88   \else
89     \global\eqfalse \let\next=\relax
90   \fi
91   \next
92 }
93 \def\checkeqA#1#2\eot#3#4\eot{%
94   \if#1#3{% the trailing '{}' is necessary to avoid
95     \ifx\relax#2\relax % extra spaces
96       \global\eqtrue \let\next=\relax
97     \else
98       \def\next{\checkeqA#2\eot#4\eot}%
99     \fi
100   \else
101     \global\eqfalse \let\next=\relax
102   \fi
103   \next
104 }
105
106 \catcode'\@=\temp % restore the original catcode for @

```

assert.tex Contains the code used for assertion purposes.

```

107 \input import \import{checkeq}
108
109 \ifDefined\ProvidesPackage\then
110   \ProvidesPackage{texassert}
111 \fi
112
113 \newcount\countassertions
114 \newcount\countassertionspassed
115 \newcount\countassertionsfailed
116 \newif\ifassertmessageonly
117 \chardef\temp=\catcode'\catcode'\@=11
118
119 \let\assertDone=\iffalse
120 \def\unexpected{\toks0={unexpected!}}
121 \def\expected{\toks0={expected}}
122 \def\assert{\asserteq\the\toks0={expected}}
123 \def\assertTrue#1\assertDone{#1\then \expected
124   \else \unexpected\fi \assert}
125 \def\assertFalse#1\assertDone{#1\then \unexpected
126   \else \expected\fi \assert}
127
128 \def\resetassertions{%
129   \countassertions=0
130   \countassertionspassed=0
131   \countassertionsfailed=0
132 }

```

`\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.

```

133 \def\asserteq#1=#2{%
134   \global\advance\countassertions by1
135   \edef\assert@a{#1}%
136   % \message{assert@a: [\meaning\assert@a]}%
137   \edef\assert@b{#2}%
138   % \message{assert@b: [\meaning\assert@b]}%
139   \ifx\assert@a\assert@b\relax\relax
140     \global\advance\countassertionspassed by1
141   \else
142     \global\advance\countassertionsfailed by1
143     \message{...}%
144     \def\errmsg{*** assertion (\the\countassertions) failure:
145       '#1' not equal '#2' ***}%
146     \message{\errmsg}%
147     \ifassertmessageonly\else
148       \medbreak
149       \indent\indent{\errmsg}%
150       \medbreak\fi
151   \fi
152 }}

```

`\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.

```

153 \def\asserteqnocat#1=#2{%
154   \global\advance\countassertions by1
155   \edef\assert@a{#1}%
156   % \message{assert@a: [\meaning\assert@a]]}%
157   \edef\assert@b{#2}%
158   % \message{assert@b: [\meaning\assert@b]]}%
159   \checkeq\assert@a\assert@b
160   \ifeq
161     \global\advance\countassertionspassed by1
162   \else
163     \global\advance\countassertionsfailed by1
164     \message{...}%
165     \def\errmsg{*** assertion (\the\countassertions) failure:
166       '#1' not equal '#2' ***}%
167     \message{\errmsg}%
168     \ifassertmessageonly\else
169       \medbreak
170       \indent\indent{\errmsg}%
171       \medbreak\fi
172   \fi
173 }}

```

`\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.

```

174 \def\assertneq#1=#2{%
175   \global\advance\countassertions by1
176   \edef\assert@a{#1}%
177   % \message{assert@a: [\meaning\assert@a]]}%
178   \edef\assert@b{#2}%
179   % \message{assert@b: [\meaning\assert@b]]}%
180   \ifx\assert@a\assert@b\relax\relax
181     \global\advance\countassertionsfailed by1
182     \message{...}%
183     \def\errmsg{*** assertion (\the\countassertions) failure:
184       '#1' equal '#2' ***}%
185     \message{\errmsg}%
186     \ifassertmessageonly\else
187       \medbreak
188       \indent\indent{\errmsg}%
189       \medbreak\fi
190   \else
191     \global\advance\countassertionspassed by1
192   \fi
193 }}

```

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

```

194 \def\assertionsummary{%
195   \def\sp{ }%
196   \def\summary{%
197     Assertion Summary:
198     \the\countassertionspassed/\the\countassertions\sp
199     assertions passed i.e.

```

```

200     \the\countassertionsfailed/\the\countassertions\sp
201     assertions failed.}%
202 \message{\summary}%
203 \ifassertmessageonly\else
204     \medbreak
205     \summary
206 \fi}\resetassertions}
207
208 \catcode'\@=\temp % restore the original catcode for @

```

texassert.sty Used for packaging purposes.

```

209 \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 | 55, 134, 140, 142, 154, 161, 163, 175, 181, 191 | |
| \aftergroup | 35 | |
| \assert | ... 122, 124, 126 | |
| \assert.tex | 107 | |
| \assert@a | . 135, 136, 139, 155, 156, 159, 176, 177, 180 | |
| \assert@b | . 137, 138, 139, 157, 158, 159, 178, 179, 180 | |
| \assertDone | 119, 123, 125 | |
| \asserteq | 122, 133 | |
| \asserteqnocat | 153 | |
| \assertFalse | 125 | |
| \assertionssummary | . 194 | |
| \assertneq | 174 | |
| \assertTrue | 123 | |
| B | | |
| \bgroup | 45 | |
| \bool | 13, 14, 36, 38 | |
| C | | |
| \catcode | 42, 64, 68, 106, 117, 208 | |
| \chardef | ... 42, 68, 117 | |
| \checkeq | 65, 159 | |
| \checkeq.tex | 65 | |
| \checkeq@fstparam | . | |
| | 73, 75, 85 | |
| \checkeq@sndparam | . | |
| | 74, 76, 86 | |
| \checkeqA | ... 85, 93, 98 | |
| \checkifempty | ... 19, 27 | |
| \common.tex | 7 | |
| \countassertions | .. | |
| | . 113, 129, 134, 144, 154, 165, 175, 183, 198, 200 | |
| \countassertionsfailed | | |
| | 115, 131, 142, 163, 181, 200 | |
| \countassertionspassed | | |
| | 114, 130, 140, 161, 191, 198 | |
| \csname | 2, 4 | |
| D | | |
| \debug | 15 | |
| \debugtrue | 16 | |
| \def | 1, 13, 14, 17, 19, 26, 29, 33, 34, 36, 37, 44, 50, 55, 59, 72, 82, 93, 98, 120, 121, 122, 123, 125, 128, 133, 144, 153, 165, 174, 183, 194, 195, 196 | |
| E | | |
| \e | 31, 32 | |
| F | | |
| \false | . 14, 33, 35, 36, 38 | |
| \fi | ... 5, 17, 20, 35, 36, 38, 51, 60, 87, 90, 99, 102, 111, 124, 126, 150, 151, 171, 172, 189, 192, 206 | |
| \edef | 30, 32, 46, 73, 74, 75, 76, 135, 137, 155, 157, 176, 178 | |
| \egroup | 53 | |
| \else | 20, 35, 36, 38, 49, 58, 80, 88, 97, 100, 124, 126, 141, 147, 162, 168, 186, 190, 203 | |
| \empty | 20 | |
| \emptyfalse | 20 | |
| \emptytrue | 20 | |
| \endcsname | 2, 4 | |
| \eot | 50, 55, 59, 86, 93, 98 | |
| \eqfalse | 89, 101 | |
| \eqtrue | 70, 79, 96 | |
| \errmsg | ... 144, 146, 149, 165, 167, 170, 183, 185, 188 | |
| \escapechar | 31, 32 | |
| \expandafter | 2, 4, 19, 34, 50, 81, 82, 83, 84, 85, 86 | |
| \expected | . 121, 123, 126 | |

| | | | |
|--|---|--|--|
| G | | R | |
| <code>\gdef</code> | 4 | <code>\lengthof</code> . . . | 39 , 75 , 76 |
| <code>\global</code> | 20 , 55 , 70 , 79 , 89 , 96 , 101 , 134 , 140 , 142 , 154 , 161 , 163 , 175 , 181 , 191 | <code>\lengthof.tex</code> | 39 |
| I | | <code>\lengthof@input</code> | 46 , 47 , 50 |
| <code>\if</code> | 94 | <code>\lengthofA</code> . . . | 50 , 55 , 59 |
| <code>\ifassertmessageonly</code> | 116 , 147 , 168 , 186 , 203 | <code>\let</code> | 13 , 14 , 25 , 31 , 48 , 57 , 79 , 89 , 96 , 101 , 119 |
| <code>\ifdebug</code> | 15 , 17 | <code>\long</code> | 29 , 36 , 37 |
| <code>\ifDefined</code> | 36 , 109 | M | |
| <code>\ifEmpty</code> | 18 , 47 , 56 | <code>\maxdimen</code> | 7 |
| <code>\ifempty</code> | 18 , 27 | <code>\meaning</code> 30 , 136 , 138 , 156 , 158 , 177 , 179 | |
| <code>\ifeq</code> | 67 , 160 | <code>\medbreak</code> | 148 , 150 , 169 , 171 , 187 , 189 , 204 |
| <code>\iffalse</code> | 14 , 22 , 25 , 33 , 119 | <code>\message</code> 136 , 138 , 143 , 146 , 156 , 158 , 164 , 167 , 177 , 179 , 182 , 185 , 202 | |
| <code>\ifnum</code> | 78 | N | |
| <code>\iftrue</code> | 13 , 33 | <code>\n</code> | 10 |
| <code>\ifUndefined</code> | 29 , 37 | <code>\newcount</code> | 10 , 11 , 41 , 113 , 114 , 115 |
| <code>\ifx</code> | 2 , 20 , 35 , 77 , 95 , 139 , 180 | <code>\newif</code> | 15 , 18 , 67 , 116 |
| <code>\immediate</code> | 17 | <code>\newtoks</code> | 9 |
| <code>\import</code> | 1 , 39 , 65 , 107 | <code>\next</code> | 34 , 35 , 48 , 50 , 52 , 57 , 59 , 61 , 79 , 83 , 89 , 91 , 96 , 98 , 101 , 103 |
| <code>\import.tex</code> | 1 | <code>\not</code> | 36 |
| <code>\indent</code> | 149 , 170 , 188 | <code>\number</code> | 75 , 76 |
| <code>\input</code> | 3 , 19 , 20 , 39 , 65 , 107 , 209 | P | |
| <code>\integer</code> | 11 | <code>\ProvidesPackage</code> | 109 , 110 |
| L | | Y | |
| <code>\lena</code> | 75 , 77 | <code>\y</code> | 32 , 35 |
| <code>\lenb</code> | 76 , 77 | X | |
| <code>\length</code> | 41 , 44 , 55 , 75 , 76 , 78 | <code>\x</code> | 30 , 35 |
| | | S | |
| | | <code>\showboxbreadth</code> | 7 |
| | | <code>\showboxdepth</code> | 7 |
| | | <code>\sp</code> | 195 , 198 , 200 |
| | | <code>\string</code> | 32 |
| | | <code>\summary</code> | 196 , 202 , 205 |
| | | T | |
| | | <code>\temp</code> | 42 , 64 , 68 , 106 , 117 , 208 |
| | | <code>\texassert.sty</code> | 209 |
| | | <code>\the</code> | 122 , 144 , 165 , 183 , 198 , 200 |
| | | <code>\then</code> | 22 , 25 , 26 , 29 , 36 , 37 , 47 , 56 , 109 , 123 , 125 |
| | | <code>\toks</code> | 120 , 121 , 122 |
| | | <code>\tokstemp</code> | 9 |
| | | <code>\true</code> | 13 , 33 , 35 , 36 , 38 |
| | | U | |
| | | <code>\undefined</code> | 32 |
| | | <code>\unexpected</code> | 120 , 124 , 125 |
| | | W | |
| | | <code>\write</code> | 17 |

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](#)