

# The `texassert` package\*

Hanson Char  
`hanson.char@gmail.com`

November 9, 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

### 2.1 Length Assertions

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

```
% 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

% 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 something like:

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

---

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

## 2.2 (More Examples ...)

TODO

## 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<sup>A</sup>T<sub>E</sub>X Package*. January 21, 2024. (I had lots of *Aha!* moments in reading this.)
7. Michel Gossens, Frank Mittelbach, and Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison Wesley, Reading, Massachusetts, October 1, 1994. ISBN 0-201-54199-8.

## 5 Implementation

**import.tex** Contains `\import`.

`\import` 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     % Imported #1\par
6   \fi
7 }
```

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

```
8 \showboxdepth=\maxdimen \showboxbreadth=\maxdimen
9
10 \newtoks\result \newtoks\tokstemp
11 \newcount\n
12 \newcount\integer
```

```

13
14 \def\true{\let\bool=\iftrue}
15 \def\false{\let\bool=\iffalse}

```

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

```

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

```

`\ifEmpty` Checks if the given parameter is empty.

```

19 \newif\ifempty
20 \def\checkifempty#1{\expandafter\def\expandafter\input\expandafter{#1}%
21   \global\ifx\input\empty \emptytrue\else\emptyfalse\fi}%
22
23 % Assigning \iffalse to \then and use as a parameter delimiter
24 % is critical in making the if-macros skippable.
25 % Source: https://tug.org/TUGboat/tb45-1/tb139wormuth-isint.pdf
26 \let\then=\iffalse
27 \def\ifEmpty#1\then{%
28   \checkifempty{#1}\ifempty
29 }

```

`\ifUndefined` Checks if the given control sequence is undefined.

```

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

```

`\ifDefined` Checks if the given control sequence is defined.

```

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

```

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

`\lengthof` Computes the length of the given string parameter.

```

40 \input import \import{common}
41
42 \newcount\length
43 \edef\temp{\the\catcode'\catcode'\@=11
44
45 \def\lengthof#1{\length=0 %
46   \bgroup
47     \edef\lengthof@input{#1}%
48     \ifEmpty\lengthof@input\then
49       \let\next=\relax
50     \else
51       \def\next{\expandafter\lengthofA\lengthof@input\eof}%
52     \fi
53   \next

```

```

54 \egroup
55 }
56 \def\lengthofA#1#2\eut{\global\advance\length by1 %
57 \ifEmpty#2\then
58 \let\next=\relax
59 \else
60 \def\next{\lengthofA#2\eut}%
61 \fi
62 \next
63 }
64
65 \catcode'\@=\temp % restore the original catcode for @

```

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

\checkeq Used to check if two given strings are equal. Assume no space in the strings.

```

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

```

```

104 \next
105 }
106
107 \catcode'\@=\temp % restore the original catcode for @

```

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

```

108 \input import \import{checked}
109
110 \ifDefined\ProvidesPackage\then
111 \ProvidesPackage{texassert}
112 \fi
113
114 \newcount\countassertions
115 \newcount\countassertionspassed
116 \newcount\countassertionsfailed
117 \newif\ifassertmessageonly
118 \edef\temp{\the\catcode'\@}\catcode'\@=11
119
120 \let\assertDone=\iffalse
121 \def\unexpected{\toks0={unexpected!}}
122 \def\expected{\toks0={expected}}
123 \def\assert{\asserteq\the\toks0={expected}}
124 \def\assertTrue#1\assertDone{#1\then \expected
125 \else \unexpected\fi \assert}
126 \def\assertFalse#1\assertDone{#1\then \unexpected
127 \else \expected\fi \assert}
128
129 \def\resetassertions{%
130 \countassertions=0
131 \countassertionspassed=0
132 \countassertionsfailed=0
133 }

```

**\asserteq** Asserts that the two given string are equal, taking catcode into account.

```

134 \def\asserteq#1=#2{%
135 \global\advance\countassertions by1
136 \edef\assert@a{#1}%
137 % \message{assert@a: [\meaning\assert@a]}%
138 \edef\assert@b{#2}%
139 % \message{assert@b: [\meaning\assert@b]}%
140 \ifx\assert@a\assert@b\relax\relax
141 \global\advance\countassertionspassed by1
142 \else
143 \global\advance\countassertionsfailed by1%
144 \message{...}%
145 \def\errmsg{*** assertion (\the\countassertions) failure:
146 '#1' not equal '#2' ***}%
147 \message{\errmsg}%
148 \ifassertmessageonly\else
149 \medbreak
150 \indent\indent{\errmsg}%
151 \medbreak\fi
152 \fi
153 }}

```

`\asserteqnocat` Asserts that the two given string are equal, disregarding any catcode differences.

```

154 \def\asserteqnocat#1=#2{%
155   \global\advance\countassertions by1
156   \edef\assert@a{#1}%
157   % \message{assert@a: [\meaning\assert@a]]}%
158   \edef\assert@b{#2}%
159   % \message{assert@b: [\meaning\assert@b]]}%
160   \checkedq\assert@a\assert@b
161   \ifeq
162     \global\advance\countassertionspassed by1
163   \else
164     \global\advance\countassertionsfailed by1
165     \message{...}%
166     \def\errmsg{*** assertion (\the\countassertions) failure:
167       '#1' not equal '#2' ***}%
168     \message{\errmsg}%
169     \ifassertmessageonly\else
170       \medbreak
171       \indent\indent{\errmsg}%
172       \medbreak\fi
173   \fi
174 }}

```

`\assertneq` Asserts that the two given string are not equal, taking catcode into account.

```

175 \def\assertneq#1=#2{%
176   \global\advance\countassertions by1
177   \edef\assert@a{#1}%
178   % \message{assert@a: [\meaning\assert@a]]}%
179   \edef\assert@b{#2}%
180   % \message{assert@b: [\meaning\assert@b]]}%
181   \ifx\assert@a\assert@b\relax\relax
182     \global\advance\countassertionsfailed by1
183     \message{...}%
184     \def\errmsg{*** assertion (\the\countassertions) failure:
185       '#1' equal '#2' ***}%
186     \message{\errmsg}%
187     \ifassertmessageonly\else
188       \medbreak
189       \indent\indent{\errmsg}%
190       \medbreak\fi
191   \else
192     \global\advance\countassertionspassed by1
193   \fi
194 }}

```

`\assertionssummary` Typesets a summary of the assertions made. Then reset to a state as if no assertion has been made.

```

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

```

```

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

```

texassert.sty Used for packaging purposes.

```

210 \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.

<b>A</b>		
\advance	56, 135, 141, 143, 155, 162, 164, 176, 182, 192	
\aftergroup	..... 36	
\assert	... 123, 125, 127	
\assert.tex	..... <u>108</u>	
\assert@a	. 136, 137, 140, 156, 157, 160, 177, 178, 181	
\assert@b	. 138, 139, 140, 158, 159, 160, 179, 180, 181	
\assertDone	120, 124, 126	
\asserteq	.... 123, <u>134</u>	
\asserteqnocat	.... <u>154</u>	
\assertFalse	..... 126	
\assertionssummary	. <u>195</u>	
\assertneq	..... <u>175</u>	
\assertTrue	..... 124	
<b>B</b>		
\bgroup	..... 46	
\bool	.... 14, 15, 37, 39	
<b>C</b>		
\catcode	..... 43, 65, 69, 107, 118, 209	
\checkeq	..... <u>66</u> , 160	
\checkeq.tex	..... <u>66</u>	
\checkeq@fstparam	. .... 74, 76, 86	
\checkeq@sndparam	. .... 75, 77, 87	
\checkeqA	... 86, 94, 99	
\checkifempty	... 20, 28	
\common.tex	..... <u>8</u>	
\countassertions	.. 114, 130, 135, 145, 155, 166, 176, 184, 199, 201	
\countassertionsfailed	.... 116, 132, 143, 164, 182, 201	
\countassertionspassed	.... 115, 131, 141, 162, 192, 199	
\csname	..... 2, 4	
<b>D</b>		
\debug	..... <u>16</u>	
\debugtrue	..... 17	
\def	1, 14, 15, 18, 20, 27, 30, 34, 35, 37, 38, 45, 51, 56, 60, 73, 83, 94, 99, 121, 122, 123, 124, 126, 129, 134, 145, 154, 166, 175, 184, 195, 196, 197	
\egroup	..... 54	
\else	21, 36, 37, 39, 50, 59, 81, 89, 98, 101, 125, 127, 142, 148, 163, 169, 187, 191, 204	
\empty	..... 21	
\emptyfalse	..... 21	
\emptytrue	..... 21	
\endcsname	..... 2, 4	
\eot	51, 56, 60, 87, 94, 99	
\eqfalse	..... 90, 102	
\eqtrue	..... 71, 80, 97	
\errmsg	... 145, 147, 150, 166, 168, 171, 184, 186, 189	
\escapechar	..... 32, 33	
\expandafter	.... 2, 4, 20, 35, 51, 82, 83, 84, 85, 86, 87	
\expected	. 122, 124, 127	
<b>F</b>		
\false	. 15, 34, 36, 37, 39	
\fi	... 6, 18, 21, 36, 37, 39, 52, 61, 88, 91, 100, 103, 112, 125, 127, 151, 152, 172, 173, 190, 193, 207	
<b>G</b>		
\gdef	..... 4	

<code>\global</code> . . . . . 21, 56, 71, 80, 90, 97, 102, 135, 141, 143, 155, 162, 164, 176, 182, 192		<code>\lengthof@input</code> . . . . . . . . 47, 48, 51	<b>R</b>
		<code>\lengthofA</code> . . . 51, 56, 60	<code>\relax</code> 2, 49, 58, 80, 90, 96, 97, 102, 140, 181
		<code>\let</code> . . . . . 14, 15, 26, 32, 49, 58, 80, 90, 97, 102, 120	<code>\resetassertions</code> . . . . . . . 129, 207
		<code>\long</code> . . . . . 30, 37, 38	<code>\result</code> . . . . . 10
<b>I</b>			<b>S</b>
<code>\if</code> . . . . . 95			<code>\showboxbreadth</code> . . . . 8
<code>\ifassertmessageonly</code> . . . . . 117, 148, 169, 187, 204	<b>M</b>		<code>\showboxdepth</code> . . . . . 8
<code>\ifdebug</code> . . . . . 16, 18	<code>\maxdimen</code> . . . . . 8		<code>\sp</code> . . . . . 196, 199, 201
<code>\ifDefined</code> . . . . 37, 110	<code>\meaning</code> 31, 137, 139, 157, 159, 178, 180		<code>\string</code> . . . . . 33
<code>\ifEmpty</code> . . . . . 19, 48, 57	<code>\medbreak</code> . . . . . . . . . . 149, 151, 170, 172, 188, 190, 205		<code>\summary</code> . . . 197, 203, 206
<code>\ifempty</code> . . . . . 19, 28	<code>\message</code> 137, 139, 144, 147, 157, 159, 165, 168, 178, 180, 183, 186, 203	<b>T</b>	
<code>\ifeq</code> . . . . . 68, 161		<code>\temp</code> . . . . . 43, 65, 69, 107, 118, 209	
<code>\iffalse</code> . . . . . 15, 23, 26, 34, 120		<code>\texassert.sty</code> . . . . 210	
<code>\ifnum</code> . . . . . 79	<b>N</b>	<code>\the</code> . . . . . 43, 69, 118, 123, 145, 166, 184, 199, 201	
<code>\iftrue</code> . . . . . 14, 34	<code>\n</code> . . . . . 11	<code>\then</code> . . . . . 23, 26, 27, 30, 37, 38, 48, 57, 110, 124, 126	
<code>\ifUndefined</code> . . . . 30, 38	<code>\newcount</code> . . . . . 11, 12, 42, 114, 115, 116	<code>\toks</code> . . . . . 121, 122, 123	
<code>\ifx</code> . . . . . 2, 21, 36, 78, 96, 140, 181	<code>\newif</code> . . . 16, 19, 68, 117	<code>\tokstemp</code> . . . . . 10	
<code>\immediate</code> . . . . . 18	<code>\newtoks</code> . . . . . 10	<code>\true</code> . . . 14, 34, 36, 37, 39	
<code>\import</code> . . . 1, 40, 66, 108	<code>\next</code> . . . . 35, 36, 49, 51, 53, 58, 60, 62, 80, 84, 90, 92, 97, 99, 102, 104	<b>U</b>	
<code>\import.tex</code> . . . . . 1		<code>\undefined</code> . . . . . 33	
<code>\indent</code> . . . 150, 171, 189	<code>\not</code> . . . . . 37	<code>\unexpected</code> 121, 125, 126	
<code>\input</code> . . . . . 3, 20, 21, 40, 66, 108, 210	<code>\number</code> . . . . . 76, 77	<b>W</b>	
<code>\integer</code> . . . . . 12		<code>\write</code> . . . . . 18	
<b>L</b>	<b>P</b>	<b>X</b>	
<code>\lena</code> . . . . . 76, 78	<code>\par</code> . . . . . 5	<code>\x</code> . . . . . 31, 36	
<code>\lenb</code> . . . . . 77, 78	<code>\ProvidesPackage</code> . . . . . . . 110, 111	<b>Y</b>	
<code>\length</code> . . . . . 42, 45, 56, 76, 77, 79		<code>\y</code> . . . . . 33, 36	
<code>\lengthof</code> . . . 40, 76, 77			
<code>\lengthof.tex</code> . . . . . 40			

## 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