

The `texassert` package*

Hanson Char
`hanson.char@gmail.com`

November 17, 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 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.
8. David Salomon. *The Advanced T_EXbook*. 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.  
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/tb139wermuth-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 \newcount\length
40 \def\lengthof#1{{\edef\x{#1}\global\length=0
41 \expandafter\lengthofA\x\end
42 }}
43 \def\lengthofA#1{\ifx#1\end\else
44 \global\advance\length by1
45 \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.

```

46 \input import \import{lengthof}
47
48 \newif\ifeq
49 \chardef\temp=\catcode'\catcode'\@=11
50
51 \global\eqtrue
52 % Assume no spaces
53 \def\checkeq#1#2{%
54 \edef\checkeq@fstparam{#1}%

```

```

55 \edef\checkeq@sndparam{#2}%
56 \lengthof\checkeq@fstparam \edef\lena{\number\length}%
57 \lengthof\checkeq@sndparam \edef\lenb{\number\length}%
58 \ifx\lena\lenb
59   \ifnum\length=0
60     \global\eqtrue \let\next=\relax
61   \else
62     \expandafter\expandafter\expandafter
63     \def\expandafter\expandafter\expandafter
64     \next\expandafter\expandafter\expandafter
65     {\expandafter\expandafter\expandafter
66      \checkeqA\expandafter\checkeq@fstparam
67      \expandafter\eot\checkeq@sndparam\eot}%
68   \fi
69 \else
70   \global\eqfalse \let\next=\relax
71 \fi
72 \next
73 }}
74 \def\checkeqA#1#2\eot#3#4\eot{%
75   \if#1#3{}% the trailing '{}' is necessary to avoid
76   \ifx\relax#2\relax % extra spaces
77     \global\eqtrue \let\next=\relax
78   \else
79     \def\next{\checkeqA#2\eot#4\eot}%
80   \fi
81 \else
82   \global\eqfalse \let\next=\relax
83 \fi
84 \next
85 }
86
87 \catcode'\@=\temp % restore the original catcode for @

```

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

```

88 \input import \import{checkeq} \import{common}
89
90 \ifDefined\ProvidesPackage\then
91   \ProvidesPackage{texassert}
92 \fi
93
94 \newcount\countassertions
95 \newcount\countassertionspassed
96 \newcount\countassertionsfailed
97 \newif\ifassertmessageonly
98 \chardef\temp=\catcode'\@ \catcode'\@=11
99
100 \let\assertDone=\iffalse
101 \def\unexpected{\toks0={unexpected!}}
102 \def\expected{\toks0={expected}}
103 \def\assert{\asserteq\the\toks0={expected}}
104 \def\assertTrue#1\assertDone{#1\then \expected
105   \else \unexpected\fi \assert}
106 \def\assertFalse#1\assertDone{#1\then \unexpected

```

```

107 \else \expected\fi \assert}
108
109 \def\resetassertions{%
110 \countassertions=0
111 \countassertionspassed=0
112 \countassertionsfailed=0
113 }

\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.
114 \def\asserteq#1=#2{%
115 \global\advance\countassertions by1
116 \edef\assert@a{#1}%
117 \edef\assert@b{#2}%
118 \ifx\assert@a\assert@b\relax\relax
119 \global\advance\countassertionspassed by1
120 \else
121 \global\advance\countassertionsfailed by1
122 \def\errmsg{*** assertion (\the\countassertions) failure:
123 '#1' not equal '#2' ***}%
124 \debug{\errmsg}%
125 \ifassertmessageonly\else
126 \medbreak
127 \indent\indent{\errmsg}%
128 \medbreak\fi
129 \fi
130 }}

\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.
131 \def\asserteqnocat#1=#2{%
132 \global\advance\countassertions by1
133 \edef\assert@a{#1}%
134 \edef\assert@b{#2}%
135 \checkeq\assert@a\assert@b
136 \ifeq
137 \global\advance\countassertionspassed by1
138 \else
139 \global\advance\countassertionsfailed by1
140 \def\errmsg{*** assertion (\the\countassertions) failure:
141 '#1' not equal '#2' ***}%
142 \debug{\errmsg}%
143 \ifassertmessageonly\else
144 \medbreak
145 \indent\indent{\errmsg}%
146 \medbreak\fi
147 \fi
148 }}

\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.

```

```

149 \def\assertneq#1=#2{%
150   \global\advance\countassertions by1
151   \edef\assert@a{#1}%
152   \edef\assert@b{#2}%
153   \ifx\assert@a\assert@b\relax\relax
154     \global\advance\countassertionsfailed by1
155     \def\errmsg{*** assertion (\the\countassertions) failure:
156       '#1' equal '#2' ***}%
157     \debug{\errmsg}%
158     \ifassertmessageonly\else
159       \medbreak
160       \indent\indent{\errmsg}%
161       \medbreak\fi
162   \else
163     \global\advance\countassertionspassed by1
164   \fi
165 }}

```

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

```

166 \def\assertionssummary{%
167   \def\summary{%
168     Assertion Summary:
169     \the\countassertionspassed/\the\countassertions\space
170     assertions passed i.e.
171     \the\countassertionsfailed/\the\countassertions\space
172     assertions failed.}%
173   \debug{\summary}%
174   \ifassertmessageonly\else
175     \medbreak
176     \summary
177   \fi\resetassertions}
178
179 \catcode'\@=\temp % restore the original catcode for @

```

`texassert.sty` Used for packaging purposes.

```

180 \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		<code>\assert@a</code> . 116, 118,	<code>\assertFalse</code> 106
<code>\advance</code> 44, 115, 119,		133, 135, 151, 153	<code>\assertionssummary</code> . 166
121, 132, 137,		<code>\assert@b</code> . 117, 118,	<code>\assertneq</code> 149
139, 150, 154, 163		134, 135, 152, 153	<code>\assertTrue</code> 104
<code>\aftergroup</code> 35		<code>\assertDone</code> 100, 104, 106	
<code>\assert</code> . . . 103, 105, 107		<code>\asserteq</code> 103, <u>114</u>	B
<code>\assert.tex</code> <u>88</u>		<code>\asserteqnocat</code> <u>131</u>	<code>\bool</code> 13, 14, 36, 38

C		
\catcode	49, 87, 98, 179	
\chardef	49, 98	
\checkeq	46, 135	
\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, 122, 132, 140, 150, 155, 169, 171	
\countassertionsfailed	96, 112, 121, 139, 154, 171	
\countassertionspassed	95, 111, 119, 137, 163, 169	
\csname	2, 4	
D		
\debug	15, 124, 142, 157, 173	
\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, 122, 131, 140, 149, 155, 166, 167	
E		
\e	31, 32	
\edef	30, 32, 40, 54, 55, 56, 57, 116, 117, 133, 134, 151, 152	
\else	20, 35, 36, 38, 43, 61, 69, 78, 81, 105, 107, 120, 125, 138, 143, 158, 162, 174	
\empty	20	
\emptyfalse	20	
\emptytrue	20	
\end	41, 43	
\endcsname	2, 4	
\eot	67, 74, 79	
\eqfalse	70, 82	
\eqtrue	51, 60, 77	
\errmsg	122, 124, 127, 140, 142, 145, 155, 157, 160	
\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, 128, 129, 146, 147, 161, 164, 177	
G		
\gdef	4	
\global	20, 40, 44, 51, 60, 70, 77, 82, 115, 119, 121, 132, 137, 139, 150, 154, 163	
I		
\if	75	
\ifassertmessageonly	97, 125, 143, 158, 174	
\ifdebug	15, 17	
\ifDefined	36, 90	
\ifEmpty	18	
\ifempty	18, 27	
\ifeq	48, 136	
\iffalse	14, 22, 25, 33, 100	
\ifnum	59	
\iftrue	13, 33	
\ifUndefined	29, 37	
\ifx	2, 20, 35, 43, 58, 76, 118, 153	
\immediate	17	
\import	1, 46, 88	
\import.tex	1	
\indent	127, 145, 160	
\input	3, 19, 20, 46, 88, 180	
\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	
\medbreak	126, 128, 144, 146, 159, 161, 175	
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, 118, 153	
\resetassertions	109, 177	
\result	9	
S		
\showboxbreadth	7	
\showboxdepth	7	
\space	169, 171	
\string	32	
\summary	167, 173, 176	
T		
\temp	49, 87, 98, 179	
\texassert.sty	180	
\the	103, 122, 140, 155, 169, 171	
\then	22, 25, 26, 29, 36, 37, 90, 104, 106	
\toks	101, 102, 103	
\tokstemp	9	
\true	13, 33, 35, 36, 38	

U	W	Y
<code>\undefined</code> 32	<code>\write</code> 17	<code>\y</code> 32, 35
X		
<code>\unexpected</code> 101, 105, 106	<code>\x</code> 30, 35, 40, 41	

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