The **texassert** package*

Hanson Char

`hanson.char@gmail.com`

November 10, 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 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
```

---

*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 something like:

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

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 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.

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/tb139wermuth-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\eot}%
52        \fi
53        \next
54      \egroup
55  }
56  \def\lengthofA#1#2\eot{\global\advance\length by1
57    \ifEmpty#2\then
58      \let\next=\relax
59    \else
60      \def\next{\lengthofA#2\eot}%
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\checkeq@fstparam{#1}%
75    \edef\checkeq@sndparam{#2}%
76    \lengthof\checkeq@fstparam \edef\lena{\number\length}%
77    \lengthof\checkeq@sndparam \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\checkeq@fstparam
87                  \expandafter\eot\checkeq@sndparam\eot}%
88      \fi
89    \else
90      \global\eqfalse \let\next=\relax
91    \fi
92    \next
93  }}
94  \def\checkeqA#1#2\eot#3#4\eot{%
95    \if#1#3{}% the trailing '{}%' is necessary to avoid
96      \ifx\relax#2\relax % extra spaces
```

4

```
97        \global\eqtrue \let\next=\relax
98      \else
99        \def\next{\checkeqA#2\eot#4\eot}%
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{checkeq}
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   \checkeq\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 }}
```

\assertionsummary   Typesets a summary of the assertions made. Then reset to a state as if no assertion

has been made.

```
195 \def\assertionsummary{{%
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.

Change History