

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

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.

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

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` 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 is critical
24 % 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 \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.

```
39 \input import \import{common}
40
41 \newcount\length
42 \edef\temp{\the\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` Used to check if two given strings are equal. Assume no space in the strings.

```
65 \input import \import{lengthof}
66
67 \newif\ifeq
68 \edef\temp{\the\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 adding extra spaces
95     \ifx\relax#2\relax
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{checked}
108
109 \ifDefined\ProvidesPackage\then
110   \ProvidesPackage{texassert}
111 \fi
112
113 \newcount\countassertions
114 \newcount\countassertionspassed
115 \newcount\countassertionsfailed
116 \newif\ifassertmessageonly
117 \edef\temp{\the\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 \else \unexpected\fi \assert}
124 \def\assertFalse#1\assertDone{#1\then \unexpected \else \expected\fi \assert}

```

```

125
126 \def\resetassertions{%
127   \countassertions=0
128   \countassertionspassed=0
129   \countassertionsfailed=0
130 }

```

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

```

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

```

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

```

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

```

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

```

172 \def\assertneq#1=#2{%

```

```

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

```

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

```

192 \def\assertionssummary{ {%
193   \def\sp{ }%
194   \def\summary{ %
195     Assertion Summary: \the\countassertionspassed/\the\countassertions\sp
196     assertions passed i.e.
197     \the\countassertionsfailed/\the\countassertions\sp assertions failed.}%
198   \message{\summary}%
199   \ifassertmessageonly\else
200   \medbreak
201   \summary
202   \fi}\resetassertions}
203
204 \catcode'\@=\temp % restore the original catcode for @

```

`texassert.sty` Used for packaging purposes.

```

205 \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.tex</code> 107	157 , 176 , 177 , 178
<code>\advance</code> 55 , 132 , 138 , 140 , 152 , 159 , 161 , 173 , 179 , 189	<code>\assert@a</code> . 133 , 134 , 137 , 153 , 154 , 157 , 174 , 175 , 178	<code>\assertDone</code> 119 , 123 , 124 <code>\asserteq</code> 122 , 131 <code>\asserteqnocat</code> 151
<code>\aftergroup</code> 36	<code>\assert@b</code> . 135 , 136 , 137 , 155 , 156 ,	<code>\assertFalse</code> 124 <code>\assertionssummary</code> . 192
<code>\assert</code> . . . 122 , 123 , 124		

<code>\assertneq</code>	172	139, 145, 160,	<code>\import.tex</code>	1		
<code>\assertTrue</code>	123	166, 184, 188, 199	<code>\indent</code>	147, 168, 186		
B						
<code>\bgroup</code>	45	<code>\empty</code>	21	<code>\input</code>	3, 20,	
<code>\bool</code>	14, 15, 37, 38	<code>\emptyfalse</code>	21	21, 39, 65, 107, 205		
C					<code>\integer</code>	12
<code>\catcode</code>	42,	<code>\emptytrue</code>	21	L		
64, 68, 106, 117, 204		<code>\endcsname</code>	2, 4	<code>\lena</code>	75, 77	
<code>\checkeq</code>	65, 157	<code>\eot</code> 50, 55, 59, 86, 93, 98		<code>\lenb</code>	76, 77	
<code>\checkeq.tex</code>	65	<code>\eqfalse</code>	89, 101	<code>\length</code>	41,	
<code>\checkeq@fstparam</code> .	73, 75, 85	<code>\eqtrue</code>	70, 79, 96	44, 55, 75, 76, 78		
<code>\checkeq@sndparam</code> .	74, 76, 86	<code>\errmsg</code>	142, 144,	<code>\lengthof</code>	39, 75, 76	
<code>\checkeqA</code>	85, 93, 98	147, 163, 165,		<code>\lengthof.tex</code>	39	
<code>\checkifempty</code>	20, 28	168, 181, 183, 186		<code>\lengthof@input</code> . . .	46, 47, 50	
<code>\common.tex</code>	8	<code>\escapechar</code>	32, 33	<code>\lengthofA</code>	50, 55, 59	
<code>\countassertions</code> . .	113, 127, 132,	<code>\expandafter</code>	2,	<code>\let</code>	14, 15,	
142, 152, 163,		4, 20, 35, 50, 81,		26, 32, 48, 57,		
173, 181, 195, 197		82, 83, 84, 85, 86		79, 89, 96, 101, 119		
<code>\countassertionsfailed</code>	115, 129,	<code>\expected</code>	121, 123, 124	<code>\long</code>	30, 37, 38	
140, 161, 179, 197		F				
<code>\countassertionspassed</code>	114, 128,	<code>\false</code>	15, 34, 36, 37, 38	M		
138, 159, 189, 195		<code>\fi</code>	6, 18, 21, 36,	<code>\maxdimen</code>	8	
<code>\csname</code>	2, 4	37, 38, 51, 60,		<code>\meaning</code> 31, 134, 136,		
D					154, 156, 175, 177	
<code>\debug</code>	16	87, 90, 99, 102,		<code>\medbreak</code>	146, 148, 167,	
<code>\debugtrue</code>	17	111, 123, 124,		169, 185, 187, 200		
<code>\def</code> 1, 14, 15, 18, 20,		148, 149, 169,		<code>\message</code> 134, 136, 141,		
27, 30, 34, 35,		170, 187, 190, 202		144, 154, 156,		
37, 38, 44, 50,		G			162, 165, 175,	
55, 59, 72, 82,		<code>\gdef</code>	4	177, 180, 183, 198		
93, 98, 120, 121,		<code>\global</code>	21, 55,	N		
122, 123, 124,		70, 79, 89, 96,		<code>\n</code>	11	
126, 131, 142,		101, 132, 138,		<code>\newcount</code>	11,	
151, 163, 172,		140, 152, 159,		12, 41, 113, 114, 115		
181, 192, 193, 194		161, 173, 179, 189		<code>\newif</code>	16, 19, 67, 116	
E					<code>\newtoks</code>	10
<code>\e</code>	32, 33	I			<code>\next</code>	35, 36, 48,
<code>\edef</code> 31, 33, 42, 46, 68,		<code>\if</code>	94	50, 52, 57, 59,		
73, 74, 75, 76,		<code>\ifassertmessageonly</code>	116,	61, 79, 83, 89,		
117, 133, 135,		145, 166, 184, 199		91, 96, 98, 101, 103		
153, 155, 174, 176		<code>\ifdebug</code>	16, 18	<code>\not</code>	37	
<code>\egroup</code>	53	<code>\ifDefined</code>	37, 109	<code>\number</code>	75, 76	
<code>\else</code> 21, 36, 37, 38, 49,		<code>\ifEmpty</code>	19, 47, 56	P		
58, 80, 88, 97,		<code>\ifempty</code>	19, 28	<code>\par</code>	5	
100, 123, 124,		<code>\ifeq</code>	67, 158	<code>\ProvidesPackage</code> . .	109, 110	
		<code>\iffalse</code>	15, 23, 26, 34, 119	R		
		<code>\ifnum</code>	78	<code>\relax</code> 2, 48, 57, 79, 89,	95, 96, 101, 137, 178	
		<code>\iftrue</code>	14, 34			
		<code>\ifUndefined</code>	30, 38			
		<code>\ifx</code>	2, 21,			
		36, 77, 95, 137, 178				
		<code>\immediate</code>	18			
		<code>\import</code>	1, 39, 65, 107			

