

# The lt3luabridge package: Lua without LuaTeX

Vít Novotný\*

Released 2022-08-16

The `lt3luabridge` `expl3` [2] package provides support for executing Lua code in LuaTeX or any other TeX engine that exposes the shell. The package provides interfaces to plain TeX, L<sup>A</sup>TeX, and ConTeXt formats:

```
\documentclass{standalone}
\usepackage{lt3luabridge}
\begin{document}
$ 1 + 2 = \luabridgeExecute{ print(1 + 2) } $
\end{document}
```

The package was previously part of the Markdown package [1], where it has been battle-tested since 2016. Since 2022, `lt3luabridge` has also been available as a separate package.

## 1 Loading the package

Use the `\input lt3luabridge\relax` command to load the package from plain TeX, use the `\usepackage{lt3luabridge}` command to load the package from L<sup>A</sup>TeX, and use the `\usemodule[t]{lt3luabridge}` command to load the package from ConTeXt.

## 2 Executing Lua code

The interface for executing Lua code mimics the `\lua_now:n` function from `l3luatex`.

---

`\luabridge_now:n` `\luabridge_now:n`  $\langle token list \rangle$

`\luabridge_now:e`

New: 2022-06-26

Updated: 2022-07-31

---

The  $\langle token list \rangle$  is first tokenized by TeX, which includes converting line ends to spaces in the usual TeX manner and which respects currently-applicable TeX category codes. The resulting  $\langle Lua input \rangle$  is passed to the Lua interpreter for processing. Each `\luabridge_now:n` block is treated by Lua as a separate chunk. The Lua interpreter executes the  $\langle Lua input \rangle$  immediately, and in an expandable manner.

Unlike `\lua_now:n`, `\luabridge_now:n` may execute  $\langle Lua input \rangle$  in a separate process from TeX. Therefore, you should not interact with TeX from  $\langle Lua input \rangle$  or create global variables. The only exception is the standard output produced by the `print()` Lua function like in the example at the top of this page. The standard output of `print()` will be inserted into TeX's input stream.

---

`\luabridgeExecute` `\luabridgeExecute`  $\langle token list \rangle$

New: 2022-06-26

Updated: 2022-07-31

---

The `\luabridgeExecute` document command aliases the `\luabridge_now:e` function.

---

\*E-mail: [witiko@mail.muni.cz](mailto:witiko@mail.muni.cz)

### 3 Setting and getting the method to execute Lua code

There are several methods that can be used to execute Lua code. This section describes the interface that the package provides to set the preferred method or to determine which method was used.

---

`\g_luabridge_method_int` This variable controls the method used to execute Lua code. The variable is set automatically when the package is loaded and changing the value of the variable afterwards has no effect. However, we can set the value of the variable before loading the package to one of the constants described below.

---

New: 2022-06-26

---

`\c_luabridge_method_shell_int`

---

New: 2022-07-31

Use shell escape through the `\write18` TeX command to execute Lua code.

---

`\c_luabridge_method_directlua_int`

---

New: 2022-06-26

Use the `\directlua` primitive of LuaTeX to execute Lua code.

### 4 Setting and getting the filenames of helper files

When shell escape is used to execute Lua code, several helper files are needed to shuffle around code and output. The following variables and constants are undefined when the `\directlua` primitive of LuaTeX is used to execute Lua code.

---

`\g_luabridge_output_dirname_str`

---

New: 2022-06-26

This variable controls the output directory that will store the helper files. The variable should be set to the same value as the `-output-directory` parameter of the TeX engine.

---

`\c_luabridge_default_output_dirname_str`

---

New: 2022-06-26

This constant is the default value of `\g_luabridge_output_dirname_str`.

---

`\g_luabridge_helper_script_filename_str`

---

New: 2022-06-26

This variable controls the filename of a helper Lua script that will be executed from the shell using the TeX Lua interpreter.

---

`\c_luabridge_default_helper_script_filename_str`

---

New: 2022-06-26

This constant is the default value of `\g_luabridge_helper_script_filename_str`.

---

---

`\g_luabridge_error_output_filename_str`

---

New: 2022-06-26

This variable controls the filename of a helper file that will contain the error output produced by the `texlua` interpreter (if any).

---

---

`\c_luabridge_default_error_output_filename_str`

---

New: 2022-06-26

This constant is the default value of `\g_luabridge_error_output_filename_str`.

## 5 Plain T<sub>E</sub>X implementation

This section contains the implementation for plain T<sub>E</sub>X using generic `expl3`.

```
1 <@@=luabridge>
2 <*generic-package>
3 \ifx\ExplSyntaxOn\undefined
4   \input expl3-generic\relax
5 \fi
6 \ExplSyntaxOn
7 \int_const:Nn
8   \c_luabridge_method_directlua_int
9   { 0 }
10 \int_const:Nn
11   \c_luabridge_method_shell_int
12   { 1 }
13 \int_if_exist:NF
14   \g_luabridge_method_int
15   {
16     \int_new:N
17       \g_luabridge_method_int
18     \sys_if_engine luatex:TF
19       {
20         \int_gset_eq:NN
21           \g_luabridge_method_int
22           \c_luabridge_method_directlua_int
23       }
24       {
25         \int_gset_eq:NN
26           \g_luabridge_method_int
27           \c_luabridge_method_shell_int
28       }
29   }
30 \msg_new:nnn
31   { luabridge }
32   { method-shell }
33   {
34     Using-shell-escape-as-the-bridging-method
35   }
36 \msg_new:nnn
37   { luabridge }
38   { method-directlua }
```

```

39  {
40    Using~direct~Lua~access~as~the~bridging~method
41  }
42  \msg_new:nnn
43  { luabridge }
44  { unknown-method }
45  {
46    Unknown~bridging~method:~#1
47  }
48  \int_case:nnF
49  { \g_luabridge_method_int }
50  {
51    { \c_luabridge_method_shell_int }
52    {
53      \msg_info:nn
54      { luabridge }
55      { method-shell }
56    }
57    { \c_luabridge_method_directlua_int }
58    {
59      \msg_info:nn
60      { luabridge }
61      { method-directlua }
62    }
63  }
64  {
65    \cs_generate_variant:Nn
66    \msg_error:nnn
67    { nnV }
68    \msg_error:nnV
69    { luabridge }
70    { unknown-method }
71    \g_luabridge_method_int
72  }
73  \int_compare:nNnT
74  { \g_luabridge_method_int }
75  =
76  { \c_luabridge_method_shell_int }
77  {
78    \str_const:Nn
79    \c_luabridge_default_output_dirname_str
80    { . }
81    \str_const:Nx
82    \c_luabridge_default_helper_script_filename_str
83    { \jobname.luabridge.lua }
84    \str_const:Nx
85    \c_luabridge_default_error_output_filename_str
86    { \jobname.luabridge.err }
87    \str_if_exist:NF
88    \g_luabridge_output_dirname_str
89    {
90      \str_new:N
91      \g_luabridge_output_dirname_str
92      \tl_gset:Nn

```

```

93     \g_luabridge_output_dirname_str
94     \c_luabridge_default_output_dirname_str
95 }
96 \str_if_exist:NF
97   \g_luabridge_helper_script_filename_str
98   {
99     \str_gset_eq:NN
100     \g_luabridge_helper_script_filename_str
101     \c_luabridge_default_helper_script_filename_str
102   }
103 \str_if_exist:NF
104   \g_luabridge_error_output_filename_str
105   {
106     \str_gset_eq:NN
107     \g_luabridge_error_output_filename_str
108     \c_luabridge_default_error_output_filename_str
109   }
110 \cs_new:Nn
111   \luabridge_now:n
112   {
113     \iow_open:NV
114     \g_tmpa_iow
115     \g_luabridge_helper_script_filename_str
116     \msg_info:nnV
117     { luabridge }
118     { writing-helper-script }
119     \g_luabridge_helper_script_filename_str
120     \tl_set:Nn
121     \l_tmpa_tl
122     { #1 }
123     \tl_set:Nx
124     \l_tmpb_tl
125     {
126       local~ran_ok, error = pcall(function()
127         local~ran_ok, kpse = pcall(require,~"kpse")
128         if~ran_ok~then~kpse.set_program_name("luatex") end~
129         \exp_not:V \l_tmpa_tl
130         \iow_newline:
131       end)
132       if~not~ran_ok~then~
133         local~file = io.open("
134           \g_luabridge_output_dirname_str /
135           \g_luabridge_error_output_filename_str
136           ", "w")
137         if~file~then~
138           file:write(error .. " \iow_char:N \\ n ")
139           file:close()
140         end~
141         print('
142           \iow_char:N \\ \iow_char:N \\ begingroup
143           \iow_char:N \\ \iow_char:N \\ ExplSyntaxOn
144           \iow_char:N \\ \iow_char:N \\ msg_error:nnv
145           { luabridge }
146           { failed-to-execute }

```

```

147         { g_luabridge_output_dirname_str }
148         { g_luabridge_error_output_filename_str }
149         \iow_char:N \ \ \iow_char:N \ \ endgroup
150     ')
151     end
152 }
153 \iow_now:NV
154 \g_tmpa_iow
155 \l_tmpb_tl
156 \iow_close:N
157 \g_tmpa_iow
158 \msg_info:nnV
159 { luabridge }
160 { executing-helper-script }
161 \g_luabridge_helper_script_filename_str
162 \sys_get_shell:xnNTF
163 {
164     texlua~
165     \g_luabridge_output_dirname_str /
166     \g_luabridge_helper_script_filename_str
167 }
168 { }
169 \l_tmpa_tl
170 {
171     \l_tmpa_tl
172 }
173 {
174     \msg_error:nn
175     { luabridge }
176     { level-disabled }
177 }
178 }
179 \prg_generate_conditional_variant:Nnn
180 \sys_get_shell:nnN
181 { xnN }
182 { TF }
183 \cs_generate_variant:Nn
184 \msg_info:nnn
185 { nnV }
186 \cs_generate_variant:Nn
187 \msg_error:nnnn
188 { nnvv }
189 \cs_generate_variant:Nn
190 \iow_open:Nn
191 { NV }
192 \cs_generate_variant:Nn
193 \iow_now:Nn
194 { NV }
195 \msg_new:nnn
196 { luabridge }
197 { writing-helper-script }
198 {
199     Writing-a-helper-Lua-script-to-file-#1
200 }

```

```

201 \msg_new:nnn
202   { luabridge }
203   { executing-helper-script }
204   {
205     Executing~a~helper~Lua~script~from~file~#1
206   }
207 \msg_new:nnnn
208   { luabridge }
209   { failed-to-execute }
210   {
211     An~error~was~encountered~while~executing~Lua~code
212   }
213   {
214     For further clues, examine file #1 / #2
215   }
216 \msg_new:nnnn
217   { luabridge }
218   { level-disabled }
219   {
220     Shell~escape~seems~to~be~disabled
221   }
222   {
223     You~may~need~to~run~TeX~with~the~--shell-escape~or~the~
224     --enable-write18~flag,~or~write~shell_escape=t~in~the~
225     texmf.cnf~file.
226   }
227 }
228 \int_compare:nNnT
229 { \g_luabridge_method_int }
230 =
231 { \c_luabridge_method_directlua_int }
232 {
233   \cs_new:Nn
234     \luabridge_now:n
235     {
236       \tl_set:Nn
237         \l_tmpa_tl
238         { #1 }
239       \tl_set:Nx
240         \l_tmpb_tl
241         {
242           _ENV = setmetatable({}, {__index = _ENV})
243           local~function~print(input)
244             input = tostring(input)
245             local~output = {}
246             for~line~in~input:gmatch("[^
247               \iow_char:N \\ r
248               \iow_char:N \\ n
249             ]+") do~
250               table.insert(output, line)
251             end~
252             tex.print(output)
253           end~
254           \exp_not:V \l_tmpa_tl

```

```

255     }
256     \lua_now:V
257     \l_tmpb_tl
258   }
259   \cs_generate_variant:Nn
260     \lua_now:n
261     { V }
262 }
263 \cs_new_protected:Npn
264   \luabridgeExecute
265   #1
266   {
267     \luabridge_now:e
268     { #1 }
269   }
270 \cs_generate_variant:Nn
271   \luabridge_now:n
272   { e }
273 \ExplSyntaxOff
274 </generic-package>

```

## 6 L<sup>A</sup>T<sub>E</sub>X implementation

This section contains the implementation for L<sup>A</sup>T<sub>E</sub>X.

```

275 <*latex-package>
276 \RequirePackage{expl3}
277 \ProvidesExplPackage
278   {lt3luabridge}%
279   {2022-08-16}%
280   {2.0.1}%
281   {An expl3 package that allows you to execute Lua code in LuaTeX or any other
282     TeX engine that exposes the shell}
283 \input lt3luabridge\relax
284 </latex-package>

```

## 7 ConT<sub>E</sub>Xt implementation

This section contains the implementation for ConT<sub>E</sub>Xt. ConT<sub>E</sub>Xt MkII, MkIV, and later formats are supported.

```

285 <*context-package>
286 \writestatus{loading}{ConTeXt User Module / lt3luabridge}
287 \startmodule[lt3luabridge]
288 \unprotect
289 \input lt3luabridge\relax
290 </context-package>

```



## References

- [1] Vít Novotný. *Markdown. A package for converting and rendering markdown documents inside T<sub>E</sub>X*. Version 2.15.2-0-gb238dbc. May 31, 2022. URL: <https://ctan.org/pkg/markdown> (visited on 06/26/2022).
- [2] The L<sup>A</sup>T<sub>E</sub>X Team. *expl3. Wrapper package for experimental L<sup>A</sup>T<sub>E</sub>X3*. June 16, 2022. URL: <https://ctan.org/pkg/expl3> (visited on 06/26/2022).

## Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

<b>Symbols</b>	
<code>\</code> . . . . .	138, 142, 143, 144, 149, 247, 248
<b>C</b>	
cs commands:	
<code>\cs_generate_variant:Nn</code> . . . . .	65, 183, 186, 189, 192, 259, 270
<code>\cs_new:Nn</code> . . . . .	110, 233
<code>\cs_new_protected:Npn</code> . . . . .	263
<b>D</b>	
<code>\directlua</code> . . . . .	2
<b>E</b>	
exp commands:	
<code>\exp_not:n</code> . . . . .	129, 254
<code>\ExplSyntaxOff</code> . . . . .	273
<code>\ExplSyntaxOn</code> . . . . .	3, 6
<b>F</b>	
<code>\fi</code> . . . . .	5
<b>I</b>	
<code>\ifx</code> . . . . .	3
<code>\input</code> . . . . .	4, 283, 289
int commands:	
<code>\int_case:nnTF</code> . . . . .	48
<code>\int_compare:nNnTF</code> . . . . .	73, 228
<code>\int_const:Nn</code> . . . . .	7, 10
<code>\int_gset_eq:NN</code> . . . . .	20, 25
<code>\int_if_exist:NTF</code> . . . . .	13
<code>\int_new:N</code> . . . . .	16
iow commands:	
<code>\iow_char:N</code> . . . . .	138, 142, 143, 144, 149, 247, 248
<code>\iow_close:N</code> . . . . .	156
<code>\iow_newline:</code> . . . . .	130
<code>\iow_now:Nn</code> . . . . .	153, 193
<code>\iow_open:Nn</code> . . . . .	113, 190
<code>\g_tmpa_iow</code> . . . . .	114, 154, 157
<b>J</b>	
<code>\jobname</code> . . . . .	83, 86
<b>L</b>	
lua commands:	
<code>\lua_now:n</code> . . . . .	1, 256, 260
luabridge commands:	
<code>\c_luabridge_default_error_output_filename_str</code> . . . . .	3, 85, 108
<code>\c_luabridge_default_helper_script_filename_str</code> . . . . .	2, 82, 101
<code>\c_luabridge_default_output_dirname_str</code> . . . . .	2, 79, 94
<code>\g_luabridge_error_output_filename_str</code> . . . . .	3, 104, 107, 135
<code>\g_luabridge_helper_script_filename_str</code> . . . . .	2, 97, 100, 115, 119, 161, 166
<code>\c_luabridge_method_directlua_int</code> . . . . .	2, 8, 22, 57, 231
<code>\g_luabridge_method_int</code> . . . . .	2, 14, 17, 21, 26, 49, 71, 74, 229
<code>\c_luabridge_method_shell_int</code> . . . . .	2, 11, 27, 51, 76
<code>\luabridge_now:n</code> . . . . .	1, 111, 234, 267, 271
<code>\g_luabridge_output_dirname_str</code> . . . . .	2, 88, 91, 93, 134, 165
<code>\luabridgeExecute</code> . . . . .	1, 264
<b>M</b>	
msg commands:	
<code>\msg_error:nn</code> . . . . .	174
<code>\msg_error:nnn</code> . . . . .	66, 68
<code>\msg_error:nnnn</code> . . . . .	187

<code>\msg_info:nn</code> .....	53, 59	<code>\str_new:N</code> .....	90
<code>\msg_info:nnn</code> .....	116, 158, 184	sys commands:	
<code>\msg_new:nnn</code> .....	30, 36, 42, 195, 201	<code>\sys_get_shell:nnN</code> .....	180
<code>\msg_new:nnnn</code> .....	207, 216	<code>\sys_get_shell:nnNTF</code> .....	162
		<code>\sys_if_engine_luatex:TF</code> .....	18
<b>P</b>			
prg commands:			
<code>\prg_generate_conditional_</code>		<b>T</b>	
<code>variant:Nnn</code> .....	179	tl commands:	
<code>\ProvidesExplPackage</code> .....	277	<code>\tl_gset:Nn</code> .....	92
		<code>\tl_set:Nn</code> .....	120, 123, 236, 239
<b>R</b>			
<code>\relax</code> .....	4, 283, 289	<code>\l_tmpa_tl</code> .	121, 129, 169, 171, 237, 254
<code>\RequirePackage</code> .....	276	<code>\l_tmpb_tl</code> .....	124, 155, 240, 257
<b>S</b>			
<code>\startmodule</code> .....	287	<b>U</b>	
str commands:			
<code>\str_const:Nn</code> .....	78, 81, 84	<code>\undefined</code> .....	3
<code>\str_gset_eq:NN</code> .....	99, 106	<code>\unprotect</code> .....	288
<code>\str_if_exist:NTF</code> .....	87, 96, 103	<b>W</b>	
		<code>\write18</code> .....	2
		<code>\writestatus</code> .....	286