

The lt3luabridge package: Lua without LuaTeX

Vít Novotný*

Released 2022-10-24

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, LaTeX, 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 LaTeX, 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

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_EX implementation

This section contains the implementation for plain T_EX 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      \str_gset_eq: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

```

Escape " and \ in the Lua code, so that we can represent it as a double-quoted string that we can pass into the load() Lua built-in and fail gracefully if the Lua code fails to compile.

```

120   \tl_set:Nx
121     \l_tmpa_tl
122     { \tl_to_str:n { #1 } }
123   \regex_replace_all:nnN
124     { [\\"] }
125     { \\ \0 }
126     \l_tmpa_tl
127   \tl_set:Nx
128     \l_tmpa_tl
129     {
130       local~ran_ok, err = pcall(function()
131         local~ran_ok, kpse = pcall(require, ~"kpse")
132         if~ran_ok~then~kpse.set_program_name("luatex") end~
133         assert(load(" \exp_not:V \l_tmpa_tl " ))()
134       end)
135       if~not~ran_ok~then~
136         local~file = io.open("
137           \g_luabridge_output_dirname_str /
138           \g_luabridge_error_output_filename_str
139           ", "w")
140         if~file~then~
141           file:write(err .. " \iow_char:N \\ n ")
142           file:close()

```

```

143         end~
144     print('
145         \iow_char:N \ \ \iow_char:N \ \ begingroup
146         \iow_char:N \ \ \iow_char:N \ \ ExplSyntaxOn
147         \iow_char:N \ \ \iow_char:N \ \ csname~
148         msg_error:nnvv\iow_char:N \ \ \iow_char:N \ \ endcsname
149         { luabridge }
150         { failed-to-execute }
151         { g_luabridge_output_dirname_str }
152         { g_luabridge_error_output_filename_str }
153         \iow_char:N \ \ \iow_char:N \ \ endgroup
154     ')
155     end
156 }
157 \iow_now:NV
158 \g_tmpa_iow
159 \l_tmpa_tl
160 \iow_close:N
161 \g_tmpa_iow
162 \msg_info:nnV
163 { luabridge }
164 { executing-helper-script }
165 \g_luabridge_helper_script_filename_str
166 \sys_get_shell:xnNTF
167 {
168     texlua~
169     \g_luabridge_output_dirname_str /
170     \g_luabridge_helper_script_filename_str
171 }
172 { }
173 \l_tmpa_tl
174 {
175     \l_tmpa_tl
176 }
177 {
178     \msg_error:nn
179     { luabridge }
180     { level-disabled }
181 }
182 }
183 \prg_generate_conditional_variant:Nnn
184 \sys_get_shell:nnN
185 { xnN }
186 { TF }
187 \cs_generate_variant:Nn
188 \msg_info:nnn
189 { nnV }
190 \cs_generate_variant:Nn
191 \msg_error:nnnn
192 { nnvv }
193 \cs_generate_variant:Nn
194 \iow_open:Nn
195 { NV }
196 \cs_generate_variant:Nn

```

```

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

```

```

251         \iow_char:N \ r
252         \iow_char:N \ n
253         ]+") do~
254         table.insert(output, line)
255     end~
256     tex.print(output)
257 end~
258 \exp_not:V \l_tmpa_tl
259 }
260 \lua_now:V
261 \l_tmpa_tl
262 }
263 \cs_generate_variant:Nn
264 \lua_now:n
265 { V }
266 }
267 \cs_new_protected:Npn
268 \luabridgeExecute
269 #1
270 {
271     \luabridge_now:e
272     { #1 }
273 }
274 \cs_generate_variant:Nn
275 \luabridge_now:n
276 { e }
277 \ExplSyntaxOff
278 </generic-package>

```

6 L^AT_EX implementation

This section contains the implementation for L^AT_EX.

```

279 <*latex-package>
280 \RequirePackage{expl3}
281 \ProvidesExplPackage
282 {lt3luabridge}%
283 {2022-10-24}%
284 {2.0.2}%
285 {An expl3 package that allows you to execute Lua code in LuaTeX or any other
286 TeX engine that exposes the shell}
287 \input lt3luabridge\relax
288 </latex-package>

```

7 ConT_EXt implementation

This section contains the implementation for ConT_EXt. ConT_EXt MkII, MkIV, and later formats are supported.

```

289 <*context-package>
290 \writestatus{loading}{ConTeXt User Module / lt3luabridge}
291 \startmodule[lt3luabridge]
292 \unprotect

```

293 `\input lt3luabridge\relax`
 294 `\</context-package>`

References

- [1] Vít Novotný. *Markdown. A package for converting and rendering markdown documents inside T_EX*. Version 2.15.2-0-gb238dbc. May 31, 2022. URL: <https://ctan.org/pkg/markdown> (visited on 06/26/2022).
- [2] The L^AT_EX Team. *expl3. Wrapper package for experimental L^AT_EX3*. 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.

	Symbols		
<code>\</code>	124, 125, 141, 145, 146, 147, 148, 153, 251, 252	
	Numbers		
<code>\0</code>	125	
	C		
cs commands:			
	<code>\cs_generate_variant:Nn</code>	65, 187, 190, 193, 196, 263, 274
	<code>\cs_new:Nn</code>	110, 237
	<code>\cs_new_protected:Npn</code>	267
	D		
<code>\directlua</code>	2	
	E		
exp commands:			
	<code>\exp_not:n</code>	133, 258
	<code>\ExplSyntaxOff</code>	277
	<code>\ExplSyntaxOn</code>	3, 6
	F		
<code>\fi</code>	5	
	I		
<code>\ifx</code>	3	
<code>\input</code>	4, 287, 293	
int commands:			
	<code>\int_case:nnTF</code>	48
	<code>\int_compare:nNnTF</code>	73, 232
	<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>	141, 145, 146, 147, 148, 153, 251, 252
	<code>\iow_close:N</code>	160
	<code>\iow_now:Nn</code>	157, 197
	<code>\iow_open:Nn</code>	113, 194
	<code>\g_tmpa_iow</code>	114, 158, 161
	J		
<code>\jobname</code>	83, 86	
	L		
lua commands:			
	<code>\lua_now:n</code>	1, 260, 264
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, 138
	<code>\g_luabridge_helper_script_- filename_str</code>	2, 97, 100, 115, 119, 165, 170
	<code>\c_luabridge_method_directlua_- int</code>	2, 8, 22, 57, 235
	<code>\g_luabridge_method_int</code>	2, 14, 17, 21, 26, 49, 71, 74, 233
	<code>\c_luabridge_method_shell_int</code>	...	2, 11, 27, 51, 76

\luabridge_now:n	1, 111, 238, 271, 275		
\g_luabridge_output_dirname_str	2, 88, 91, 93, 137, 169		
\luabridgeExecute	1, 268		
M			
msg commands:			
\msg_error:nn	178		
\msg_error:nnn	66, 68		
\msg_error:nmnn	191		
\msg_info:nn	53, 59		
\msg_info:nnn	116, 162, 188		
\msg_new:nnn	30, 36, 42, 199, 205		
\msg_new:nmnn	211, 220		
P			
prg commands:			
\prg_generate_conditional_-variant:Nnn	183		
\ProvidesExplPackage	281		
R			
regex commands:			
\regex_replace_all:nnN	123		
\relax	4, 287, 293		
\RequirePackage	280		
S			
\startmodule	291		
str commands:			
\str_const:Nn	78, 81, 84		
\str_gset_eq:NN	92, 99, 106		
\str_if_exist:NTF	87, 96, 103		
\str_new:N	90		
sys commands:			
\sys_get_shell:nnN	184		
\sys_get_shell:nnNTF	166		
\sys_if_engine luatex:TF	18		
T			
tl commands:			
\tl_set:Nn	120, 127, 240, 243		
\tl_to_str:n	122		
\l_tmpa_tl	121, 126, 128, 133, 159, 173, 175, 241, 244, 258, 261		
U			
\undefined	3		
\unprotect	292		
W			
\write18	2		
\writestatus	290		