

The lt3luabridge package: Lua without LuaTeX

Vít Starý Novotný*

Released 2025-06-24

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, L^ATeX, 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^ATeX, 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.

*E-mail: witiko@mail.muni.cz

`\luabridgeExecute` `\luabridgeExecute` $\{\langle token list \rangle\}$
New: 2022-06-26 The `\luabridgeExecute` document command aliases the `\luabridge_now:e` function.
Updated: 2022-07-31

`\luabridge_tl_set:Nn` `\luabridge_tl_set:Nn` $\langle tl var \rangle \{\langle token list \rangle\}$
New: 2024-02-14 Like `\lua_now:n` but the result of executing the Lua code is stored in $\langle tl var \rangle$ instead of being inserted into \TeX 's input stream.

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.

`\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 $\text{Lua}\TeX$ 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 $\text{Lua}\TeX$ 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
Updated: 2024-07-03

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`.
None of the variables from this section may contain spaces.

5 Plain `TEX` implementation

This section contains the implementation for plain `TEX` using generic `expl3`.

```
1 <@@=luabridge>
2 <*generic-package>
3 \expandafter\ifx\csname ExplSyntaxOn\endcsname\relax
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 {

```

Instead of assuming the current working directory as the output directory, try to determine the value of ‘-output-directory’ from the environmental variable `TEXMF_OUTPUT_DIRECTORY`,

which is automatically defined by T_EX engines and accessible from child processes.

Note that this environmental variable is **not currently set by MikT_EX**. Therefore, when shell escape is used to execute Lua code, the value of the variable `\g_luabridge_output_dirname_str` won't be correctly determined by MikT_EX, unless you a) manually set the environmental variable `TEXMF_OUTPUT_DIRECTORY` to the same value as `-output-directory` or b) do not set `-output-directory`. If neither a) or b) is an option for you, you may also c) manually set the variable `\g_luabridge_output_dirname_str` to the same value as `-output-directory`.

```

78   \sys_if_platform_unix:TF
79   {
80     \str_const:Nn
81     \c_luabridge_default_output_dirname_str
82     { $TEXMF_OUTPUT_DIRECTORY }
83   }
84   {
85     \sys_if_platform_windows:TF
86     {
87       \str_set:Nn
88       \l_tmpa_str
89       { TEXMF_OUTPUT_DIRECTORY }
90       \str_put_left:NV
91       \l_tmpa_str
92       \c_percent_str
93       \str_put_right:NV
94       \l_tmpa_str
95       \c_percent_str
96       \str_const:NV
97       \c_luabridge_default_output_dirname_str
98       \l_tmpa_str
99     }
100    {
101      \str_const:Nn
102      \c_luabridge_default_output_dirname_str
103      { . }
104    }
105  }

```

The constants `\c_luabridge_default_helper_script_filename_str` and `\c_luabridge_default_error_output_filename_str` are derived from `\jobname` by default. Since these constants may not contain spaces, `\jobname` must also not contain spaces, i.e. the filenames of your `.tex` files must not contain spaces.

```

106   \str_const:Nx
107   \c_luabridge_default_helper_script_filename_str
108   { \jobname.luabridge.lua }
109   \str_const:Nx
110   \c_luabridge_default_error_output_filename_str
111   { \jobname.luabridge.err }
112   \str_if_exist:NF
113   \g_luabridge_output_dirname_str
114   {
115     \str_new:N
116     \g_luabridge_output_dirname_str
117     \str_gset_eq:NN

```

```

118     \g_luabridge_output_dirname_str
119     \c_luabridge_default_output_dirname_str
120 }
121 \str_if_exist:NF
122   \g_luabridge_helper_script_filename_str
123   {
124     \str_gset_eq:NN
125     \g_luabridge_helper_script_filename_str
126     \c_luabridge_default_helper_script_filename_str
127   }
128 \str_if_exist:NF
129   \g_luabridge_error_output_filename_str
130   {
131     \str_gset_eq:NN
132     \g_luabridge_error_output_filename_str
133     \c_luabridge_default_error_output_filename_str
134   }
135 \cs_new:Nn
136   \luabridge_tl_set:Nn
137   {
138     \iow_open:NV
139     \g_tmpa_iow
140     \g_luabridge_helper_script_filename_str
141     \msg_info:nnV
142     { luabridge }
143     { writing-helper-script }
144     \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.

```

145   \tl_set:Nx
146     \l_tmpa_tl
147     { \tl_to_str:n { #2 } }
148   \regex_replace_all:nnN
149     { [\\"] }
150     { \\ \0 }
151     \l_tmpa_tl
152   \tl_set:Nx
153     \l_tmpa_tl
154     {
155     local~ran_ok, err = pcall(function()
156       local~ran_ok, kpse = pcall(require, ~"kpse")
157       if~ran_ok~then~kpse.set_program_name("luatex") end~
158       assert(load(" \exp_not:V \l_tmpa_tl " ))()
159     end)
160     if~not~ran_ok~then~
161       local~file = io.open("
162         \g_luabridge_output_dirname_str /
163         \g_luabridge_error_output_filename_str
164         ", "w")
165       if~file~then~
166         file:write(err .. " \iow_char:N \\ n ")
167         file:close()

```

```

168         end~
169     print('
170         \iow_char:N \ \ \iow_char:N \ \ begingroup
171         \iow_char:N \ \ \iow_char:N \ \ ExplSyntaxOn
172         \iow_char:N \ \ \iow_char:N \ \ csname~
173         msg_error:nvv \iow_char:N \ \ \iow_char:N \ \ endcsname
174         { luabridge }
175         { failed-to-execute }
176         { g_luabridge_output_dirname_str }
177         { g_luabridge_error_output_filename_str }
178         \iow_char:N \ \ \iow_char:N \ \ endgroup
179     ')
180     end
181 }
182 \iow_now:NV
183 \g_tmpa_iow
184 \l_tmpa_tl
185 \iow_close:N
186 \g_tmpa_iow
187 \msg_info:nnV
188 { luabridge }
189 { executing-helper-script }
190 \g_luabridge_helper_script_filename_str
191 \sys_get_shell:xnNTF
192 {

```

If the environmental variable `TEXMF_OUTPUT_DIRECTORY` is undefined, use the current working directory `(.)` instead.

```

193     \str_if_eq:NNTF
194     \g_luabridge_output_dirname_str
195     \c_luabridge_default_output_dirname_str
196     {
197     \sys_if_platform_windows:TF
198     {
199         if~not~defined~TEXMF_OUTPUT_DIRECTORY~(
200             texlua~
201             \g_luabridge_helper_script_filename_str
202         )~else~(
203             texlua~
204             \g_luabridge_output_dirname_str /
205             \g_luabridge_helper_script_filename_str
206         )
207     }
208     {
209     \sys_if_platform_unix:T
210     {
211         TEXMF_OUTPUT_DIRECTORY =
212             ${TEXMF_OUTPUT_DIRECTORY:-.} % noqa: S204
213         \iow_newline:
214     }
215     texlua~
216     \g_luabridge_output_dirname_str /
217     \g_luabridge_helper_script_filename_str
218 }

```

```

219         }
220     {
221         texlua~
222         \g_luabridge_output_dirname_str /
223         \g_luabridge_helper_script_filename_str
224     }
225 }
226 { }
227 #1
228 {
229 }
230 {
231     \msg_error:nn
232     { luabridge }
233     { level-disabled }
234 }
235 }
236 \prg_generate_conditional_variant:Nnn
237 \sys_get_shell:nnN
238 { xnN }
239 { TF }
240 \cs_generate_variant:Nn
241 \msg_info:nnn
242 { nnV }
243 \cs_generate_variant:Nn
244 \msg_error:nnnn
245 { nnvv }
246 \cs_generate_variant:Nn
247 \iow_open:Nn
248 { NV }
249 \cs_generate_variant:Nn
250 \iow_now:Nn
251 { NV }
252 \msg_new:nnn
253 { luabridge }
254 { writing-helper-script }
255 {
256     Writing~a~helper~Lua~script~to~file~#1
257 }
258 \msg_new:nnn
259 { luabridge }
260 { executing-helper-script }
261 {
262     Executing~a~helper~Lua~script~from~file~#1
263 }
264 \msg_new:nnnn
265 { luabridge }
266 { failed-to-execute }
267 {
268     An~error~was~encountered~while~executing~Lua~code
269 }
270 {
271     For~further~clues,~examine~file~#1 / #2
272 }

```

```

273 \msg_new:nnnn
274 { luabridge }
275 { level-disabled }
276 {
277   Shell~escape~seems~to~be~disabled
278 }
279 {
280   You~may~need~to~run~TeX~with~the~---shell-escape~or~the~
281   --enable-write18~flag,~or~write~shell_escape=t~in~the~
282   texmf.cnf~file.
283 }
284 }
285 \int_compare:nNnT
286 { \g_luabridge_method_int }
287 =
288 { \c_luabridge_method_directlua_int }
289 {
290   \cs_new:Nn
291     \luabridge_tl_set:Nn
292     {
293       \tl_set:Nn
294         \l_tmpa_tl
295         { #2 }
296       \tl_set:Nx
297         \l_tmpa_tl
298         {
299           _ENV = setmetatable({}, {__index = _ENV}) % noqa: S204
300           local~function~print(input)
301             input = tostring(input)
302             local~output = {}
303             for~line~in~input:gmatch("[^
304               \iow_char:N \\ r
305               \iow_char:N \\ n
306             ]+") do~
307               table.insert(output, line)
308             end~
309             tex.print(output)
310           end~
311           \exp_not:V \l_tmpa_tl
312         }
313       \tl_set:No
314         #1
315         {
316           \directlua
317           {
318             \tl_use:N
319               \l_tmpa_tl
320           }
321         }
322     }
323   \cs_generate_variant:Nn
324     \lua_now:n
325     { V }
326 }

```

```

327 \cs_new:Nn
328   \luabridge_now:n
329   {
330     \luabridge_tl_set:Nn
331       \l_tmpb_tl
332       { #1 }
333     \tl_use:N
334       \l_tmpb_tl
335   }
336 \cs_new_protected:Npn
337   \luabridgeExecute
338   #1
339   {
340     \luabridge_now:e
341     { #1 }
342   }
343 \cs_generate_variant:Nn
344   \luabridge_now:n
345   { e }
346 \ExplSyntaxOff
347 </generic-package>

```

6 L^AT_EX implementation

This section contains the implementation for L^AT_EX.

```

348 <*latex-package>
349 \RequirePackage{expl3}
350 \ProvidesExplPackage
351   {lt3luabridge}%
352   {2025-06-24}%
353   {2.2.2}%
354   {An expl3 package that allows you to execute Lua code in LuaTeX or any other
355     TeX engine that exposes the shell}
356 \input lt3luabridge\relax
357 </latex-package>

```

7 ConT_EXt implementation

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

```

358 <*context-package>
359 \writestatus{loading}{ConTeXt User Module / lt3luabridge}
360 \startmodule[lt3luabridge]
361 \unprotect
362 \input lt3luabridge\relax
363 </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_EX 3*. 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>	149, 150, 166, 170, 171, 172, 173, 178, 304, 305
Numbers	
<code>\0</code>	150
C	
cs commands:	
<code>\cs_generate_variant:Nn</code> 65, 240, 243, 246, 249, 323, 343
<code>\cs_new:Nn</code>	135, 290, 327
<code>\cs_new_protected:Npn</code>	336
<code>\csname</code>	3
D	
<code>\directlua</code>	2, 316
E	
<code>\endcsname</code>	3
exp commands:	
<code>\exp_not:n</code>	158, 311
<code>\expandafter</code>	3
<code>\ExplSyntaxOff</code>	346
<code>\ExplSyntaxOn</code>	6
F	
<code>\fi</code>	5
I	
<code>\ifx</code>	3
<code>\input</code>	4, 356, 362
int commands:	
<code>\int_case:nnTF</code>	48
<code>\int_compare:nNnTF</code>	73, 285
<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>	166, 170, 171, 172, 173, 178, 304, 305
<code>\iow_close:N</code>	185
<code>\iow_newline:</code>	213
<code>\iow_now:Nn</code>	182, 250
<code>\iow_open:Nn</code>	138, 247
<code>\g_tmpa_iow</code>	139, 183, 186
J	
<code>\jobname</code>	5, 108, 111
L	
lua commands:	
<code>\lua_now:n</code>	1, 2, 324
luabridge commands:	
<code>\c_luabridge_default_error_- output_filename_str</code> ..	3, 5, 110, 133
<code>\c_luabridge_default_helper_- script_filename_str</code> ..	3, 5, 107, 126
<code>\c_luabridge_default_output_- dirname_str</code> ..	2, 81, 97, 102, 119, 195
<code>\g_luabridge_error_output_- filename_str</code>	3, 129, 132, 163
<code>\g_luabridge_helper_script_- filename_str</code>	3, 122, 125, 140, 144, 190, 201, 205, 217, 223
<code>\c_luabridge_method_directlua_- int</code>	2, 8, 22, 57, 288
<code>\g_luabridge_method_int</code> 2, 14, 17, 21, 26, 49, 71, 74, 286
<code>\c_luabridge_method_shell_int</code> 2, 11, 27, 51, 76
<code>\luabridge_now:n</code> ..	1, 2, 328, 340, 344
<code>\g_luabridge_output_dirname_str</code> 2, 5, 113, 116, 118, 162, 194, 204, 216, 222

