

The lt3luabridge package: Lua without LuaTeX

Vít Novotný*

Released 2022-06-26

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.

<code>\luabridge_now:n</code>	<code>\luabridge_now:n {⟨token list⟩}</code>
-------------------------------	--

<code>\luabridge_now:e</code>	
-------------------------------	--

New: 2022-06-26	
-----------------	--

The *⟨token list⟩* 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 *⟨Lua input⟩* 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 *⟨Lua input⟩* immediately, and in an expandable manner.

Unlike `\lua_now:n`, `\luabridge_now:n` may execute *⟨Lua input⟩* in a separate process from TeX. Therefore, you should not interact with TeX from *⟨Lua input⟩*. The only exception is the standard output produced by *⟨Lua input⟩* using for example the `print()` Lua function like in the example at the top of this page. The standard output will be inserted into TeX's input stream after *⟨Lua input⟩* has been processed at the latest.

<code>\luabridgeExecute</code>	<code>\luabridgeExecute {⟨token list⟩}</code>
--------------------------------	---

New: 2022-06-26	
-----------------	--

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`

New: 2022-06-26

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_write_eighteen_int`

New: 2022-06-26

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

`\c_luabridge_method_os_execute_int`

New: 2022-06-26

Use shell escape through the `os.execute()` Lua function 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_standard_output_filename_str`

New: 2022-06-26

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

`\c_luabridge_default_standard_output_filename_str`

New: 2022-06-26

This constant is the default value of `\g_luabridge_standard_output_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_write_eighteen_int
9   { 0 }
10 \int_const:Nn
11   \c_luabridge_method_os_execute_int
12   { 1 }
13 \int_const:Nn
14   \c_luabridge_method_directlua_int
15   { 2 }
16 \int_if_exist:NF
17   \g_luabridge_method_int
18   {
19     \int_new:N
20     \g_luabridge_method_int
```

```

21 \sys_if_engine luatex:TF
22 {
23     \int_gset_eq:NN
24     \g_luabridge_method_int
25     \c_luabridge_method_directlua_int
26 }
27 {
28     \int_gset_eq:NN
29     \g_luabridge_method_int
30     \c_luabridge_method_write_eighteen_int
31 }
32 }
33 \msg_new:nnn
34 { luabridge }
35 { unknown-method }
36 {
37     Unknown~bridging~method:~#1
38 }
39 \msg_new:nnn
40 { luabridge }
41 { method-write-eighteen }
42 {
43     Using~shell~escape~via~write18~as~the~bridging~method
44 }
45 \msg_new:nnn
46 { luabridge }
47 { method-os-execute }
48 {
49     Using~shell~escape~via~os.execute()~as~the~bridging~method
50 }
51 \msg_new:nnn
52 { luabridge }
53 { method-directlua }
54 {
55     Using~direct~Lua~access~as~the~bridging~method
56 }
57 \int_case:nnF
58 { \g_luabridge_method_int }
59 {
60     { \c_luabridge_method_write_eighteen_int }
61     {
62         \msg_info:nn
63         { luabridge }
64         { method-write-eighteen }
65     }
66     { \c_luabridge_method_os_execute_int }
67     {
68         \msg_info:nn
69         { luabridge }
70         { method-os-execute }
71     }
72     { \c_luabridge_method_directlua_int }
73     {
74         \msg_info:nn

```

```

75         { luabridge }
76         { method-directlua }
77     }
78 }
79 {
80     \cs_generate_variant:Nn
81     \msg_error:nnn
82     { nnV }
83     \msg_error:nnV
84     { luabridge }
85     { unknown-method }
86     \g_luabridge_method_int
87 }
88 \bool_if:nTF
89 {
90     \int_compare_p:nNn
91     { \g_luabridge_method_int }
92     =
93     { \c_luabridge_method_write_eighteen_int } ||
94     \int_compare_p:nNn
95     { \g_luabridge_method_int }
96     =
97     { \c_luabridge_method_os_execute_int }
98 }
99 {
100     \int_const:Nn
101     \c__luabridge_level_disabled_int
102     { 0 }
103     \int_const:Nn
104     \c__luabridge_level_enabled_int
105     { 1 }
106     \int_const:Nn
107     \c__luabridge_level_restricted_int
108     { 2 }
109     \int_new:N
110     \l__luabridge_level_int
111     \cs_if_exist:NTF
112     \pdfshellescape
113     {
114         \int_gset:Nn
115         \l__luabridge_level_int
116         { \pdfshellescape }
117     }
118     {
119         \cs_if_exist:NTF
120         \shellescape
121         {
122             \int_gset:Nn
123             \l__luabridge_level_int
124             { \shellescape }
125         }
126         {
127             \int_case:nnF
128             { \g_luabridge_method_int }

```

```

129         {
130             { \c_luabridge_method_write_eighteen_int }
131             {
132                 \int_gset_eq:NN
133                 \l__luabridge_level_int
134                 \c__luabridge_level_enabled_int
135             }
136         }
137         {
138             \int_gset:Nn
139             \l__luabridge_level_int
140             {
141                 \lua_now:n
142                 {
143                     tex.sprint(status.shell_escape or "1")
144                 }
145             }
146         }
147     }
148 }
149 \msg_new:nnn
150 { luabridge }
151 { unknown-level }
152 {
153     Unknown~shell~escape~level:~#1
154 }
155 \msg_new:nnnn
156 { luabridge }
157 { level-disabled }
158 {
159     Shell~escape~seems~to~be~disabled
160 }
161 {
162     You~may~need~to~run~TeX~with~the~---shell~escape~or~the~
163     --enable-write18~flag,~or~write~shell_escape=t~in~the~
164     texmf.cnf~file.
165 }
166 \msg_new:nnn
167 { luabridge }
168 { level-enabled }
169 {
170     Shell~escape~seems~to~be~enabled
171 }
172 \msg_new:nnnn
173 { luabridge }
174 { level-restricted }
175 {
176     Shell~escape~seems~to~be~restricted
177 }
178 {
179     You~may~need~to~run~TeX~with~the~---shell~escape~or~the~
180     --enable-write18~flag,~or~write~shell_escape=t~in~the~
181     texmf.cnf~file.
182 }

```

```

183 \str_const:Nn
184   \c_luabridge_default_output_dirname_str
185   { . }
186 \str_const:Nx
187   \c_luabridge_default_helper_script_filename_str
188   { \jobname.luabridge.lua }
189 \str_const:Nx
190   \c_luabridge_default_error_output_filename_str
191   { \jobname.luabridge.err }
192 \str_const:Nx
193   \c_luabridge_default_standard_output_filename_str
194   { \jobname.luabridge.out }
195 \int_case:nnF
196   { \l__luabridge_level_int }
197   {
198     { \c__luabridge_level_disabled_int }
199     {
200       \msg_warning:nn
201       { luabridge }
202       { level-disabled }
203     }
204     { \c__luabridge_level_enabled_int }
205     {
206       \msg_info:nn
207       { luabridge }
208       { level-enabled }
209     }
210     { \c__luabridge_level_restricted_int }
211     {
212       \msg_warning:nn
213       { luabridge }
214       { level-restricted }
215     }
216   }
217   {
218     \msg_error:nnx
219     { luabridge }
220     { unknown-level }
221     { \l__luabridge_level_int }
222   }
223 \cs_new:Nn
224   \_luabridge_assert_shell_escape:
225   {
226     \int_case:nnF
227     { \l__luabridge_level_int }
228     {
229       { \c__luabridge_level_disabled_int }
230       {
231         \msg_error:nn
232         { luabridge }
233         { level-disabled }
234       }
235     }
236   }

```

```

237 \int_case:nn
238 { \g_luabridge_method_int }
239 {
240   { \c_luabridge_method_write_eighteen_int }
241   {
242     \cs_new:Nn
243       \_luabridge_execute_shell:n
244       {
245         \_luabridge_assert_shell_escape:
246         \immediate
247           \write 18
248             { #1 }
249       }
250   }
251   { \c_luabridge_method_os_execute_int }
252   {
253     \cs_new:Nn
254       \_luabridge_execute_shell:n
255       {
256         \_luabridge_assert_shell_escape:
257         \lua_now:e
258         {
259           os.execute(
260             " \lua_escape:e { #1 } "
261           )
262         }
263       }
264   }
265 }
266 \str_if_exist:NF
267 \g_luabridge_output_dirname_str
268 {
269   \str_new:N
270   \g_luabridge_output_dirname_str
271   \tl_gset:Nn
272     \g_luabridge_output_dirname_str
273     \c_luabridge_default_output_dirname_str
274 }
275 \str_if_exist:NF
276 \g_luabridge_helper_script_filename_str
277 {
278   \str_gset_eq:NN
279     \g_luabridge_helper_script_filename_str
280     \c_luabridge_default_helper_script_filename_str
281 }
282 \str_if_exist:NF
283 \g_luabridge_error_output_filename_str
284 {
285   \str_gset_eq:NN
286     \g_luabridge_error_output_filename_str
287     \c_luabridge_default_error_output_filename_str
288 }
289 \str_if_exist:NF
290 \g_luabridge_standard_output_filename_str

```



```

291 {
292   \str_gset_eq:NN
293   \g_luabridge_standard_output_filename_str
294   \c_luabridge_default_standard_output_filename_str
295 }
296 \cs:w newwrite \cs_end:
297   \l__luabridge_output_stream
298 \cs_new:Nn
299   \luabridge_now:n
300 {
301   \immediate \openout
302     \l__luabridge_output_stream
303     \g_luabridge_helper_script_filename_str
304   \msg_info:nnV
305     { luabridge }
306     { writing-helper-script }
307     \g_luabridge_helper_script_filename_str
308   \tl_set:Nn
309     \l_tmpa_tl
310     { #1 }
311   \tl_set:Nx
312     \l_tmpb_tl
313     {
314       local~ran_ok,~error~==pcall(function()~
315         local~ran_ok,~kpse~==pcall(require,~"kpse")~
316         if~ran_ok~then~kpse.set_program_name("luatex")~end~
317         \exp_not:V \l_tmpa_tl~
318       end)~
319       if~not~ran_ok~then~
320         local~file~==io.open("
321           \g_luabridge_output_dirname_str /
322           \g_luabridge_error_output_filename_str
323           ",~"w")~
324         if~file~then~
325           file:write(error~.." \iow_char:N \\ n ")~
326           file:close()~
327         end~
328         print('
329           \iow_char:N \\ \iow_char:N \\ begingroup
330           \iow_char:N \\ \iow_char:N \\ ExplSyntaxOn
331           \iow_char:N \\ \iow_char:N \\ msg_error:nnvv
332             { luabridge }
333             { failed-to-execute }
334             { \g_luabridge_output_dirname_str }
335             { \g_luabridge_output_dirname_str }
336           \iow_char:N \\ \iow_char:N \\ endgroup
337         ')~
338       end
339     }
340   \immediate \write
341     \l__luabridge_output_stream
342     { \exp_not:V \l_tmpb_tl }
343   \immediate \closeout
344     \l__luabridge_output_stream

```

```

345     \msg_info:nnVV
346     { luabridge }
347     { executing-helper-script }
348     \g_luabridge_helper_script_filename_str
349     \g_luabridge_standard_output_filename_str
350     \tl_set:Nx
351     \l_tmpa_tl
352     {
353         texlua~"
354         \g_luabridge_output_dirname_str /
355         \g_luabridge_helper_script_filename_str
356         "~>~"
357         \g_luabridge_output_dirname_str /
358         \g_luabridge_standard_output_filename_str
359         "
360     }
361     \luabridge_execute_shell:V
362     \l_tmpa_tl
363     \file_if_exist_input:VF
364     \g_luabridge_standard_output_filename_str
365     {
366         \msg_error:nn
367         { luabridge }
368         { level-disabled }
369     }
370 }
371 \cs_generate_variant:Nn
372   \msg_info:nnn
373   { nnV }
374 \cs_generate_variant:Nn
375   \msg_info:nnnn
376   { nnVV }
377 \cs_generate_variant:Nn
378   \msg_error:nnnn
379   { nnvv }
380 \cs_generate_variant:Nn
381   \luabridge_execute_shell:n
382   { V }
383 \prg_generate_conditional_variant:Nnn
384   \file_if_exist_input:n
385   { V }
386   { F }
387 \msg_new:nnn
388   { luabridge }
389   { writing-helper-script }
390   {
391       Writing~a~helper~Lua~script~to~file~#1
392   }
393 \msg_new:nnn
394   { luabridge }
395   { executing-helper-script }
396   {
397       Executing~a~helper~Lua~script~from~file~#1~
398       and~storing~the~result~in~file~#2

```

```

399     }
400     \msg_new:nnnn
401     { luabridge }
402     { failed-to-execute }
403     {
404         An~error~was~encountered~while~executing~Lua~code
405     }
406     {
407         For further clues, examine file #1/#2
408     }
409 }
410 {
411     \cs_new:Nn
412     \luabridge_now:n
413     {
414         \tl_set:Nn
415         \l_tmpa_tl
416         { #1 }
417         \tl_set:Nx
418         \l_tmpb_tl
419         {
420             local~function~print(input)~
421             input~==~tostring(input)~
422             local~output~==~{}~
423             for~line~in~input:gmatch("[^
424                 \iow_char:N \\ r
425                 \iow_char:N \\ n
426             ]+")~do~
427                 table.insert(output,~line)~
428             end~
429             tex.print(output)~
430             end~
431             \exp_not:V \l_tmpa_tl
432         }
433         \lua_now:V
434         \l_tmpb_tl
435     }
436     \cs_generate_variant:Nn
437     \lua_now:n
438     { V }
439 }
440 \cs_new_protected:Npn
441 \luabridgeExecute
442 #1
443 {
444     \luabridge_now:e
445     { #1 }
446 }
447 \cs_generate_variant:Nn
448 \luabridge_now:n
449 { e }
450 \ExplSyntaxOff
451 </generic-package>

```

6 L^AT_EX implementation

This section contains the implementation for L^AT_EX.

```

452 <*latex-package>
453 \RequirePackage{expl3}
454 \ProvidesExplPackage
455   {lt3luabridge}%
456   {2022-06-26}%
457   {1.0.1}%
458   {An expl3 package that allows you to execute Lua code in LuaTeX or any other
459     TeX engine that exposes the shell}
460 \input lt3luabridge\relax
461 </latex-package>

```

7 ConT_EXt implementation

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

```

462 <*context-package>
463 \writestatus{loading}{ConTEXt User Module / lt3luabridge}
464 \startmodule[lt3luabridge]
465 \unprotect
466 \input lt3luabridge\relax
467 </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		cs commands:	
<code>\</code>	325, 329, 330, 331, 336, 424, 425	<code>\cs:w</code>	296
B		<code>\cs_end:</code>	296
bool commands:		<code>\cs_generate_variant:Nn</code>	80, 371, 374, 377, 380, 436, 447
<code>\bool_if:nTF</code>	88	<code>\cs_if_exist:Nn</code>	111, 119
C		<code>\cs_new:Nn</code>	223, 242, 253, 298, 411
<code>\closeout</code>	343	<code>\cs_new_protected:Npn</code>	440

D	
\directlua	2
E	
exp commands:	
\exp_not:n	317, 342, 431
\ExplSyntaxOff	450
\ExplSyntaxOn	3, 6
F	
\fi	5
file commands:	
\file_if_exist_input:n	384
\file_if_exist_input:nTF	363
I	
\ifx	3
\immediate	246, 301, 340, 343
\input	4, 460, 466
int commands:	
\int_case:nn	237
\int_case:nnTF	57, 127, 195, 226
\int_compare_p:nNn	90, 94
\int_const:Nn	7, 10, 13, 100, 103, 106
\int_gset:Nn	114, 122, 138
\int_gset_eq:NN	23, 28, 132
\int_if_exist:NTF	16
\int_new:N	19, 109
iow commands:	
\iow_char:N	325, 329, 330, 331, 336, 424, 425
J	
\jobname	188, 191, 194
L	
lua commands:	
\lua_escape:n	260
\lua_now:n	1, 141, 257, 433, 437
luabridge commands:	
_luabridge_assert_shell_escape:	224, 245, 256
_c_luabridge_default_error_- output_filename_str ..	3, 190, 287
_c_luabridge_default_helper_- script_filename_str ..	3, 187, 280
_c_luabridge_default_output_- dirname_str	2, 184, 273
_c_luabridge_default_standard_- output_filename_str ..	3, 193, 294
_g_luabridge_error_output_- filename_str	3, 283, 286, 322
_luabridge_execute_shell:n	243, 254, 361, 381
_g_luabridge_helper_script_- filename_str	2, 3, 276, 279, 303, 307, 348, 355
_c_luabridge_method_directlua_- int	2, 14, 25, 72
_g_luabridge_method_int	2, 17, 20, 24, 29, 58, 86, 91, 95, 128, 238
_c_luabridge_method_os_execute_- int	2, 11, 66, 97, 251
_c_luabridge_method_write_- eighteen_int ..	2, 8, 30, 60, 93, 130, 240
_luabridge_now:n ..	1, 299, 412, 444, 448
_g_luabridge_output_dirname_str ..	2, 267, 270, 272, 321, 354, 357
_g_luabridge_standard_output_- filename_str	3, 290, 293, 349, 358, 364
luabridge internal commands:	
_c__luabridge_level_disabled_int	101, 198, 229
_c__luabridge_level_enabled_int ..	104, 134, 204
_l__luabridge_level_int	110, 115, 123, 133, 139, 196, 221, 227
_c__luabridge_level_restricted_- int	107, 210
_l__luabridge_output_stream	297, 302, 341, 344
\luabridgeExecute	1, 441
M	
msg commands:	
_msg_error:nn	231, 366
_msg_error:nnn	81, 83, 218
_msg_error:nnnn	378
_msg_info:nn	62, 68, 74, 206
_msg_info:nnn	304, 372
_msg_info:nnnn	345, 375
_msg_new:nnn	33, 39, 45, 51, 149, 166, 387, 393
_msg_new:nnnn	155, 172, 400
_msg_warning:nn	200, 212
O	
\openout	301
P	
\pdfshellescape	112, 116
prg commands:	
_prg_generate_conditional_- variant:Nnn	383
\ProvidesExplPackage	454
R	
\relax	4, 460, 466

<code>\RequirePackage</code>	453		
		T	
		tl commands:	
		<code>\tl_gset:Nn</code>	271
		<code>\tl_set:Nn</code>	308, 311, 350, 414, 417
		<code>\l_tmpa_tl</code> .	309, 317, 351, 362, 415, 431
		<code>\l_tmpb_tl</code>	312, 342, 418, 434
	S		
<code>\shellescape</code>	120, 124		
<code>\startmodule</code>	464		
str commands:			
<code>\str_const:Nn</code>	183, 186, 189, 192		
<code>\str_gset_eq:NN</code>	278, 285, 292		
<code>\str_if_exist:NTF</code> ..	266, 275, 282, 289		
<code>\str_new:N</code>	269		
sys commands:		U	
<code>\sys_if_engine luatex:TF</code>	21	<code>\undefined</code>	3
		<code>\unprotect</code>	465
		W	
		<code>\write</code>	247, 340
		<code>\write18</code>	2
		<code>\writestatus</code>	463