

# The shellesc Package\*

L<sup>A</sup>T<sub>E</sub>X project

2023/04/15

## 1 Introduction

For many years web2c based T<sub>E</sub>X implementations have used the syntax of the `\write` command to access system commands by using a special stream 18 (streams above 15 can not be allocated to files in classical T<sub>E</sub>X so stream 18 would otherwise just print to the terminal).

This is a useful extension that did not break the strict rules on extensions in classical T<sub>E</sub>X. This package provides a simple macro level interface hiding the `write18` implementation so a command to remove a file on a unix-like system could be specified using `\ShellEscape{rm file.txt}` (or `del` in windows). Note that by default system access is not allowed and L<sup>A</sup>T<sub>E</sub>X will typically need to be called with the `--shell-escape` command line option.

The package may be used with standard `latex` or `pdflatex` or `xetex`, however it is mostly motivated by `lualatex` as from LuaT<sub>E</sub>X 0.87 onwards LuaT<sub>E</sub>X does *not* support the `\write18` syntax to access system commands: it has 256 write streams and stream 18 can be associated to a file and (without this package) has no special significance. This package defines the same `\ShellEscape` syntax in LuaL<sup>A</sup>T<sub>E</sub>X, but the implementation is via Lua and the `os.execute` function.

`\ShellEscape` in fact corresponds to `\immediate\write18` (or `\directlua`). Very rarely you may need to delay a system command until the current page is output (when page numbers are known) for this classically you could use `\write18` (or `\latelua`). This package provides `\DelayedShellEscape` as a common syntax for this use.

The shell escape status may be queried by checking the integer (chardef) command `\ShellEscapeStatus`, 0 (disabled) 1 (enabled) 2 (restricted).

To aid porting existing documents to LuaT<sub>E</sub>X 0.87 this package does overload the `\write` command so that `\write18{rm file.txt}` will work with LuaT<sub>E</sub>X. Note that the redefinition of `\write` can not detect whether `\immediate` has been used, `\immediate` will work as normal when writing to file streams or the terminal but the special case of stream 18 which is defined to use `os.execute` always uses `\directlua` (so corresponds to `\immediate\write18`). In the rare situations that you need non-immediate `\write18` in a document being ported to current LuaT<sub>E</sub>X, you will need to change to use the `\DelayedShellEscape` command.

## 2 Implementation

1 `\package`

---

\*This file has version number v1.0d, last revised 2023/04/15.

```

2 \chardef\shellesc@quotecat\catcode'\ "
3 \chardef\shellesc@underscorecat\catcode'\_
4 \@makeother\"
5 \@makeother\_

```

## 2.1 Status Check

## 2.2 The shellesc package interface

`\ShellEscapeStatus` Integer value with meanings 0 (shell escape disabled), 1 (shell escape allowed), 2 (Restricted shell escape).

```

6 \chardef\ShellEscapeStatus
7 \ifx\pdfshellescape\@undefined
8 \ifx\shellescape\@undefined
9 \ifx\directlua\@undefined
10 \z@
11 \else
12 \directlua{%
13 tex.sprint((status.shell_escape or os.execute()) .. " ")}
14 \fi
15 \else
16 \shellescape
17 \fi
18 \else
19 \pdfshellescape
20 \fi

21 \ifcase\ShellEscapeStatus
22 \PackageWarning{shellesc}{Shell escape disabled}
23 \or
24 \PackageInfo {shellesc}{Unrestricted shell escape enabled}
25 \else
26 \PackageInfo {shellesc}{Restricted shell escape enabled}
27 \fi

```

`\ShellEscape` Execute the supplied tokens as a system dependent command, assuming such execution is allowed.

```

28 \ifx\lastsavedimageresourcepages\@undefined
29 \protected\def\ShellEscape{\immediate\write18 }
30 \else
31 \protected\def\ShellEscape{\directlua\ShellEscape@Lua}
32 \fi

```

`\DelayedShellEscape` Execute the supplied tokens as a system dependent command, when this node is shipped out with the completed page, assuming such execution is allowed.

```

33 \ifx\lastsavedimageresourcepages\@undefined
34 \protected\def\DelayedShellEscape{\relax\write18 }
35 \else
36 \protected\def\DelayedShellEscape{\latelua\ShellEscape@Lua}
37 \fi

```

`\ShellEscape@Lua` Shared Lua code for `\DelayedShellEscape` and `\ShellEscape`.

```

38 \ifx\directlua\@undefined\else

```

```

39 \protected\def\ShellEscape@Lua#1{%
40 local status, msg = os.execute("\luaescapestring{#1}")%
41 if status == nil then
42   texio.write_nl("log",%
43     "runsystem(" .. "\luaescapestring{#1}"%
44     .. ")...(" .. msg .. ").\string\n")
45 elseif status == 0 then
46   texio.write_nl("log",%
47     "runsystem(" .. "\luaescapestring{#1}"%
48     .. ")...executed.\string\n")
49 else
50   texio.write_nl("log",%
51     "runsystem(" .. "\luaescapestring{#1}"%
52     .. ")...failed. " .. (msg or "") .. "\string\n")
53 end
54 }}
55 \fi

```

### 2.3 The write18 package interface

In web2c based engines other than Lua $\TeX$ , `\write18` may be used directly. The same was true in older Lua $\TeX$ , but from version 0.85 onwards that is not available.

The above `shellesc` package interface is recommended for new code, however for ease of porting existing documents and packages to newer Lua $\TeX$  releases, a `\write18` interface is provided here via a call to Lua's `os.execute`.

Note that as currently written this always does an *immediate* call to the system.

`\immediate` is supported but ignored, `\immediate\write18` and `\write18` both execute immediately. To use a delayed execution at the next shipout, use the `\DelayedShellEscape` command defined above.

Note that it would be easy to make `\wriete18` defined here use delayed execution, just use `\DelayedShellEscape` instead of `ShellEscape` in the definition below. However detecting `\immediate` is tricky so the choice here is to always use the immediate form, which is overwhelmingly more commonly used with `\write18`.

Stop at this point if not a recent Lua $\TeX$ .

```

56 \ifx\lastsavedimageresourcepages\@undefined
57 \catcode'\shellesc@quotecat
58 \catcode'\shellesc@underscorecat
59 \expandafter\endinput
60 \fi

61 \directlua{%

62 shellesc = shellesc or {}

```

Lua function to use the token scanner to grab the following  $\TeX$  number, and then test if stream 18 is being used, and then insert an appropriate  $\TeX$  command to handle the following brace group in each case.

```

63 local function write_or_execute()
64   local s = token.scan_int()
65   if (s==18) then
66     tex.sprint(\the\numexpr\catcodetable@atletter\relax,
67       "\string\ShellEscape ")
68   else
69     tex.sprint(\the\numexpr\catcodetable@atletter\relax,

```

```
70             "\string\shellesc@write " .. s)
71   end
72 end
73 shellesc.write_or_execute=write_or_execute
74 }
75 \let\shellesc@write\write
76 \protected\def\write{\directlua{shellesc.write_or_execute()}}
77 \catcode'\shellesc@quotecat
78 \catcode'\shellesc@underscorecat
79 \end{package}
```