

iexec: L^AT_EX Package for Inputable Shell Executions*

Yegor Bugayenko
yegor256@gmail.com

2026-07-13, 0.16.1

NB! This package does not operate on Windows!

1 Introduction

This package permits the execution of shell commands directly from within the document and the inclusion of their output in the resulting text:

Today is **13-Jul-2026!**

```
1 \documentclass{article}
2 \usepackage{iexec}
3 \usepackage[paperwidth=3in]{geometry}
4 \pagestyle{empty}
5 \begin{document}
6 Today is \textbf{%
7   \iexec{date +%e-%b-%Y}}\unskip!
8 \end{document}
```

\iexec The sole command supplied by this package is `\iexec` [*options*] {*cmd*}. Its only mandatory argument, *cmd*, denotes the command to be executed through the terminal shell (`bash`, or whichever shell is configured as the default in the host console).

Compilation must be performed with `pdflatex` (or simply `latex`) under the `--shell-escape` flag, so that [shellesc](#) may execute the supplied shell command.

It is worth noting that L^AT_EX invariably employs the “`/bin/sh`” shell. This behaviour cannot be altered, as [explained here](#).

2 Options

quiet Should the output be required to remain invisible, `\phantom{\iexec{...}}` may be employed. Alternatively, the “`quiet`” option is available:

```
I just want to delete some file:
\iexec[quiet]{rm -f foo.txt}
```

Under this option, whatever the shell command produces is excluded from the document.

stdout The output of the executed code is written to the file specified as an optional

*The sources are in GitHub at [yegor256/iexec](#)

argument of `\iexec` (the default being “`iexec.tmp`”):

```
Today is \iexec[stdout=date.txt]{date +%e-%b-%Y | tr -d '\n'}.
```

The trailing portion of the command above removes all line terminators.

stderr The error output of the executed code is written to the file specified as an optional argument of `\iexec` (by default the error output is redirected into “`stdout`”):

```
Today is \iexec[stderr=my.txt]{broken-command}.
```

exit The exit code of the command is written to a file. The name of this file may be altered by means of the “`exit`” option:

```
Today is \iexec[exit=code.txt]{./broken-command.sh}.
```

trace The file thus produced is deleted immediately after its use. To prevent this deletion, the “`trace`” package option may be employed: all files will then remain in the directory where they were created. Tracing may also be enabled globally, for the entire document, by means of the “`trace`” option of the package:

```
\documentclass{article}
\usepackage[trace]{iexec}
\begin{document}
This file is preserved: \iexec[stdout=me.txt]{whoami}.
\end{document}
```

append The “`stdout`” thus produced is appended to the file specified:

```
\documentclass{article}
\usepackage[trace]{iexec}
\begin{document}
\iexec[append,stdout=foo.txt,quiet]{echo 'Hello, '}
\iexec[append,stdout=foo.txt,quiet]{echo 'Jeffrey!'}
\input{foo.txt}
\end{document}
```

unskip To remove the trailing space following the content, the `unskip` package option may be employed, which appends an `\unskip` command to every `\iexec`:

```
\documentclass{article}
\usepackage[unskip]{iexec}
\begin{document}
Today is \iexec{date +%Y}!
\end{document}
```

log The “`stdout`” thus produced is printed to the TeX log:

```
\iexec[log]{echo 'Hello, \LaTeX!'}
```

null The “`stdout`” of the command is redirected to “`/dev/null`” (or to “`NUL`” on Windows):

```
\iexec[null]{rm some-file.txt}
```

ignore By default, an error is reported whenever the exit code differs from zero. This behaviour may be suppressed by means of the “`ignore`” option:

```
\iexec[ignore]{broken-command}
```

maybe In the absence of the `--shell-escape` flag, the `\iexec` command would normally produce a compilation failure. Such failure may be averted by means of the `maybe` option, whereby the execution of `\iexec` is silently skipped whenever `--shell-escape` is not in force:

```
\iexec[maybe]{echo 'Hello, world!'}
```

3 Implementation

First, we include the [shellesc](#) package, which we use in order to execute shell commands:

```
1 \RequirePackage{shellesc}
```

Then, we include the [pdftexcmds](#) package, which provides `\pdf@filesize`, working uniformly across `pdftex`, `luatex`, and `xetex` (the `\pdf@filesize` primitive exists in `pdftex` only):

```
2 \RequirePackage{pdftexcmds}
```

Then, we parse package options, with the help of [pgfopts](#):

```
3 \RequirePackage{pgfopts}
4 \pgfkeys{
5   /iexec/.cd,
6   trace/.store in=\iexec@trace,
7 }
8 \ProcessPgfPackageOptions{/iexec}
```

Then, we prepare to parse the options of the `\iexec` command, with the help of [pgfkeys](#):

```
9 \RequirePackage{pgfkeys}
10 \makeatletter\pgfkeys{
11   /iexec/.is family,
12   /iexec,
13   exit/.store in = \iexec@exit,
14   exit/.default = iexec.ret,
15   stdout/.store in = \iexec@stdout,
16   stdout/.default = iexec.tmp,
17   stderr/.store in = \iexec@stderr,
18   trace/.store in = \iexec@traceit,
19   append/.store in = \iexec@append,
20   log/.store in = \iexec@log,
21   null/.store in = \iexec@null,
22   unskip/.store in = \iexec@unskip,
23   quiet/.store in = \iexec@quiet,
24   ignore/.store in = \iexec@ignore,
25   maybe/.store in = \iexec@maybe,
26   stdout,exit
27 }\makeatother
```

`\iexec@typeout` Then, we define an internal command `\iexec@typeout` for printing the content of a file, as suggested [here](#):

```
28 \RequirePackage{expl3}
29 \makeatletter\ExplSyntaxOn
30 \NewDocumentCommand{\iexec@typeout}{m}{
```

```

31 \iexec_typeout_file:n { #1 }}
32 \ior_new:N \g_iexec_typeout_ior
33 \cs_new_protected:Nn \iexec_typeout_file:n
34 {
35 \ior_open:Nn \g_iexec_typeout_ior { #1 }
36 \ior_str_map_inline:Nn \g_iexec_typeout_ior
37 {\iow_term:n { ##1 }}
38 \ior_close:N \g_iexec_typeout_ior
39 }
40 \ExplSyntaxOff\makeatother

```

Then, we detect the operating system, because a few shell fragments differ between Windows and Unix: the null device (`/dev/null` versus `NUL`), the file removal command (`rm` versus `del`), and the way the exit code of the command is captured.

```

41 \makeatletter
42 \newif\ifiexec@windows
43 \ExplSyntaxOn
44 \sys_if_platform_windows:TF{\iexec@windowstrue}{\iexec@windowsfalse}
45 \ExplSyntaxOff
46 \ifiexec@windows
47 \def\iexec@devnull{NUL}
48 \def\iexec@rm{del}
49 \else
50 \def\iexec@devnull{/dev/null}
51 \def\iexec@rm{rm}
52 \fi
53 \makeatother

```

`\iexec` Then, we define `\iexec` command. It is implemented with the help of `\ShellEscape` from `shellesc` package:

```

54 \makeatletter
55 \newread\iexec@exitfile
56 \newcommand\iexec[2][]{%
57 \begingroup%
58 \pgfqkeys{/iexec}{#1}%

```

First, we verify that `latex` is running with the `--shell-escape` option, since without it nothing will work; so, it's better to throw an error earlier than later:

```

59 \ifnum\ShellEscapeStatus=1%
60 \begingroup%

```

Then, we start the log from a clean line:

```

61 \ifdefined\iexec@log%
62 \message{^^J}%
63 \fi%

```

Then, we define a few special chars in order to escape them in the shell (the full list of them is in [macros2e](#)):

```

64 \let%\@percentchar%
65 \let\\ \@backslashchar%
66 \let\{\@charlb%
67 \let\}\@charrb%

```

Then, we execute it and save exit code into a file (where we also add % in order to trim the content to exactly one number, as suggested [here](#)): On Windows, `cmd.exe` parentheses are not a subshell, so a `cd` inside the user's command would leak out and make the trailing exit-code echo resolve its file against the changed directory; therefore, on Windows we run the command through a nested `cmd /c "..."`, keeping its directory change isolated (on Unix the (...) subshell already does this):

```

68      \def\iexec@cmd{\ifiexec@windows cmd\space/c\space"(\fi\space
69      \ifdefined\iexec@append>\fi>
70      \ifdefined\iexec@null\iexec@devnull\else\iexec@stdout\fi
71      \space\ifdefined\iexec@stderr2>\iexec@stderr\else2>&1\fi%
72      \ifiexec@windows%
73      \space&&\space(echo 0)>\iexec@exit\space||\space(echo 1)>\iexec@exit%
74      \else%
75      ;\space/bin/echo -n \string$?\% >\iexec@exit%
76      \fi}%
77      \ShellEscape{\iexec@cmd}%

```

Then, a message is printed to T_EX log:

```

78      \ifdefined\iexec@log%
79      \message{iexec: [\iexec@cmd]^^J}%
80      \fi%
81      \endgroup%

```

Then, we read back the exit code, from the file:

```

82      \immediate\openin\iexec@exitfile=\iexec@exit%
83      \read\iexec@exitfile to \iexec@code%
84      \immediate\closein\iexec@exitfile%

```

Then, if required, we print the content of the stdout file to T_EX log:

```

85      \ifdefined\iexec@null\else%
86      \IfFileExists
87      {\iexec@stdout}
88      {}
89      {\PackageError{iexec}{The "\iexec@stdout" file is absent
90      after processing, looks like some internal error}{}}%
91      \ifdefined\iexec@log%
92      \message{iexec: This is the content of '\iexec@stdout'%
93      \space(\pdf@filesize{\iexec@stdout} bytes):^^J}%
94      \IfFileExists
95      {\iexec@stdout}
96      {\iexec@typeout{\iexec@stdout}}
97      {\PackageError{iexec}{The "\iexec@stdout" file is absent
98      after processing, looks like some internal error}{}}%
99      \message{<EOF>^^J}%
100     \else%
101     \ifnum\iexec@code=0\else%
102     \ifdefined\iexec@ignore\else%
103     \message{iexec: See the content of '\iexec@stdout'%
104     \space(\pdf@filesize{\iexec@stdout} bytes)
105     after failure:^^J}%
106     \iexec@typeout{\iexec@stdout}%
107     \message{<EOF>^^J}%
108     \fi%
109     \fi%

```

```

110     \fi%
111 \fi%

```

Then, we check whether it's zero or not (if not zero, we either print a message or fail the build, depending on the presence of ignore option):

```

112     \ifnum\iexec@code=0\else%
113         \ifdefined\iexec@ignore%
114             \ifdefined\iexec@log%
115                 \message{iexec: Execution failure ignored,
116                     the exit code was \iexec@code^^J}%
117             \fi%
118         \else%
119             \PackageError{iexec}{Execution failure,
120                 the exit code was \iexec@code}{}%
121         \fi%
122 \fi%

```

Then, we include the produced output into the current document:

```

123 \ifdefined\iexec@null\else%
124 \ifdefined\iexec@quiet%
125     \ifdefined\iexec@log%
126         \message{iexec: Due to 'quiet' option we didn't read
127             the content of '\iexec@stdout'
128             (\pdf@filesize{\iexec@stdout} bytes)^^J}%
129     \fi%
130 \else%
131     \ifdefined\iexec@log%
132         \message{iexec: We are going to include the content of
133             '\iexec@stdout'(\pdf@filesize{\iexec@stdout} bytes)...^^J}%
134     \fi%
135     \input{\iexec@stdout}%
136     \ifdefined\iexec@unskip\unskip\fi%
137     \message{iexec: The content of '\iexec@stdout'
138         was included into the document^^J}%
139 \fi\fi%

```

Then, we delete the file or leave it untouched:

```

140 \ifdefined\iexec@null\else%
141 \ifdefined\iexec@trace%
142     \ifdefined\iexec@log%
143         \message{iexec: Due to package option 'trace',
144             the files '\iexec@stdout' and '\iexec@exit'
145             were not deleted^^J}%
146     \fi%
147 \else%
148     \ifdefined\iexec@traceit%
149         \ifdefined\iexec@log%
150             \message{iexec: Due to 'trace' package option,
151                 the files '\iexec@stdout' and '\iexec@exit'
152                 were not deleted^^J}%
153         \fi%
154     \else%
155         \ShellEscape{\iexec@rm\space\iexec@stdout}%
156         \ifdefined\iexec@log%
157             \message{iexec: The file '\iexec@stdout' was deleted^^J}%

```

```

158         \fi%
159         \ShellEscape{\iexec@rm\space\iexec@exit}%
160         \ifdefined\iexec@log%
161             \message{iexec: The file '\iexec@exit' was deleted^^J}%
162         \fi%
163     \fi%
164 \fi\fi%

```

Finally, we ignore the whole story if the maybe option is provided and the `--shell-escape` is not set:

```

165     \else%
166         \ifdefined\iexec@maybe%
167             \message{iexec: The execution skipped because --shell-escape
168                 is not set and 'maybe' option is provided^^J}%
169         \else%
170             \PackageError{iexec}{You must run TeX processor with
171                 --shell-escape option}{}%
172         \fi%
173     \fi%
174 \endgroup%
175 }\makeatother
176 \endinput

```

Change History

0.10.0	<code>\iexec</code> : The ability to track exit code was added. Now, the code is saved into “ <code>iexec.ret</code> ” file, which is then read and checked for zero value. 5	<code>\iexec</code> command, when <code>--shell-escape</code> option is off. . . 7
	The file “ <code>iexec.ret</code> ” is reused for all scripts. 4	
	The option “ <code>ignore</code> ” suppresses the checking of “ <code>iexec.ret</code> ” value. 4	
0.11.0	<code>\iexec</code> : The file with exit code now contains just numbers, without end of line. 5	0.16.0 General: The package now works on Windows too, not only on Unix, because the shell fragments are chosen depending on the operating system. 4
	The option “ <code>exit</code> ” allows to change the name of the file with exit code. 4	0.16.1 General: The file size is now printed to the log under <code>luatex</code> and <code>xetex</code> too, not only <code>pdfTeX</code> , because we use <code>\pdf@filesize</code> from <code>pdfTeXcmds</code> instead of the <code>\pdffilesize</code> primitive. 3
0.11.1	<code>\iexec</code> : When exit code is printed to the file, we add percentchar at the end of line in order to avoid extra space when reading it back. 5	0.16.2 <code>\iexec</code> : On Windows, the user command is wrapped in <code>cmd /c "..."</code> , so that a <code>cd</code> inside it stays in a child process and doesn't leak into the exit-code capture that follows. 5
0.11.2	<code>\iexec</code> : If execution fails, we print the content of “ <code>stdout</code> ” anyway, even if the “ <code>log</code> ” is not turned on. 5	0.7.0 <code>\iexec</code> : The option “ <code>append</code> ” was introduced — if it's turned on, <code>stdout</code> will be appended to the file, instead of rewriting it (this is how it was before). 4
0.11.3	<code>\iexec</code> : Bug fixed, because of which we had an extra leading space. 5	The option “ <code>log</code> ” was introduced, to turn on log/debug messages in TeX log (they were all visible always, which was sometimes annoying). Also, this option enables printing of the entire content of <code>stdout</code> to the log too (this may be pretty convenient for debugging). 4
0.11.4	<code>\iexec</code> : In this version we escape dollar sign with <code>\string</code> command. 5	0.8.0 <code>\iexec</code> : The option “ <code>null</code> ” was introduced, allowing redirection of <code>stdout</code> to “ <code>/dev/null</code> ”. Doesn't work on Windows, though. 5
0.12.0	<code>\iexec</code> : The option “ <code>unskip</code> ” adds <code>\unskip</code> after each <code>\iexec</code> , in order to strip the trailing end-of-line space. 4	0.9.0 <code>\iexec</code> : The option “ <code>stderr</code> ” was introduced, allowing redirection of <code>stderr</code> to a file. Without this option specified, <code>stderr</code> will go to <code>stdout</code> 5
0.14.0	General: The <code>xkeyval</code> package is not used anymore. Instead, we use <code>pgfopts</code> in order to parse package options. 3	
	<code>\iexec</code> : The <code>maybe</code> option introduced, allowing the user to skip the entire execution of the	

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		
\% 64, 75	\iexec@maybe ... 25, 166	\message 62,
\@backslashchar ... 65	\iexec@null 21, 70, 85, 123, 140	79, 92, 99, 103,
\@charlb 66	\iexec@quiet ... 23, 124	107, 115, 126,
\@charrb 67	\iexec@rm 48, 51, 155, 159	132, 137, 143,
\@percentchar 64	\iexec@stderr ... 17, 71	150, 157, 161, 167
\\ 65	\iexec@stdout 15, 70,	
\{ 66	87, 89, 92, 93, 95,	N
\} 67	96, 97, 103, 104,	\NewDocumentCommand 30
	106, 127, 128,	\newif 42
	133, 135, 137,	\newread 55
C	144, 151, 155, 157	
\closein 84	\iexec@trace ... 6, 141	O
\cs 33	\iexec@traceit . 18, 148	\openin 82
	\iexec@typeout 28, 96, 106	
D	\iexec@unskip .. 22, 136	P
\def ... 47, 48, 50, 51, 68	\iexec@windowsfalse 44	\PackageError 89, 97, 119, 170
	\iexec@windowstrue . 44	\pdf@filesize 93, 104, 128, 133
E	\ifdefined 61,	\pgfkeys 4, 10
\endinput 176	69, 70, 71, 78, 85,	\pgfqkeys 58
\ExplSyntaxOff .. 40, 45	91, 102, 113, 114,	\ProcessPgfPackageOptions 8
\ExplSyntaxOn ... 29, 43	123, 124, 125,	
	131, 136, 140,	
G	141, 142, 148,	
\g 32, 35, 36, 38	149, 156, 160, 166	R
	\IfFileExists ... 86, 94	\read 83
I	\ifiexec@windows .. 42, 46, 68, 72	\RequirePackage ... 1, 2, 3, 9, 28
\iexec 31, 33, 54	\ifnum 59, 101, 112	
\iexec@append ... 19, 69	\immediate 82, 84	S
\iexec@cmd ... 68, 77, 79	\input 135	\ShellEscape 77, 155, 159
\iexec@code 83,	\ior 32, 35, 36, 38	\ShellEscapeStatus . 59
101, 112, 116, 120	\iow 37	\space ... 68, 71, 73,
\iexec@devnull 47, 50, 70		75, 93, 104, 155, 159
\iexec@exit 13, 73, 75, 82,	M	\string 75
144, 151, 159, 161	\makeatletter 10, 29, 41, 54	\sys 44
\iexec@exitfile ... 55, 82, 83, 84	\makeatother 27, 40, 53, 175	
\iexec@ignore 24, 102, 113		U
\iexec@log 20, 61, 78,		\unskip 136
91, 114, 125, 131,		
142, 149, 156, 160		