

iexec: L^AT_EX Package for Inputable Shell Executions*

Yegor Bugayenko
yegor256@gmail.com

2023-10-12, 0.12.0

NB! This package doesn't work on Windows!

1 Introduction

This package helps you execute shell commands right from the document and then put their output to the document:

Today is **12-Oct-2023!**

```
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 only command provided by this package is `\iexec [options] {cmd}`. Its only mandatory argument *cmd* is the command to be executed through the terminal shell (bash, or whatever is set as the default one in your console).

You have to run `pdflatex` (or just `latex`) with the `--shell-escape` flag in order to let [shellesc](#) execute your shell command.

2 Options

quiet If you don't want the output to be visible, use `\phantom{\iexec{...}}`. Otherwise, you can use the “quiet” option:

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

In this case, whatever the shell command produces will not be included into the document.

stdout The output of your code is saved into the file provided as an optional argument of `\iexec` (the default value is “iexec.tmp”):

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

*The sources are in GitHub at [yegor256/iexec](https://github.com/yegor256/iexec)

The trailing part of the command here removes all ends-of-line.

stderr The error output of the code is saved into the file provided as an optional argument of `\iexec` (by default the error output is streamed into “stdout”):

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

exit The exit code of the command is saved into a file. You can change the name of it using the “exit” option:

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

trace The file specified will be deleted right after its usage. If you don’t want this to happen, use the “trace” package option: all files will remain in the directory where they were created. It’s possible to turn on the tracing globally, for the entire document, using the “trace” option of the package:

```
\documentclass{article}
\usepackage[trace]{iexec}
\begin{document}
This file won't be deleted: \iexec[stdout=me.txt]{whoami}.
\end{document}
```

append The “stdout” produced will be 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 In order to remove the trailing spacing after the content, you may use `unskip` package option, which will append `\unskip` command to every `\iexec`:

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

log The “stdout” produced will be printed in the \TeX log:

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

null The “stdout” of the command will be sent to “/dev/null”:

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

ignore By default, we report an error if the exit code is not equal to zero. You can suppress this with the “ignore” option:

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

3 Implementation

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

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

Then, we parse package options:

```
2 \RequirePackage{xkeyval}
3 \makeatletter
4 \newif\ifiexec@trace
5 \DeclareOptionX{trace}{\ifiexec@tracetrue}
6 \ProcessOptionsX\relax
7 \makeatother
```

Then, we prepare to parse the options of `\iexec` command:

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

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

```
26 \RequirePackage{expl3}
27 \makeatletter\ExplSyntaxOn
28 \NewDocumentCommand{\iexec@typeout}{m}{
29 \iexec_typeout_file:n { #1 }}
30 \ior_new:N \g_iexec_typeout_ior
31 \cs_new_protected:Nn \iexec_typeout_file:n
32 {
33 \ior_open:Nn \g_iexec_typeout_ior { #1 }
34 \ior_str_map_inline:Nn \g_iexec_typeout_ior
35 {\iow_term:n { ##1 }}
36 \ior_close:N \g_iexec_typeout_ior
37 }
38 \ExplSyntaxOff\makeatother
```

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

```
39 \makeatletter
40 \newread\iexec@exitfile
41 \newcommand\iexec[2][]{%
```

```

42 \begingroup%
43 \pgfqkeys{/iexec}{#1}%

```

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

```

44 \ifnum\ShellEscapeStatus=1\else%
45 \PackageError{iexec}{You must run TeX processor with
46 --shell-escape option}{}%
47 \fi%
48 \begingroup%

```

Then, we start the log from a clean line:

```

49 \ifdefined\iexec@log%
50 \message{^^J}%
51 \fi%

```

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

```

52 \let%\@percentchar%
53 \let\\ \@backslashchar%
54 \let\{\@charlb%
55 \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](#)):

```

56 \def\iexec@cmd{(#2)
57 \ifdefined\iexec@append>\fi>
58 \ifdefined\iexec@null/dev/null\else\iexec@stdout\fi
59 \space\ifdefined\iexec@stderr2>\iexec@stderr\else2>&1\fi;
60 /bin/echo -n \string$?\% >\iexec@exit}%
61 \ShellEscape{\iexec@cmd}%

```

Then, a message is printed to T_EX log:

```

62 \ifdefined\iexec@log%
63 \message{iexec: [\iexec@cmd]^^J}%
64 \fi%
65 \endgroup%

```

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

```

66 \immediate\openin\iexec@exitfile=\iexec@exit%
67 \read\iexec@exitfile to \iexec@code%
68 \immediate\closein\iexec@exitfile%

```

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

```

69 \ifdefined\iexec@null\else%
70 \ifdefined\iexec@log%
71 \message{iexec: This is the content of '\iexec@stdout':^^J}%
72 \iexec@typeout{\iexec@stdout}%
73 \message{<EOF>^^J}%
74 \else%
75 \ifnum\iexec@code=0\else%
76 \ifdefined\iexec@ignore\else%
77 \message{iexec: See the content of '\iexec@stdout'
78 after failure:^^J}%
79 \iexec@typeout{\iexec@stdout}%
80 \message{<EOF>^^J}%
81 \fi%

```

```

82     \fi%
83 \fi\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):

```

84     \ifnum\iexec@code=0\else%
85         \ifdefined\iexec@ignore%
86             \ifdefined\iexec@log%
87                 \message{iexec: Execution failure ignored,
88                     the exit code was \iexec@code^^J}%
89             \fi%
90         \else%
91             \PackageError{iexec}{Execution failure,
92                 the exit code was \iexec@code}{}%
93         \fi%
94     \fi%

```

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

```

95     \ifdefined\iexec@null\else%
96     \ifdefined\iexec@quiet%
97         \ifdefined\iexec@log%
98             \message{iexec: Due to 'quiet' option we didn't read
99                 the content of '\iexec@stdout'
100             \ifdefined\pdffilesize (\pdffilesize{\iexec@stdout}
101                 bytes)\fi^^J}%
102         \fi%
103     \else%
104         \ifdefined\iexec@log%
105             \message{iexec: We are going to include the content of
106                 '\iexec@stdout'\ifdefined\pdffilesize (\pdffilesize
107                 {\iexec@stdout} bytes)\fi...^^J}%
108         \fi%
109         \input{\iexec@stdout}%
110         \ifdefined\iexec@unskip\unskip\fi%
111         \message{iexec: The content of '\iexec@stdout'
112             was included into the document^^J}%
113     \fi\fi%

```

Finally, we delete the file or leave it untouched:

```

114     \ifdefined\iexec@null\else%
115     \ifiexec@trace%
116         \ifdefined\iexec@log%
117             \message{iexec: Due to package option 'trace',
118                 the files '\iexec@stdout' and '\iexec@exit' were
119                 not deleted^^J}%
120         \fi%
121     \else%
122         \ifdefined\iexec@traceit%
123             \ifdefined\iexec@log%
124                 \message{iexec: Due to 'trace' package option,
125                     the files '\iexec@stdout' and '\iexec@exit'
126                     were not deleted^^J}%
127             \fi%
128         \else%
129             \ShellEscape{rm \iexec@stdout}%

```

```

130         \ifdefined\iexec@log%
131             \message{iexec: The file '\iexec@stdout' was deleted^^J}%
132         \fi%
133         \ShellEscape{rm \iexec@exit}%
134         \ifdefined\iexec@log%
135             \message{iexec: The file '\iexec@exit' was deleted^^J}%
136         \fi%
137     \fi%
138 \fi\fi%
139 \endgroup%
140 }\makeatother

```

Change History

0.10.0	General: The option “ignore” suppresses the checking of “iexec.ret” value.	3	dollar sign with \string command.	4	
	\iexec: The ability to track exit code was added. Now, the code is saved into “iexec.ret” file, which is then read and checked for zero value.	4			
	The file “iexec.ret” is reused for all scripts.	3			
0.11.0	General: The option “exit” allows to change the name of the file with exit code.	3			
	\iexec: The file with exit code now contains just numbers, without end of line.	4			
0.11.1	\iexec: 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.	4			
0.11.2	\iexec: If execution fails, we print the content of “stdout” anyway, even if the “log” is not turned on.	4			
0.11.3	\iexec: Bug fixed, because of which we had an extra leading space. . . .	4			
0.11.4	\iexec: In this version we escape				
			0.12.0	General: The option “unskip” adds \unskip after each \iexec, in order to trip the trailing end of line space.	3
			0.7.0	General: The option “append” was introduced — if it’s turned on, stdout will be appended to the file, instead of rewriting it (this is how it was before).	3
				The option “log” 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 stdout to the log too (this may be pretty convenient for debugging).	3
			0.8.0	\iexec: The option “null” was introduced, allowing redirection of stdout to “/dev/null”. Doesn’t work on Windows, though.	4
			0.9.0	\iexec: The option “stderr” was introduced, allowing redirection of stderr to a file. Without this option specified, stderr will go to stdout. . .	4

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	<code>\iexec@exitfile</code> ...	<code>\makeatother</code> 7, 25, 38, 140
<code>\%</code> 52, 60 40, 66, 67, 68	<code>\message</code> 50, 63, 71, 73,
<code>\@backslashchar</code> 53	<code>\iexec@ignore</code> .. 23, 76, 85	77, 80, 87, 98, 105,
<code>\@charlb</code> 54	<code>\iexec@log</code> 19,	111, 117, 124, 131, 135
<code>\@charrb</code> 55	49, 62, 70, 86, 97,	N
<code>\@percentchar</code> 52	104, 116, 123, 130, 134	<code>\NewDocumentCommand</code> . 28
<code>\\</code> 53	<code>\iexec@null</code>	<code>\newif</code> 4
<code>\{</code> 54	... 20, 58, 69, 95, 114	<code>\newread</code> 40
<code>\}</code> 55	<code>\iexec@quiet</code> 22, 96	
	<code>\iexec@stderr</code> 16, 59	O
C	<code>\iexec@stdout</code> . 14, 58,	<code>\openin</code> 66
<code>\closein</code> 68	71, 72, 77, 79, 99,	
<code>\cs</code> 31	100, 106, 107, 109,	P
	111, 118, 125, 129, 131	<code>\PackageError</code> 45, 91
D	<code>\iexec@traceit</code> .. 17, 122	<code>\pdffilesize</code> ... 100, 106
<code>\DeclareOptionX</code> 5	<code>\iexec@tracetrue</code> 5	<code>\pgfkeys</code> 9
<code>\def</code> 56	<code>\iexec@typeout</code> . 26, 72, 79	<code>\pgfqkeys</code> 43
E	<code>\iexec@unskip</code> ... 21, 110	<code>\ProcessOptionsX</code> 6
<code>\ExplSyntaxOff</code> 38	<code>\ifdefined</code> 49, 57, 58, 59,	R
<code>\ExplSyntaxOn</code> 27	62, 69, 70, 76, 85,	<code>\read</code> 67
	86, 95, 96, 97, 100,	<code>\relax</code> 6
G	104, 106, 110, 114,	<code>\RequirePackage</code> 1, 2, 8, 26
<code>\g</code> 30, 33, 34, 36	116, 122, 123, 130, 134	
	<code>\ifiexec@trace</code> ... 4, 115	S
I	<code>\ifnum</code> 44, 75, 84	<code>\ShellEscape</code> . 61, 129, 133
<code>\iexec</code> 29, 31, 39	<code>\immediate</code> 66, 68	<code>\ShellEscapeStatus</code> .. 44
<code>\iexec@append</code> 18, 57	<code>\input</code> 109	<code>\space</code> 59
<code>\iexec@cmd</code> 56, 61, 63	<code>\ior</code> 30, 33, 34, 36	<code>\string</code> 60
<code>\iexec@code</code> 67, 75, 84, 88, 92	<code>\iow</code> 35	
<code>\iexec@exit</code> ... 12, 60,	M	U
66, 118, 125, 133, 135	<code>\makeatletter</code> . 3, 9, 27, 39	<code>\unskip</code> 110