iexec: LaTeX Package for Inputable Shell Executions*

Yegor Bugayenko
yegor256@gmail.com

2024-01-14, 0.14.0

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:

<table>
<tr>
<td>Today is 14-Jan-2024!</td>
<td>

```
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}
```

</td>
</tr>
</table>

\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.

It is important to remember that LaTeX always uses "/bin/sh" shell. This can't be changed, as explained here.

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

---

*The sources are in GitHub at yegor256/iexec

\iexec (the default value is "`iexec.tmp`"):

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

The tailing 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 globbaly, 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 tailing spacing after the content, you may use unskip package option, which will append \unskip commmand 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}
```

maybe   If -shell-escape is not set, the \iexec command will lead to compilation failure. This failure may be avoided with the help of the maybe option, which means that the execution of \iexec must be quietly skipped if -shell-escape is not set:

```
\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 parse package options, with the help of pgfopts:

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

Then, we prepare to parse the options of the \iexec command, with the help of pgfkeys:

```
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   maybe/.store in = \iexec@maybe,
25   stdout,exit
26 }\makeatother
```

\iexec@typeout Then, we define an internal command \iexec@typeout for printing the content of a file, as suggested here:

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

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

```
40 \makeatletter
```

```
41 \newread\iexec@exitfile
42 \newcommand\iexec[2][]{%
43   \begingroup%
44     \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:

```
45     \ifnum\ShellEscapeStatus=1%
46       \begingroup%
```

Then, we start the log from a clean line:

```
47       \ifdefined\iexec@log%
48         \message{^^J}%
49       \fi%
```

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

```
50       \let\%\@percentchar%
51       \let\\\@backslashchar%
52       \let\{\@charlb%
53       \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):

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

Then, a message is printed to TeX log:

```
60       \ifdefined\iexec@log%
61         \message{iexec: [\iexec@cmd]^^J}%
62       \fi%
63     \endgroup%
```

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

```
64     \immediate\openin\iexec@exitfile=\iexec@exit%
65     \read\iexec@exitfile to \iexec@code%
66     \immediate\closein\iexec@exitfile%
```

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

```
67     \ifdefined\iexec@null\else%
68       \IfFileExists
69         {\iexec@stdout}
70         {}
71         {\PackageError{iexec}{The "\iexec@stdout" file is absent
72           after processing, looks like some internal error}{}}%
73       \ifdefined\iexec@log%
74         \message{iexec: This is the content of '\iexec@stdout':^^J}%
75         \IfFileExists
76           {\iexec@stdout}
77           {\iexec@typeout{\iexec@stdout}}
78           {\PackageError{iexec}{The "\iexec@stdout" file is absent
79             after processing, looks like some internal error}{}}%
80         \message{<EOF>^^J}%
```

```
81          \else%
82            \ifnum\iexec@code=0\else%
83              \ifdefined\iexec@ignore\else%
84                \message{iexec: See the content of '\iexec@stdout'
85                  after failure:^^J}%
86                \iexec@typeout{\iexec@stdout}%
87                \message{<EOF>^^J}%
88              \fi%
89            \fi%
90          \fi%
91        \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):

```
92        \ifnum\iexec@code=0\else%
93          \ifdefined\iexec@ignore%
94            \ifdefined\iexec@log%
95              \message{iexec: Execution failure ignored,
96                the exit code was \iexec@code^^J}%
97            \fi%
98          \else%
99            \PackageError{iexec}{Execution failure,
100               the exit code was \iexec@code}{}%
101          \fi%
102       \fi%
```

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

```
103       \ifdefined\iexec@null\else%
104       \ifdefined\iexec@quiet%
105         \ifdefined\iexec@log%
106           \message{iexec: Due to 'quiet' option we didn't read
107           the content of '\iexec@stdout'
108           \ifdefined\pdffilesize (\pdffilesize{\iexec@stdout}
109           bytes)\fi^^J}%
110         \fi%
111       \else%
112         \ifdefined\iexec@log%
113           \message{iexec: We are going to include the content of
114           '\iexec@stdout'\ifdefined\pdffilesize (\pdffilesize
115           {\iexec@stdout} bytes)\fi...^^J}%
116         \fi%
117         \input{\iexec@stdout}%
118         \ifdefined\iexec@unskip\unskip\fi%
119         \message{iexec: The content of '\iexec@stdout'
120         was included into the document^^J}%
121       \fi\fi%
```

Then, we delete the file or leave it untouched:

```
122       \ifdefined\iexec@null\else%
123       \ifdefined\iexec@trace%
124         \ifdefined\iexec@log%
125           \message{iexec: Due to package option 'trace',
126           the files '\iexec@stdout' and `\iexec@exit` were
127           not deleted^^J}%
128         \fi%
```

```
129      \else%
130        \ifdefined\iexec@traceit%
131          \ifdefined\iexec@log%
132            \message{iexec: Due to 'trace' package option,
133            the files '\iexec@stdout' and '\iexec@exit'
134            were not deleted^^J}%
135          \fi%
136        \else%
137          \ShellEscape{rm \iexec@stdout}%
138          \ifdefined\iexec@log%
139            \message{iexec: The file '\iexec@stdout' was deleted^^J}%
140          \fi%
141          \ShellEscape{rm \iexec@exit}%
142          \ifdefined\iexec@log%
143            \message{iexec: The file '\iexec@exit' was deleted^^J}%
144          \fi%
145        \fi%
146      \fi\fi%
```

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

```
147      \else%
148        \ifdefined\iexec@maybe%
149          \message{iexec: The execution skipped because -shell-escape
150            is not set and 'maybe' option is provided^^J}%
151        \else%
152          \PackageError{iexec}{You must run TeX processor with
153          --shell-escape option}{}%
154        \fi%
155      \fi%
156    \endgroup%
157 }\makeatother
```

Change History

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.