# iexec: LaTeX Package for Inputable Shell Executions*

Yegor Bugayenko
yegor256@gmail.com

09.10.20220, 0.7.0

## 1 Introduction

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

```
\documentclass{article}
\usepackage{iexec}
\begin{document}
Today is
  \iexec{date +\%e-\%b-\%Y}
\end{document}
```

Today is 9-Oct-2022

`\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 on your user console).

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

## 2 Options

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

```
\documentclass{article}
\usepackage{iexec}
\begin{document}
I just want to delete some file:
\iexec[quiet]{rm -f foo.txt}
\end{document}
```

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

*The sources are in GitHub at yegor256/iexec

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

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

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

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

`log` The stdout produced will be printed in TEX log:

```
\documentclass{article}
\usepackage{iexec}
\begin{document}
\iexec[log]{echo 'Hello, \\LaTeX!'}
\input{foo.txt}
\end{document}
```

# 3 Implementation

First, we include a few packages:

```
\RequirePackage{shellesc}
\RequirePackage{pgfkeys}
\RequirePackage{expl3}
```

Then, we parse package options:

```
\RequirePackage{xkeyval}
\makeatletter\newif\ifiexec@trace
\DeclareOptionX{trace}{\iexec@tracetrue}
\ProcessOptionsX\relax\makeatother
\makeatletter\pgfkeys{
  /iexec/.is family,
```

```
  /iexec,
  stdout/.estore in = \iexec@stdout,
  stdout/.default = iexec.tmp,
  trace/.estore in = \iexec@traceit,
  append/.estore in = \iexec@append,
  log/.estore in = \iexec@log,
  quiet/.estore in = \iexec@quiet,
  stdout
}\makeatother
```

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

```
\makeatletter\ExplSyntaxOn
\NewDocumentCommand{\iexec@typeout}{m}{
  \iexec_typeout_file:n { #1 }}
\ior_new:N \g_iexec_typeout_ior
\cs_new_protected:Nn \iexec_typeout_file:n
{
  \ior_open:Nn \g_iexec_typeout_ior { #1 }
  \ior_str_map_inline:Nn \g_iexec_typeout_ior
    {\iow_term:n { ##1 }}
  \ior_close:N \g_iexec_typeout_ior
}
\ExplSyntaxOff\makeatother
```

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

```
\makeatletter\newcommand\iexec[2][]{%
  \begingroup
    \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:

```
    \ifdefined\pdfshellescape\ifnum\pdfshellescape=1\else%
      \PackageError{iexec}{You must run TeX processor with
      --shell-escape option}{}%
    \fi\fi%
    \begingroup%
```

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

```
      \let\%\@percentchar%
      \let\\\@backslashchar%
      \let\{\@charlb%
      \let\}\@charrb%
```

Then, we execute it:

```
      \ShellEscape{#2 \ifdefined\iexec@append>\fi> \iexec@stdout}%
```

Then, a message is printed to TeX log:

```
      \message{^^Jiexec: [#2 > \iexec@stdout]^^J}%
    \endgroup%
```

Then, if required, the content of the stdout file will be printed to the log:

```
    \ifdefined\iexec@log
```

```
      \message{^^Jiexec: this is the content of \iexec@stdout:^^J}%
      \iexec@typeout{\iexec@stdout}
      \message{^^J<EOF>^^J}%
    \fi%
```

Then, include the produced output into the current document:

```
    \ifdefined\iexec@quiet
      \message{^^Jiexec: Due to 'quiet' option we didn't read
      the content of '\iexec@stdout' (\pdffilesize{} bytes)^^J}%
    \else%
      \message{^^Jiexec: We include the content of
        '\iexec@stdout' (\pdffilesize{} bytes)...^^J}%
      \input{\iexec@stdout}%
      \message{^^Jiexec: The content of '\iexec@stdout'
      was included into the document^^J}%
    \fi%
```

Finally, delete the file or leave it untouched:

```
    \ifiexec@trace%
      \message{^^Jiexec: Due to package option 'trace',
      the file '\iexec@stdout' was not deleted^^J}%
    \else%
      \ifdefined\iexec@traceit%
        \message{^^Jiexec: Due to 'trace' option,
        the file '\iexec@stdout' was not deleted^^J}%
      \else%
        \ShellEscape{rm \iexec@stdout}%
        \message{^^Jiexec: The file '\iexec@stdout' was deleted^^J}%
      \fi%
    \fi%
  \endgroup
}\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.