iexec: LaTeX Package for Inputable Shell Executions*

Yegor Bugayenko
yegor256@gmail.com

2025/01/16, 0.15.0

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 **16-Jan-2025**!

```
\documentclass{article}
\usepackage{iexec}
\usepackage[paperwidth=3in]{geometry}
\pagestyle{empty}
\begin{document}
Today is \textbf{%
  \iexec{date +\%e-\%b-\%Y}}\unskip!
\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.

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:

```
\RequirePackage{shellesc}
```

Then, we parse package options, with the help of pgfopts:

```
\RequirePackage{pgfopts}
\pgfkeys{
  /iexec/.cd,
  trace/.store in=\iexec@trace,
}
\ProcessPgfPackageOptions{/iexec}
```

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

```
\RequirePackage{pgfkeys}
\makeatletter\pgfkeys{
  /iexec/.is family,
  /iexec,
  exit/.store in = \iexec@exit,
  exit/.default = iexec.ret,
  stdout/.store in = \iexec@stdout,
  stdout/.default = iexec.tmp,
  stderr/.store in = \iexec@stderr,
  trace/.store in = \iexec@traceit,
  append/.store in = \iexec@append,
  log/.store in = \iexec@log,
  null/.store in = \iexec@null,
  unskip/.store in = \iexec@unskip,
  quiet/.store in = \iexec@quiet,
  ignore/.store in = \iexec@ignore,
  maybe/.store in = \iexec@maybe,
  stdout,exit
}\makeatother
```

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

```
\RequirePackage{expl3}
\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
```

```
\newread\iexec@exitfile
\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:

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

Then, we start the log from a clean line:

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

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 and save exit code into a file (where we also add `%` in order to trim the content to exactly one number, as suggested here):

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

Then, a message is printed to TEX log:

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

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

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

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

```
      \ifdefined\iexec@null\else%
        \IfFileExists
          {\iexec@stdout}
          {}
          {\PackageError{iexec}{The "\iexec@stdout" file is absent
            after processing, looks like some internal error}{}}%
        \ifdefined\iexec@log%
          \message{iexec: This is the content of '\iexec@stdout'\ifdefined\pdffilesize
            \space(\pdffilesize{\iexec@stdout} bytes)\else\fi:^^J}%
          \IfFileExists
            {\iexec@stdout}
            {\iexec@typeout{\iexec@stdout}}
            {\PackageError{iexec}{The "\iexec@stdout" file is absent
              after processing, looks like some internal error}{}}%
```

```
      \message{<EOF>^^J}%
    \else%
      \ifnum\iexec@code=0\else%
        \ifdefined\iexec@ignore\else%
          \message{iexec: See the content of '\iexec@stdout'\ifdefined\pdffilesize
            \space(\pdffilesize{\iexec@stdout} bytes)\fi
            after failure:^^J}%
          \iexec@typeout{\iexec@stdout}%
          \message{<EOF>^^J}%
        \fi%
      \fi%
    \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):

```
  \ifnum\iexec@code=0\else%
    \ifdefined\iexec@ignore%
      \ifdefined\iexec@log%
        \message{iexec: Execution failure ignored,
          the exit code was \iexec@code^^J}%
      \fi%
    \else%
      \PackageError{iexec}{Execution failure,
        the exit code was \iexec@code}{}%
    \fi%
  \fi%
```

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

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

Then, we delete the file or leave it untouched:

```
  \ifdefined\iexec@null\else%
  \ifdefined\iexec@trace%
    \ifdefined\iexec@log%
      \message{iexec: Due to package option 'trace',
      the files '\iexec@stdout' and `\iexec@exit` were
```

```
          not deleted^^J}%
        \fi%
      \else%
        \ifdefined\iexec@traceit%
          \ifdefined\iexec@log%
            \message{iexec: Due to 'trace' package option,
            the files '\iexec@stdout' and '\iexec@exit'
            were not deleted^^J}%
          \fi%
        \else%
          \ShellEscape{rm \iexec@stdout}%
          \ifdefined\iexec@log%
            \message{iexec: The file '\iexec@stdout' was deleted^^J}%
          \fi%
          \ShellEscape{rm \iexec@exit}%
          \ifdefined\iexec@log%
            \message{iexec: The file '\iexec@exit' was deleted^^J}%
          \fi%
        \fi%
      \fi\fi%
```

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

```
    \else%
      \ifdefined\iexec@maybe%
        \message{iexec: The execution skipped because -shell-escape
          is not set and 'maybe' option is provided^^J}%
      \else%
        \PackageError{iexec}{You must run TeX processor with
        --shell-escape option}{}%
      \fi%
    \fi%
  \endgroup%
}\makeatother

\endinput
```

Change History

0.10.0

\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

The option “ignore” suppresses the checking of “iexec.ret” value. . . 3

0.11.0

\iexec: The file with exit code now contains just numbers, without end of line. 4

The option “exit” allows to change the name of the file with exit code. 3

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 dollar sign with \string command. 4

0.12.0

\iexec: The option “unskip” adds \unskip after each \iexec, in order to trip the tailing end of line space. 3

0.14.0

General: The xkeyval package is not used anymore. Instead, we use pfgopts in order to parse package options. 3

\iexec: The maybe option introduced, allowing the user to skip the entire execution of the \iexec command, when -shell-escape option is off. 6

0.7.0

\iexec: 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

\% 50, 58
\@backslashchar 51
\@charlb 52
\@charrb 53
\@percentchar 50
\\ 51
\{ 52
\} 53

C

\closein 66
\cs 32

D

\def 54

E

\endinput 160
\ExplSyntaxOff 39
\ExplSyntaxOn 28

G

\g 31, 34, 35, 37

I

\iexec 30, 32, 40
\iexec@append 18, 55
\iexec@cmd 54, 59, 61
\iexec@code 65, 83, 94, 98, 102
\iexec@exit . . . 12, 58, 64, 128, 135, 143, 145
\iexec@exitfile 41, 64, 65, 66
\iexec@ignore . . 23, 84, 95
\iexec@log 19, 47, 60, 73, 96, 107, 114, 126, 133, 140, 144
\iexec@maybe 24, 150
\iexec@null 20, 56, 67, 105, 124
\iexec@quiet 22, 106
\iexec@stderr 16, 57
\iexec@stdout 14, 56, 69, 71, 74, 75, 77, 78, 79, 85, 86, 88, 109, 110, 116, 117, 119, 121, 128, 135, 139, 141
\iexec@trace 5, 125
\iexec@traceit . . 17, 132
\iexec@typeout . 27, 78, 88
\iexec@unskip . . . 21, 120
\ifdefined 47, 55, 56, 57, 60, 67, 73, 74, 84, 85, 95, 96, 105, 106, 107, 110, 114, 116, 120, 124, 125, 126, 132, 133, 140, 144, 150
\IfFileExists 68, 76
\ifnum 45, 83, 94
\immediate 64, 66
\input 119
\ior 31, 34, 35, 37
\iow 36

M

\makeatletter . . 9, 28, 40
\makeatother . . 26, 39, 159
\message 48, 61, 74, 81, 85, 89, 97, 108, 115, 121, 127, 134, 141, 145, 151

N

\NewDocumentCommand . 29
\newread 41

O

\openin 64

P

\PackageError 71, 79, 101, 154
\pdffilesize 74, 75, 85, 86, 110, 116
\pgfkeys 3, 9
\pgfqkeys 44
\ProcessPgfPackageOptions 7

R

\read 65
\RequirePackage 1, 2, 8, 27

S

\ShellEscape . 59, 139, 143
\ShellEscapeStatus . . 45
\space 57, 75, 86
\string 58

U

\unskip 120