

# Package: scriptexec (via r-universe)

October 10, 2024

**Title** Execute Native Scripts

**Version** 0.3.1

**Description** Run complex native scripts with a single command, similar to system commands.

**License** Apache License 2.0

**URL** <https://github.com/sagiegurari/scriptexec>

**BugReports** <https://github.com/sagiegurari/scriptexec/issues>

**Depends** R (>= 3.2.3)

**Encoding** UTF-8

**RoxygenNote** 6.1.1

**Suggests** knitr (>= 1.22), testthat (>= 2.0.1), lintr (>= 1.0.3), formatR (>= 1.6), devtools (>= 2.0.2), roxygen2 (>= 6.1.1), rmarkdown (>= 1.12), Rd2md (>= 0.0.2)

**VignetteBuilder** knitr

**Repository** <https://sagiegurari.r-universe.dev>

**RemoteUrl** <https://github.com/sagiegurari/scriptexec>

**RemoteRef** HEAD

**RemoteSha** b6d4d97c5472e1f70587c3d44ba8d0547c48f042

## Contents

build_output . . . . .	2
create_script_file . . . . .	2
create_system_call_args . . . . .	3
execute . . . . .	4
generate_args_setup_script . . . . .	5
generate_env_setup_script . . . . .	6
get_command . . . . .	6
get_platform_value . . . . .	7
is_windows . . . . .	8

modify_script . . . . .	8
on_invoke_error . . . . .	9
scriptexec . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

build_output	<i>Builds the output structure.</i>
--------------	-------------------------------------

---

### Description

Builds the output structure.

### Usage

```
build_output(output, wait)
```

### Arguments

output	The invocation output
wait	A TRUE/FALSE parameter, indicating whether the function should wait for the command to finish, or run it asynchronously

### Value

The script output structure

### Examples

```
output <- c('line 1', '\n', 'line 2')
attr(output, 'status') <- 15
script_output <- build_output(output)
```

---

create_script_file	<i>Creates a temporary file, writes the provided script content into it and returns the file name.</i>
--------------------	--

---

### Description

Creates a temporary file, writes the provided script content into it and returns the file name.

### Usage

```
create_script_file(script = "")
```

### Arguments

script	The script text
--------	-----------------

**Value**

The temporary file name

**Examples**

```
filename <- create_script_file('echo test')
```

---

```
create_system_call_args
```

*Returns the system call arguments.*

---

**Description**

Returns the system call arguments.

**Usage**

```
create_system_call_args(command, cli_args, wait, env, is_windows_os)
```

**Arguments**

command	The command to invoke
cli_args	Possible list of command line arguments
wait	A TRUE/FALSE parameter, indicating whether the function should wait for the command to finish, or run it asynchronously
env	Optional character vector of name=value strings to set environment variables
is_windows_os	True if windows based OS, false for unix based OS

**Value**

The system call arguments

**Examples**

```
filename <- './myfile.sh'  
arg_list <- create_system_call_args('sh', c(filename), TRUE, character(), FALSE)
```

---

execute	<i>Executes a script and returns the output. The stdout and stderr are captured and returned. In case of errors, the exit code will return in the status field.</i>
---------	---

---

## Description

Executes a script and returns the output. The stdout and stderr are captured and returned. In case of errors, the exit code will return in the status field.

## Usage

```
execute(script = "", args = c(), env = character(), wait = TRUE,
        runner = NULL, print_commands = FALSE, get_runtime_script = FALSE)
```

## Arguments

script	The script text
args	Optional script command line arguments (arguments are added as variables in the script named ARG1, ARG2, ...)
env	Optional character vector of name=value strings to set environment variables
wait	A TRUE/FALSE parameter, indicating whether the function should wait for the command to finish, or run it asynchronously (output status will be -1)
runner	The executable used to invoke the script (by default cmd.exe for windows, sh for other platforms)
print_commands	True if to print each command before invocation (not available for windows)
get_runtime_script	True to return the actual invoked script in a script output parameter

## Value

The process output, status code (in case wait=TRUE), error message (in case of any errors) and invoked script in the form of list(status = status, output = output\_text, error = error\_message, script = script)

## Examples

```
library('scriptexec')
library('testthat')

# execute script text
output <- scriptexec::execute('echo command1\necho command2')
expect_equal(output$status, 0)
expect_equal(grepl('command1', output$output), TRUE)
expect_equal(grepl('command2', output$output), TRUE)
```

```

if (.Platform$OS.type == 'windows') {
  ls_command <- 'dir'
} else {
  ls_command <- 'ls'
}
output <- scriptexec::execute(c('echo user home:', ls_command))
expect_equal(output$status, 0)

# execute multiple commands as a script
output <- scriptexec::execute(c('cd', 'echo test'))
expect_equal(output$status, 0)

# pass arguments (later defined as ARG1, ARG2, ...) and env vars
if (.Platform$OS.type == 'windows') {
  command <- 'echo %ARG1% %ARG2% %MYENV%'
} else {
  command <- 'echo $ARG1 $ARG2 $MYENV'
}
output <- scriptexec::execute(command, args = c('TEST1', 'TEST2'), env = c('MYENV=TEST3'))
expect_equal(output$status, 0)
expect_equal(grepl('TEST1 TEST2 TEST3', output$output), TRUE)

# non zero status code is returned in case of errors
expect_warning(output <- scriptexec::execute('exit 1'))
expect_equal(output$status, 1)

# do not wait for command to finish
output <- scriptexec::execute('echo my really long task', wait = FALSE)
expect_equal(output$status, -1)

```

---

```
generate_args_setup_script
```

*Generates and returns a script which sets up the env vars for the script arguments*

---

## Description

Generates and returns a script which sets up the env vars for the script arguments

## Usage

```
generate_args_setup_script(args = character())
```

## Arguments

args                    Optional script command line arguments

## Value

The script text which sets up the env vars for the script arguments

**Examples**

```
script <- generate_args_setup_script(args = c('first', 'second'))
```

---

```
generate_env_setup_script
```

*Generates and returns a script which sets up the env vars for the script execution.*

---

**Description**

Generates and returns a script which sets up the env vars for the script execution.

**Usage**

```
generate_env_setup_script(env = character())
```

**Arguments**

env                   Optional character vector of name=value strings to set environment variables

**Value**

The script text which sets up the env

**Examples**

```
script <- generate_env_setup_script(c('ENV_TEST=MYENV'))
```

---

```
get_command
```

*Returns the command and arguments needed to execute the provided script file on the current platform.*

---

**Description**

Returns the command and arguments needed to execute the provided script file on the current platform.

**Usage**

```
get_command(filename, runner = NULL)
```

**Arguments**

filename           The script file to execute

runner             The executable used to invoke the script (by default cmd.exe for windows, sh for other platforms)

**Value**

A list holding the command and arguments

**Examples**

```
command_struct <- get_command('myfile.sh')
command <- command_struct$command
cli_args <- command_struct$args
```

---

*get\_platform\_value*      *Returns the value based on the current platform.*

---

**Description**

Returns the value based on the current platform.

**Usage**

```
get_platform_value(unix_value = c(), windows_value = c()),
force_windows = FALSE)
```

**Arguments**

- unix\_value      The unix platform value
- windows\_value    The windows platform value
- force\_windows    True to force windows (defaulted to OS validation)

**Value**

unix\_value in case of unix system, else the windows\_value

**Examples**

```
platform_value <- get_platform_value('.sh', '.bat')
```

---

is_windows	<i>Returns true if windows, else false.</i>
------------	---

---

**Description**

Returns true if windows, else false.

**Usage**

```
is_windows()
```

**Value**

True if windows, else false.

**Examples**

```
windows <- is_windows()
```

---

modify_script	<i>Modifies the provided script text and ensures the script content is executed in the correct location.</i>
---------------	--

---

**Description**

Modifies the provided script text and ensures the script content is executed in the correct location.

**Usage**

```
modify_script(script, args = c(), env = character(),
  print_commands = FALSE, is_windows_os = FALSE)
```

**Arguments**

script	The script text
args	Optional script command line arguments
env	Optional character vector of name=value strings to set environment variables
print_commands	True if to print each command before invocation (not available for windows)
is_windows_os	True if windows based OS, false for unix based OS

**Value**

The modified script text

**Examples**

```
script <- modify_script(script = 'echo test', args = c('first', 'second'), env = c('MYENV=MYENV'))
```



---

on_invoke_error	<i>Internal error handler.</i>
-----------------	--------------------------------

---

**Description**

Internal error handler.

**Usage**

```
on_invoke_error(error)
```

**Arguments**

error	The invocation error
-------	----------------------

**Value**

The invocation output

---

scriptexec	<i>scriptexec: Execute native scripts</i>
------------	---

---

**Description**

This package provides one main function: `execute` which executes the provided script and returns its output.

# Index

`build_output`, 2

`create_script_file`, 2

`create_system_call_args`, 3

`execute`, 4

`generate_args_setup_script`, 5

`generate_env_setup_script`, 6

`get_command`, 6

`get_platform_value`, 7

`is_windows`, 8

`modify_script`, 8

`on_invoke_error`, 9

`scriptexec`, 9

`scriptexec-package (scriptexec)`, 9