Package: MakefileR (via r-universe)

May 20, 2026

Encoding UTF-8

Title Create 'Makefiles' Using R

Description A user-friendly interface for the construction of 'Makefiles'.

Version 1.1.0.9000

Date 2025-10-14, 18:02:39

Imports magrittr

Suggests knitr, pkgload, rprojroot, RUnit, testthat, tinytest, fritools, rmarkdown

License GPL-3

URL https://gitlab.com/fvafrcu/MakefileR

VignetteBuilder knitr

RoxygenNote 7.3.3

Author Andreas Dominik Cullmann [aut, cre], Kirill Müller [aut]

Maintainer Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Repository https://fvafrcu.r-universe.dev

Date/Publication 2025-10-21 20:28:53 UTC

RemoteUrl https://gitlab.com/fvafrcu/makefiler

RemoteRef HEAD

RemoteSha c4711ef055c07b5ee6a2662d9b1f87bbcf7facc7

Contents

`MakefileR-package` *Create 'Makefiles' Using R*

Description

A user-friendly interface for the construction of 'Makefiles'.

Details

You will find the details in
`vignette("An_Introduction_to_MakefileR", package = "MakefileR")`.

Author(s)

Maintainer: Andreas Dominik Cullmann <fvafrcu@mailbox.org>

Authors:

- Kirill Müller

See Also

Useful links:

- https://gitlab.com/fvafrcu/MakefileR

`c.MakefileR_group` *Concatenation of rules*

Description

Rules can be appended to groups and Makefiles using the `c` function or the `+` operator.

Usage

```
## S3 method for class 'MakefileR_group'
c(..., recursive = FALSE)

## S3 method for class 'MakefileR_group'
x + y
```

Arguments

`..., x, y`	`[MakefileR]` Rules, the first (`x` or the first element of `...`) must be of class `MakefileR_group` (created by `make_group` or `makefile`)
`recursive`	`[any]` Unused

Examples

```
c(make_group(sep = ""),
  make_group(make_comment("Dummy targets"),
             make_rule(".FORCE"), make_rule(".SILENT")),
  make_group(make_comment("Definitions"),
             make_def("A", "a")))

makefile() + (make_group() + make_comment("Definitions") + make_def("A", "a"))
```

---

make_comment *Creates a Makefile comment*

Description

For helping the reader understand what's happening

Usage

```
make_comment(...)
```

Arguments

...	[character] Comments without leading hash #

Details

Use the c function or the + operator to append comments to groups and Makefiles.

Value

An object of class MakefileR_comment

References

https://www.gnu.org/software/make/manual/

See Also

Other items: make_def(), make_group(), make_rule(), make_text()

Examples

```
make_comment("This is a comment")
```

make_def *Creates a variable definition in a Makefile*

Description

A variable definition in a `Makefile` consists of a variable name and its defition. Both are separated by the equality sign `=`.

Usage

```
make_def(variable, definition, operator = "=")
```

Arguments

`variable`	`[character(1)]` Variable name
`definition`	`[character(1)]` Definition for this variable
`operator`	`[character(1)]` Which operator to use, default: `=`

Details

No quoting is applied to the definition by this function. Currently, both variable and definition are required to be character values of length one.

Use the `c` function or the `+` operator to append definitions to groups and Makefiles.

Value

An object of class `MakefileR_def`

References

https://www.gnu.org/software/make/manual/

See Also

`makefile`, `make_group`

Other items: `make_comment()`, `make_group()`, `make_rule()`, `make_text()`

Examples

```
make_def("R_USER_LIBRARY", .libPaths()[[1L]])
makefile() +
  make_def("R_USER_LIBRARY", .libPaths()[[1L]])
```

make_group *Creates a group of Makefile items*

Description

Helps separating similar rules.

Usage

```
make_group(..., .dots = NULL, sep = NULL)
```

Arguments

...	[MakefileR] Items created by make_rule or other make_ functions
.dots	[list] Further rules in addition to ...
sep	[character(1)] Separator between group items, NULL (the default) means no separator.

Details

Use the c function or the + operator to append groups to other groups and Makefiles (thus creating nested groups).

Value

An object of class MakefileR_group

References

https://www.gnu.org/software/make/manual/

See Also

c.MakefileR_group

Other items: make_comment(), make_def(), make_rule(), make_text()

Examples

```
makefile(make_rule("all", c("first_target", "second_target")))
```

make_rule *Creates a Makefile rule*

Description

A rule in a `Makefile` consists of a (list of) targets which may depend on one or more dependencies each. Optionally, a script is executed to create the target. Generally, multiple targets mean that the rule is identical for each of the individual targets, and multiple dependencies mean that *all* of them are required to build *each* of the targets. In the script, the target can be referenced by `$@`, and the first dependency can be referenced by `$<`. Note that the dollar sign has a special meaning in a `Makefile`, use `$$` in scripts that need to use the dollar sign themselves.

Usage

```
make_rule(targets, deps = NULL, script = NULL)
```

Arguments

`targets`	`[character]` Target names
`deps`	`[character]` Dependency names
`script`	`[character]` A script to execute to build the targets.

Details

Use the `c` function or the `+` operator to append rules to groups and Makefiles.

Value

An object of class `MakefileR_rule`

References

https://www.gnu.org/software/make/manual/

See Also

`makefile`

Other items: `make_comment()`, `make_def()`, `make_group()`, `make_text()`

Examples

```
make_rule("all", c("first_target", "second_target"))
make_rule(".FORCE")
make_rule("first_target", ".FORCE", "echo 'Building first target'")
make_rule("second_target", "first_target",
 c("echo 'Building second target'", "echo 'Done'"))

makefile() +
  make_rule("all", c("first_target", "second_target")) +
  make_rule(".FORCE") +
  make_rule("first_target", ".FORCE", "echo 'Building first target'") +
  make_rule("second_target", "first_target",
    c("echo 'Building second target'", "echo 'Done'"))
```

---

make_text *Creates a custom Makefile entry*

Description

For anything else not covered by the other `make_` functions, such as the `export` statement for exporting Makefile variables.

Usage

```
make_text(...)
```

Arguments

`...`	`[character]` Custom text

Details

Use the `c` function or the `+` operator to append comments to groups and Makefiles.

Value

An object of class `MakefileR_text`

References

https://www.gnu.org/software/make/manual/

See Also

Other items: `make_comment()`, `make_def()`, `make_group()`, `make_rule()`

Examples

```
make_text("export SOME_VARIABLE")
```

`makefile` *Creates a Makefile*

Description

A `Makefile` consists of a list of rules, definition, comments and other items.

Usage

```
makefile(..., .dots = NULL)
```

Arguments

`...`	`[MakefileR]` Items created by `make_rule` or other `make_` functions
`.dots`	`[list]` Further rules in addition to `...`

Details

Use the `c` function or the `+` operator to append rules, definitions, comments, plain text, and groups.

Value

An object of class `MakefileR_file`

References

https://www.gnu.org/software/make/manual/

See Also

`make_rule`, `make_def`, `make_comment`, `make_text`, `make_group`, `c.MakefileR_group`

Examples

```
makefile(make_rule("all", c("first_target", "second_target")))
```

`write_makefile` *Writes a Makefile to a file*

Description

Makefiles, as created by `makefile`, only exist in memory until they are written to a file by this function.

Usage

```
write_makefile(makefile, file_name)
```

Arguments

`makefile`	`[MakefileR]` A Makefile, created by `makefile`
`file_name`	`[character(1)]` Target file name

Value

The value returned by `writeLines`

Index