

Package: fakemake (via r-universe)

October 9, 2024

Title Mock the Unix Make Utility

Version 1.11.1.9000

Description Use R as a minimal build system. This might come in handy if you are developing R packages and can not use a proper build system. Stay away if you can (use a proper build system).

License BSD_2_clause + file LICENSE

URL <https://gitlab.com/fvafrcu/fakemake>

Depends R (>= 3.3.0)

Imports graphics, fritools, igraph, MakefileR, utils

Suggests cleanr, covr, cyclocomp, devtools, hunspell, knitr, lintr, pkgbuild, pkgload, rasciidoc, rmarkdown, roxygen2, rprojroot, RUnit, spelling, testthat, withr, usethis

VignetteBuilder knitr, rasciidoc

Encoding UTF-8

RoxygenNote 7.2.3

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

RemoteUrl <https://gitlab.com/fvafrcu/fakemake>

RemoteRef HEAD

RemoteSha 21c468e8aa0142dd0d0eb9baf3eabca77a6d6e9d

Contents

fakemake-package	2
add_target	2
get_target	3
make	4
provide_make_list	5
remove_target	6
visualize	7
Index	8

fakemake-package	<i>Mock the Unix Make Utility</i>
------------------	-----------------------------------

Description

Use R as a minimal build system. This might come in handy if you are developing R packages and can not use a proper build system. Stay away if you can (use a proper build system).

Details

You will find the details in
 vignette("An_Introduction_to_fakemake", package = "fakemake").

add_target	<i>Add a Target to a Makelist</i>
------------	-----------------------------------

Description

Add a target to an existing makelist.

Usage

```
add_target(
  makelist,
  target,
  code,
  prerequisites = NULL,
  prerequisite_to = NULL,
  sink = NULL,
  alias = sub("\\.(Rout|log)$", "", basename(target))
)
```

Arguments

makelist	A list for make .
target	The target to remove from makelist.
code	The code for the new target.
prerequisites	The prerequisites for the new target.
prerequisite_to	The targets the new target is a prerequisite to. Set to TRUE to add it as a prerequisite to all existing targets.
sink	The sink for the new target.
alias	The alias for the new target.

Value

A list for [make](#).

See Also

Other functions to manipulate makelists: [get_target\(\)](#), [remove_target\(\)](#)

<code>get_target</code>	<i>Get a Makelist's Target</i>
-------------------------	--------------------------------

Description

Get a single target from a makelist by alias.

Usage

```
get_target(makelist, alias)
```

Arguments

<code>makelist</code>	A list for make .
<code>alias</code>	The alias of the target in question.

Value

A list (the target requested).

See Also

Other functions to manipulate makelists: [add_target\(\)](#), [remove_target\(\)](#)

Examples

```
m1 <- provide_make_list()
visualize(m1, root = "all.Rout")
i <- which(sapply(m1, "[[", "target") == "b1.Rout")
m1[[i]][["alias"]] <- "b1"
t <- get_target(m1, "b1")
m1 <- remove_target(m1, t[["target"]])
visualize(m1)
m1 <- add_target(m1, target = t[["target"]], code = t[["code"]],
                sink = t[["sink"]],
                prerequisite_to = "a1.Rout", alias = NULL)
all.equal(m1, provide_make_list())
```

 make

Mock the Unix Make Utility

Description

Mock the Unix Make Utility

Usage

```
make(
  name,
  make_list,
  force = FALSE,
  recursive = force,
  verbose = FALSE,
  verbosity = 2,
  dry_run = FALSE,
  unconditionally = FALSE,
  stop_on_warning = FALSE
)
```

Arguments

name	The name or alias of a make target.
make_list	The makelist (a listed version of a Makefile).
force	Force the target to be build? See <i>Details</i> .
recursive	Force the target to be build recursively? See <i>Details</i> .
verbose	Be verbose?
verbosity	Give the level of verbosity.
dry_run	Run dry? Mock GNU make's -n option.
unconditionally	Force the target's code to be evaluated unconditionally to any prerequisites? See <i>Details</i> .
stop_on_warning	Throw an error and abort if a recipe throws a warning ?

Details

force, recursive

Forcing a target mocks adding .PHONY to a GNU Makefile if you set recursive to FALSE. If recursive is TRUE, then the whole make chain will be forced.

unconditionally

Setting unconditionally to [TRUE](#) allows you to fool [make](#) similarly to using GNU make's -touch option.

Value

Invisibly a character vector containing the targets made during the current run.

Examples

```
str(make_list <- provide_make_list("minimal"))
# build all
withr::with_dir(tempdir(), print(make("all.Rout", make_list)))
# nothing to be done
withr::with_dir(tempdir(), print(make("all.Rout", make_list)))
# forcing all.Rout
withr::with_dir(tempdir(), print(make("all.Rout", make_list, force = TRUE,
                                     recursive = FALSE)))
# forcing all.Rout recursively
withr::with_dir(tempdir(), print(make("all.Rout", make_list, force = TRUE)))

# show files
dir(tempdir(), pattern = ".*\\.Rout")

# dry run
file.remove(dir(tempdir(), pattern = ".*\\.Rout", full.names = TRUE))
withr::with_dir(tempdir(), print(make("all.Rout", make_list,
                                     dry_run = TRUE)))
dir(tempdir(), pattern = ".*\\.Rout")

# make unconditionally
dir(tempdir(), pattern = ".*\\.Rout")
withr::with_dir(tempdir(), print(make("all.Rout", make_list,
                                     unconditionally = TRUE)))
dir(tempdir(), pattern = ".*\\.Rout")
```

provide_make_list *Load an Example Makelist Provided by **fakemake**.*

Description

Load an Example Makelist Provided by **fakemake**.

Usage

```
provide_make_list(
  type = c("minimal", "testing"),
  prune = TRUE,
  clean_sink = FALSE
)
```

Arguments

type	The type of makelist. package makelist.
prune	Prune the makelist of NULL items?
clean_sink	Remove sinks identical to corresponding targets from the list? Since makelists are parsed, missing sinks are set to the corresponding targets, but this makes them harder to read.

Value

A makelist.

Examples

```
str(provide_make_list("minimal"))
visualize(provide_make_list("minimal"))
```

remove_target	<i>Remove a Target From a Makelist</i>
---------------	--

Description

Remove a target and all its appearances as other targets' dependencies from a makelist.

Usage

```
remove_target(makelist, target)
```

Arguments

makelist	A list for make .
target	The target to remove from makelist.

Value

A list for [make](#).

See Also

Other functions to manipulate makelists: [add_target\(\)](#), [get_target\(\)](#)

visualize	<i>Visualize a Makelist</i>
-----------	-----------------------------

Description

Parse a makelist, convert it into an igraph and plot it.

Usage

```
visualize(make_list, root = NULL)
```

Arguments

make_list	The makelist.
root	The root of a tree.

Value

Invisibly an **igraph** representation of the makelist.

Examples

```
str(ml <- provide_make_list())  
visualize(ml)  
visualize(ml, root = "all.Rout")
```

Index

* functions to manipulate makelists

- add_target, 2
- get_target, 3
- remove_target, 6

* package

- fakemake-package, 2

add_target, 2, 3, 6

fakemake-package, 2

get_target, 3, 3, 6

Invisibly, 5

make, 2-4, 4, 6

provide_make_list, 5

remove_target, 3, 6

TRUE, 2, 4

visualize, 7

warning, 4