Package: maketools (via r-universe)

November 4, 2024

Type Package

Title Exploring and Testing the Toolchain and System Libraries

Version 1.3.1

Description Helper functions that interface with the system utilities
to learn about the local build environment. Lets you explore
'make' rules to test the local configuration, or query
'pkg-config' to find compiler flags and libs needed for
building packages with external dependencies. Also contains
tools to analyze which libraries that a installed R package
linked to by inspecting output from 'ldd' in combination with
information from your distribution package manager, e.g. 'rpm'
or 'dpkg'.

License MIT + file LICENSE

URL https://jeroen.r-universe.dev/maketools

BugReports https://github.com/jeroen/maketools/issues

Encoding UTF-8

Imports sys (>= 3.1)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

VignetteBuilder knitr

Suggests curl, knitr, rmarkdown, testthat

Language en-US

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

RemoteUrl https://github.com/jeroen/maketools

RemoteRef HEAD

RemoteSha d46f92ca8e11a85eece34ff72a0a50b5143d3636

Contents

`diagnostics` *Diagnostics Report*

Description

Print some diagnostics about your compiler environment. These are also shown when the `maketools` package is attached.

Usage

```
maketools_diagnostics()
```

See Also

Other maketools: `make()`, `pkgconfig`, `r_config`, `sysdeps`

`find_logo` *Package tools*

Description

Get some extra info about packages.

Usage

```
find_logo(path = ".")
```

Arguments

`path`	root directory of package

make *Make*

Description

Compile C / C++ / Fortran source files using the compiler configured by your R `Makeconf` file.

Usage

```
make(target = "all", makefile = r_makeconf_path())

make_call(cmd = "$(CC)", args = "--version")

make_echo(cmd = "$(CC)")

make_info()
```

Arguments

`target` name of output file that you want to make

`makefile` path to the `Makefile`. Defaults to the `Makeconf` which R uses when building R packages.

`cmd` command to invoke (may be a variable)

`args` additional arguments for `cmd`

Details

The `make` function literally calls `make yourfile.o -f /path/to/R/Makeconf`. This is exactly what R does when building packages and hence the best way to test if the compiler is working.

See Also

Other maketools: `diagnostics`, `pkgconfig`, `r_config`, `sysdeps`

Examples

```
# Test the CXX compiler
if(cxx_info()$available){
testprog <- '#include <iostream>\nint main() {std::cout << "Hello World!";}'
writeLines(testprog, con = 'testprog.cc')
make('testprog')

# Test and cleanup
system('./testprog')
unlink('testprog*', recursive = TRUE)
}

# Run a program from a make variable
```

```
make_call('$(CXX)', '--version')

# Where your makeconf is stored:
make_info()
```

pkgconfig *Query pkg-config*

Description

Wrappers for the pkg-config utility to query information on C/C++ libraries that are available on your system.

Usage

```
pc_info()

pc_pkg_list()

pc_pkg_exists(pkg = "libcurl")

pc_pkg_version(pkg = "libcurl")

pc_pkg_cflags(pkg = "libcurl")

pc_pkg_libs(pkg = "libcurl", static = FALSE)

pc_pkg_info(pkg = "libcurl")
```

Arguments

`pkg`	names of the pkg-config libraries to query
`static`	get libs for static linking, i.e. include dependencies

See Also

Other maketools: `diagnostics`, `make()`, `r_config`, `sysdeps`

Examples

```
# Check if pkg-config is available
(info <- pc_info())
if(info$available)
  pc_pkg_list()
```

r_config *R CMD Config*

Description

Cross-platform wrappers for `R CMD config` to lookup the availability of the compiler.

Usage

```
cc_info()

cxx_info()

cxx11_info()

cxx14_info()

cxx17_info()

fc_info()

r_cmd_config(VAR = "--all")
```

Arguments

`VAR`	value passed to `R CMD config` such as `CXX` or `FC`

See Also

Other maketools: `diagnostics`, `make()`, `pkgconfig`, `sysdeps`

Examples

```
# This runs 'R CMD CONFIG CXX'
r_cmd_config("CXX")

# Show C++ config:
cxx_info()
```

sysdeps *Package System Dependencies*

Description

Shows the external shared libraries that an installed R package is linked to by running `ldd` on the package `so` file. Then uses system package manager (e.g. `dpkg` or `rpm` or `brew`) to locate which system package that contains the binaries, headers, and (if available) sources for this library.

Usage

```
package_sysdeps(pkg, lib.loc = NULL)

package_sysdeps_string(pkg, lib.loc = NULL)

package_links_to(pkg, lib.loc = NULL)
```

Arguments

`pkg`	name of an installed R package
`lib.loc`	path to the R package directory for this package

Details

For common distributions, the output also includes a URL to the distro-homepage of the system package. Here we can typically find more information about the package, such as configuration options, dependencies, and custom patches applied by your distribution.

Because we use `ldd`, this only shows run-time dependencies of an installed R package. This is especially relevant if you distribute the compiled R package in binary form, because the same external libraries need to be available on the user/deployment machine. This tool does not show dependencies that are only needed at build-time, such as static or header-only libraries, and other utilities required to build the package.

See Also

Other maketools: `diagnostics`, `make()`, `pkgconfig`, `r_config`

Index