Rake Ruby task runner

START HERE

`rake`
Run the default task

`rake <task>`
Run one task

`rake -T`
List tasks that have a desc

`rake -P`
Show tasks and their prerequisites

`rake -n <task>`
Dry run: show what would run

`rake -j <N>`
Run with N threads

`rake <task>[<arg>,<arg>]`
Pass task arguments

`rake -t <task>`
Trace invoke and execute

Rakefile Ruby DSL → *load* → **Task graph** tasks + prereqs → *resolve* → **Prerequisites** each runs once → *then* → **Task block** Ruby code

Rake resolves prerequisites before running a task

CONCEPTS

A Rakefile is plain Ruby. A task runs its prerequisites first, once per run, then its block. Use do...end, not braces, so prerequisites bind correctly.

Rake looks for Rakefile, rakefile, Rakefile.rb or rakefile.rb in the current directory, then in parent directories. Tasks run from the Rakefile's directory.

Only tasks with a desc appear in rake -T. Use rake -T -A or rake -P to see the rest. Quote task arguments in zsh: rake 'build[x]'.

RUN

`rake`
Run the default task

`rake <task> <task>`
Run several tasks in order

`rake <ns>:<task>`
Run a namespaced task

`rake -n <task>`
Dry run: print tasks without executing

`rake -j <N>`
Run up to N tasks concurrently
Only tasks that are independent run in parallel; multitask and -m force it.

RUN (cont.)

`rake -m`
Treat all tasks as multitasks

`rake -f <file>`
Use another Rakefile

`rake -N`
Do not search parent directories

`rake <NAME>=<value>`
Set an ENV variable for the run

`bundle exec rake <task>`
Run with the bundled gem versions

INSPECT

`rake -T`
List tasks that have a desc

`rake -T <pattern>`
List tasks matching pattern

`rake -T -A`
List all tasks, including undocumented

`rake -D <task>`
Show the long description of tasks

`rake -P`
List tasks with prerequisites

`rake -W <task>`
Show where a task is defined

`rake --rules`
Trace how rules resolve tasks

`rake -V`
Print the Rake version

INSPECT (cont.)

`rake -H`
Show help

DEBUG AND OUTPUT

`rake -t`
Trace invoke and execute, with backtraces

`rake --backtrace`
Show full backtraces on errors

`rake -v`
Echo shell commands run by FileUtils

`rake -q`
Do not echo FileUtils commands

`rake -s`
Quiet, and no in directory message

DEFINE TASKS

`task :<name> do ... end`
Define a task

`task <name>: %w[<a> <b>]`
Task with prerequisites

`task default: :<name>`
Set the default task

`desc '<text>'`
Describe the next task for -T

`namespace :<ns> do ... end`
Group tasks under a prefix

DEFINE TASKS (cont.)

`task :<name>, [:<a>, :<b>] do |t, args|`
Task with named arguments

`args.with_defaults(<a>: <v>)`
Default values for arguments

`task :<name> => :environment`
Load Rails app first (Rails)

`Rake::Task[:<name>].invoke`
Call a task once from Ruby
Skipped if it already ran in this process.

`Rake::Task[:<name>].reenable`
Allow a task to run again

`Rake::Task[:<name>].execute`
Run a task's actions, ignoring prerequisites

`task :<name>, [:<a>] => [:<pre>]`
Arguments plus prerequisites

`multitask :<name> => %w[<a> <b>]`
Run prerequisites in parallel

FILES AND HELPERS

`file '<out>' => %w[<src>] do |t|`
Rebuild when a source is newer

FILES AND HELPERS (cont.)

`directory '<path>'`
Create a directory (and parents) on demand

`rule '.o' => '.c' do |t|`
Pattern rule for generated files

`t.name / t.source`
Target name / rule source in a block

`t.prerequisites`
Prerequisite names in a block

`FileList['<glob>']`
Lazy list of files by glob

`sh '<cmd>'`
Run a shell command, fail the task on error

`cp / mv / rm_rf / mkdir_p`
FileUtils helpers available in tasks

`require 'rake/clean'`
Add clean and clobber tasks

`CLEAN.include('<glob>')`
Files removed by rake clean

`Rake::TestTask.new do |t| ... end`
Define a test task

`rake <task>[<a>,<b>]`
Pass arguments on the command line

`rake 'name[a b,c]'`
Quote arguments that contain spaces

RECIPES

Documented tasks with arguments

```
desc "Deploy to an environment"
task :deploy, [:env] => :build
do |t, args|
  args.with_defaults(env:
"staging")
  sh "./deploy.sh #{args.env}"
end
# rake deploy[production]
```

Compile C files incrementally

```
SRC = FileList["src/*.c"]
OBJ = SRC.ext("o")

rule ".o" => ".c" do |t|
  sh "cc -c -o #{t.name}
#{t.source}"
end

file "app" => OBJ do |t|
  sh "cc -o #{t.name}
#{t.prerequisites.join(' ')}"
end
```

Namespaces and a default task

```
namespace :db do
  desc "Run migrations"
  task :migrate do ... end
end

task default: %w[db:migrate
test]
```

Inspect a big Rakefile

```
rake -T -A
rake -P | grep <task>
rake -W <task>
rake -n -t <task>
```