Ninja Fast build executor

CONCEPTS

Ninja is a fast executor, not a build language: a generator such as CMake or Meson writes build.ninja for you.

CMakeLists / meson.build you write
↓ `cmake / meson setup`
build.ninja generated
↓ `ninja`
Outputs only what is stale

Ninja skips the work when outputs are newer than their inputs and the recorded command is unchanged.

BUILDING

`ninja`
Build the default targets

`ninja` *`<target>`*
Build only the named target(s)

`ninja -C` *`<dir>`*
Change to a directory first

`ninja -j` *`<N>`*
Run N jobs in parallel

`ninja -n`
Dry run, do not execute

`ninja -v`
Show full command lines

`ninja -k` *`<N>`*
Keep going until N jobs fail

`ninja -f` *`<file>`*
Use another build file

`ninja -l` *`<N>`*
Hold back at load average above N

`ninja --quiet`
Hide progress, show command output

`ninja --version`
Print the Ninja version

`ninja -h`
Show help and options

INSPECTING THE GRAPH

`ninja -t targets all`
List every target and its rule

`ninja -t targets rule` *`<rule>`*
List targets built by a rule

INSPECTING THE GRAPH (cont.)

`ninja -t commands` *`<target>`*
Print commands to rebuild a target

`ninja -t query` *`<path>`*
Show a file's inputs and outputs

`ninja -t inputs` *`<target>`*
List all inputs of a target

`ninja -t graph` *`<target>`*
Emit a Graphviz dot file

CLEANING AND TOOLING

`ninja -t clean`
Remove all built files

`ninja -t clean` *`<target>`*
Clean one target and its deps

`ninja -t compdb` *`<rule>`* **`...`**
Dump compile_commands.json

DEBUGGING AND ENV

`ninja -d explain`
Say why each command reruns

`NINJA_STATUS="[%f/%t %e] " ninja`
Customize progress line

`ninja -j1 -v`
Serial verbose build to find a failure

WRITING BUILD.NINJA

`rule` *`<name>`*
Declare a rule with a command

`command = cc -c $in -o $out`
Rule command using $in and $out

`build` *`<out>`***`:`** *`<rule>`* *`<in>`*
Declare a build edge

`build` *`<out>`***`:`** *`<rule>`* *`<in>`* **`|`** *`<dep>`*
Add an implicit dependency

`build` *`<out>`***`:`** *`<rule>`* *`<in>`* **`||`** *`<dep>`*
Add an order-only dependency

`default` *`<target>`*
Set what a bare ninja builds

`build` *`<name>`***`: phony`** *`<targets>`*
Alias for other targets

`depfile = $out.d`
Read header deps from a depfile

`description = CC $out`
Short text shown while building