NAME

cmake − CMake Command-Line Reference

SYNOPSIS

Generate a Project Buildsystem

```
cmake [<options>] <path-to-source>
cmake [<options>] <path-to-existing-build>
cmake [<options>] -S <path-to-source> -B <path-to-build>
```

Build a Project

```
cmake --build <dir> [<options>] [-- <build-tool-options>]
```

Install a Project

```
cmake --install <dir> [<options>]
```

Open a Project

```
cmake --open <dir>
```

Run a Script

```
cmake [{-D <var>=<value>}...] -P <cmake-script-file>
```

Run a Command-Line Tool

```
cmake -E <command> [<options>]
```

Run the Find-Package Tool

```
cmake --find-package [<options>]
```

View Help

```
cmake --help[-<topic>]
```

DESCRIPTION

The **cmake** executable is the command−line interface of the cross−platform buildsystem generator CMake. The above *Synopsis* lists various actions the tool can perform as described in sections below.

To build a software project with CMake, *Generate a Project Buildsystem*. Optionally use**cmak e** to *Build a Project*, *Install a Project* or just run the corresponding build tool (e.g. **make**) directly. **cmake** can also be used to *View Help*.

The other actions are meant for use by software developers writing scripts in the **CMake language** to support their builds.

For graphical user interfaces that may be used in place of **cmake**, see **ccmake** and **cmake−gui**. For command−line interfaces to the CMake testing and packaging facilities, see **ctest** and **cpack**.

For more information on CMake at large, *see also* the links at the end of this manual.

INTRODUCTION TO CMAKE BUILDSYSTEMS

A *buildsystem* describes how to build a project's executables and libraries from its source code using a *build tool* to automate the process. For example, a buildsystem may be a **Makefile** for use with a command−line **make** tool or a project file for an Integrated Development Environment (IDE). In order to avoid maintaining multiple such buildsystems, a project may specify its buildsystem abstractly using files written in the **CMake language**. From these files CMake generates a preferred buildsystem locally for each user through a backend called a *generator*.

To generate a buildsystem with CMake, the following must be selected:

Source Tree
The top−level directory containing source files provided by the project. The project specifies its buildsystem using files as described in the **cmake−language(7)** manual, starting with a top−level file named **CMakeLists.txt**. These files specify build targets and their dependencies as described in the **cmake−buildsystem(7)** manual.

Build Tree
The top−level directory in which buildsystem files and build output artifacts (e.g. executables and libraries) are to be stored. CMake will write a **CMakeCache.txt** file to identify the directory as a build tree and store persistent information such as buildsystem configuration options.

To maintain a pristine source tree, perform an *out−of−source* build by using a separate dedicated build tree. An *in−source* build in which the build tree is placed in the same directory as the source tree is also supported, but discouraged.

Generator
This chooses the kind of buildsystem to generate. See the **cmake−generators(7)** manual for documentation of all generators. Run **cmake −−help** to see a list of generators available locally. Optionally use the **−G** option below to specify a generator, or simply accept the default CMake chooses for the current platform.

When using one of the Command−Line Build Tool Generators CMake expects that the environment needed by the compiler toolchain is already configured in the shell. When using one of the IDE Build Tool Generators, no particular environment is needed.

GENERATE A PROJECT BUILDSYSTEM

Run CMake with one of the following command signatures to specify the source and build trees and generate a buildsystem:

cmake [<options>] <path−to−source>
Uses the current working directory as the build tree, and **<path−to−source>** as the source tree. The specified path may be absolute or relative to the current working directory. The source tree must contain a **CMakeLists.txt** file and must *not* contain a **CMakeCache.txt** file because the latter identifies an existing build tree. For example:

```
$ mkdir build ; cd build
$ cmake ../src
```

cmake [<options>] <path−to−existing−build>
Uses **<path−to−existing−build>** as the build tree, and loads the path to the source tree from its **CMakeCache.txt** file, which must have already been generated by a previous run of CMake. The specified path may be absolute or relative to the current working directory. For example:

```
$ cd build
$ cmake .
```

cmake [<options>] −S <path−to−source> −B <path−to−build>
Uses **<path−to−build>** as the build tree and **<path−to−source>** as the source tree. The specified paths may be absolute or relative to the current working directory. The source tree must contain a **CMakeLists.txt** file. The build tree will be created automatically if it does not already exist. For example:

```
$ cmake −S src −B build
```

In all cases the **<options>** may be zero or more of the *Options* below.

After generating a buildsystem one may use the corresponding native build tool to build the project. For example, after using the **Unix Makefiles** generator one may run **make** directly:

```
$ make
$ make install
```

Alternatively, one may use **cmake** to *Build a Project* by automatically choosing and invoking the appropriate native build tool.

Options

−S <path−to−source>
Path to root directory of the CMake project to build.

−B <path−to−build>
Path to directory which CMake will use as the root of build directory.

If the directory doesn't already exist CMake will make it.

−C <initial−cache>
Pre−load a script to populate the cache.

When CMake is first run in an empty build tree, it creates a **CMakeCache.txt** file and populates it with customizable settings for the project. This option may be used to specify a file from which to load cache entries before the first pass through the project's CMake listfiles. The loaded entries take priority over the project's default values. The given file should be a CMake script containing **set()** commands that use the **CACHE** option, not a cache−format file.

References to **CMAKE_SOURCE_DIR** and **CMAKE_BINARY_DIR** within the script evaluate to the top−level source and build tree.

−D <var>:<type>=<value>, −D <var>=<value>
Create or update a CMake **CACHE** entry.

When CMake is first run in an empty build tree, it creates a **CMakeCache.txt** file and populates it with customizable settings for the project. This option may be used to specify a setting that takes priority over the project's default value. The option may be repeated for as many **CACHE** entries as desired.

If the **:<type>** portion is given it must be one of the types specified by the **set()** command documentation for its **CACHE** signature. If the **:<type>** portion is omitted the entry will be created with no type if it does not exist with a type already. If a command in the project sets the type to **PATH** or **FILEPATH** then the **<value>** will be converted to an absolute path.

This option may also be given as a single argument: **−D<var>:<type>=<value>** or **−D<var>=<value>**.

−U <globbing_expr>
Remove matching entries from CMake **CACHE**.

This option may be used to remove one or more variables from the **CMakeCache.txt** file, globbing expressions using ***** and **?** are supported. The option may be repeated for as many **CACHE** entries as desired.

Use with care, you can make your **CMakeCache.txt** non−working.

−G <generator−name>
Specify a build system generator.

CMake may support multiple native build systems on certain platforms. A generator is responsible for generating a particular build system. Possible generator names are specified in the **cmake−generators(7)** manual.

If not specified, CMake checks the **CMAKE_GENERATOR** environment variable and otherwise falls back to a builtin default selection.

−T <toolset−spec>
Toolset specification for the generator, if supported.

Some CMake generators support a toolset specification to tell the native build system how to choose a compiler. See the**CMAKE_GENERA TOR_TOOLSET** variable for details.

−A <platform−name>
Specify platform name if supported by generator.

Some CMake generators support a platform name to be given to the native build system to choose a compiler or SDK. See the **CMAKE_GENERATOR_PLATFORM** variable for details.

−−toolchain <path−to−file>
Specify the cross compiling toolchain file, equivalent to setting **CMAKE_TOOLCHAIN_FILE** variable.

−−install−prefix <directory>
Specify the installation directory, used by the **CMAKE_INSTALL_PREFIX** variable. Must be an absolute path.

−Wno−dev
Suppress developer warnings.

Suppress warnings that are meant for the author of the **CMakeLists.txt** files. By default this will also turn off deprecation warnings.

−Wdev Enable developer warnings.

Enable warnings that are meant for the author of the **CMakeLists.txt** files. By default this will also turn on deprecation warnings.

−Werror=dev
Make developer warnings errors.

Make warnings that are meant for the author of the **CMakeLists.txt** files errors. By default this will also turn on deprecated warnings as errors.

−Wno−error=dev
Make developer warnings not errors.

Make warnings that are meant for the author of the **CMakeLists.txt** files not errors. By default this will also turn off deprecated warnings as errors.

−Wdeprecated
Enable deprecated functionality warnings.

Enable warnings for usage of deprecated functionality, that are meant for the author of the **CMakeLists.txt** files.

−Wno−deprecated
Suppress deprecated functionality warnings.

Suppress warnings for usage of deprecated functionality, that are meant for the author of the **CMakeLists.txt** files.

−Werror=deprecated
Make deprecated macro and function warnings errors.

Make warnings for usage of deprecated macros and functions, that are meant for the author of the **CMakeLists.txt** files, errors.

−Wno−error=deprecated
Make deprecated macro and function warnings not errors.

Make warnings for usage of deprecated macros and functions, that are meant for the author of the **CMakeLists.txt** files, not errors.

−L[A][H]
List non−advanced cached variables.

List **CACHE** variables will run CMake and list all the variables from the CMake **CACHE** that are not marked as **INTERNAL** or **ADVANCED**. This will effectively display current CMake settings, which can then be changed with **−D** option. Changing some of the variables may result in more variables being created. If **A** is specified, then it will display also advanced variables. If**H** is specified, it will also display help for each variable.

−N View mode only.

Only load the cache. Do not actually run configure and generate steps.

−−graphviz=[file]
Generate graphviz of dependencies, see **CMakeGraphVizOptions** for more.

Generate a graphviz input file that will contain all the library and executable dependencies in the project. See the documentation for**CMak eGraphVizOptions** for more details.

−−system−information [file]
Dump information about this system.

Dump a wide range of information about the current system. If run from the top of a binary tree for a CMake project it will dump additional information such as the cache, log files etc.

−−log−level=<ERROR|WARNING|NOTICE|STATUS|VERBOSE|DEBUG|TRACE>
Set the log level.

The **message()** command will only output messages of the specified log level or higher. The default log level is **STATUS**.

To make a log level persist between CMake runs, set **CMAKE_MESSAGE_LOG_LEVEL** as a cache variable instead. If both the command line option and the variable are given, the command line option takes precedence.

For backward compatibility reasons, **−−loglevel** is also accepted as a synonym for this option.

−−log−context
Enable the **message()** command outputting context attached to each message.

This option turns on showing context for the current CMake run only. To make showing the context persistent for all subsequent CMake runs, set **CMAKE_MESSAGE_CONTEXT_SHOW** as a cache variable instead. When this command line option is given, **CMAKE_MESSAGE_CONTEXT_SHOW** is ignored.

−−debug−trycompile
Do not delete the **try_compile()** build tree. Only useful on one **try_compile()** at a time.

Do not delete the files and directories created for **try_compile()** calls. This is useful in debugging failed try_compiles. It may however change the results of the try−compiles as old junk from a

previous try−compile may cause a different test to either pass or fail incorrectly. This option is best used for one try−compile at a time, and only when debugging.

−−debug−output
Put cmake in a debug mode.

Print extra information during the cmake run like stack traces with **message(SEND_ERROR)** calls.

−−debug−find
Put cmake find commands in a debug mode.

Print extra find call information during the cmake run to standard error. Output is designed for human consumption and not for parsing. See also the **CMAKE_FIND_DEBUG_MODE** variable for debugging a more local part of the project.

−−trace
Put cmake in trace mode.

Print a trace of all calls made and from where.

−−trace−expand
Put cmake in trace mode.

Like **−−trace**, but with variables expanded.

−−trace−format=<format>
Put cmake in trace mode and sets the trace output format.

<format> can be one of the following values.

human Prints each trace line in a human−readable format. This is the default format.

json−v1
Prints each line as a separate JSON document. Each document is separated by a newline (**\n**). It is guaranteed that no newline characters will be present inside a JSON document.

JSON trace format:

```
{
  "file": "/full/path/to/the/CMake/file.txt",
  "line": 0,
  "cmd": "add_executable",
  "args": ["foo", "bar"],
  "time": 1579512535.9687231,
  "frame": 2
}
```

The members are:

file The full path to the CMake source file where the function was called.

line The line in **file** of the function call.

defer Optional member that is present when the function call was deferred by **cmake_language(DEFER)**. If present, its value is a string containing the deferred call **<id>**.

cmd The name of the function that was called.

args A string list of all function parameters.

time Timestamp (seconds since epoch) of the function call.

frame Stack frame depth of the function that was called.

Additionally, the first JSON document outputted contains the **version** key for the current major and minor version of the

JSON trace format:

```
{
  "version": {
    "major": 1,
    "minor": 1
  }
}
```

The members are:

version
Indicates the version of the JSON format. The version has a major and minor components following semantic version conventions.

−−trace−source=<file>
Put cmake in trace mode, but output only lines of a specified file.

Multiple options are allowed.

−−trace−redirect=<file>
Put cmake in trace mode and redirect trace output to a file instead of stderr.

−−warn−uninitialized
Warn about uninitialized values.

Print a warning when an uninitialized variable is used.

−−warn−unused−vars
Does nothing. In CMake versions 3.2 and below this enabled warnings about unused variables. In CMake versions 3.3 through 3.18 the option was broken. In CMake 3.19 and above the option has been removed.

−−no−warn−unused−cli
Don't warn about command line options.

Don't find variables that are declared on the command line, but not used.

−−check−system−vars
Find problems with variable usage in system files.

Normally, unused and uninitialized variables are searched for only in **CMAKE_SOURCE_DIR** and **CMAKE_BINARY_DIR**. This flag tells CMake to warn about other files as well.

−−profiling−output=<path>
Used in conjunction with **−−profiling−format** to output to a given path.

−−profiling−format=<file>
Enable the output of profiling data of CMake script in the given format.

This can aid performance analysis of CMake scripts executed. Third party applications should be used to process the output into human readable format.

Currently supported values are: **google−trace** Outputs in Google Trace Format, which can be parsed by the *about:tracing* tab of Google Chrome or using a plugin for a tool like Trace Compass.

−−preset <preset>, −−preset=<preset>
Reads a **preset** from **<path−to−source>/CMakePresets.json** and **<path−to−source>/CMakeUserPresets.json**. The preset may specify the generator and the build directory, and a list of variables and other arguments to pass to CMake. The current working directory must contain CMake preset files. The **CMake GUI** can also recognize **CMakePresets.json** and **CMakeUserPresets.json** files. For full details on these files, see **cmake−presets(7)**.

The presets are read before all other command line options. The options specified by the preset (variables, generator, etc.) can all be overridden by manually specifying them on the command line. For example, if the preset sets a variable called **MYVAR** to **1**, but the user sets it to **2** with a **−D** argument, the value **2** is preferred.

−−list−presets, −−list−presets=<[configure | build | test | all]>
Lists the available presets. If no option is specified only configure presets will be listed. The current working directory must contain CMake preset files.

BUILD A PROJECT

CMake provides a command−line signature to build an already−generated project binary tree:

```
cmake --build <dir>             [<options>] [-- <build-tool-options>]
cmake --build --preset <preset> [<options>] [-- <build-tool-options>]
```

This abstracts a native build tool's command−line interface with the following options:

−−build <dir>
Project binary directory to be built. This is required (unless a preset is specified) and must be first.

−−preset <preset>, −−preset=<preset>
Use a build preset to specify build options. The project binary directory is inferred from the **configurePreset** key. The current working directory must contain CMake preset files. See **preset** for more details.

−−list−presets
Lists the available build presets. The current working directory must contain CMake preset files.

−−parallel [<jobs>], −j [<jobs>]
The maximum number of concurrent processes to use when building. If**<jobs>** is omitted the native build tool's default number is used.

The **CMAKE_BUILD_PARALLEL_LEVEL** environment variable, if set, specifies a default parallel level when this option is not given.

Some native build tools always build in parallel. The use of **<jobs>** value of **1** can be used to limit to a single job.

−−target <tgt>..., −t <tgt>...
Build **<tgt>** instead of the default target. Multiple targets may be given, separated by spaces.

−−config <cfg>
For multi−configuration tools, choose configuration **<cfg>**.

−−clean−first
Build target **clean** first, then build. (To clean only, use **−−target clean**.)

−−use−stderr
Ignored. Behavior is default in CMake >= 3.0.

−−verbose, −v
Enable verbose output − if supported − including the build commands to be executed.

This option can be omitted if **VERBOSE** environment variable or **CMAKE_VERBOSE_MAKEFILE** cached variable is set.

−− Pass remaining options to the native tool.

Run **cmake −−build** with no options for quick help.

INSTALL A PROJECT

CMake provides a command−line signature to install an already−generated project binary tree:

```
cmake --install <dir> [<options>]
```

This may be used after building a project to run installation without using the generated build system or the native build tool. The options are:

−−install <dir>
Project binary directory to install. This is required and must be first.

−−config <cfg>
For multi−configuration generators, choose configuration **<cfg>**.

−−component <comp>
Component−based install. Only install component **<comp>**.

−−default−directory−permissions <permissions>
Default directory install permissions. Permissions in format **<u=rwx,g=rx,o=rx>**.

−−prefix <prefix>
Override the installation prefix, **CMAKE_INSTALL_PREFIX**.

−−strip
Strip before installing.

−v, −−verbose
Enable verbose output.

This option can be omitted if **VERBOSE** environment variable is set.

Run **cmake −−install** with no options for quick help.

OPEN A PROJECT

```
cmake --open <dir>
```

Open the generated project in the associated application. This is only supported by some generators.

RUN A SCRIPT

```
cmake [{-D <var>=<value>}...] -P <cmake-script-file> [-- <unparsed-options>...]
```

Process the given cmake file as a script written in the CMake language. No configure or generate step is performed and the cache is not modified. If variables are defined using **−D**, this must be done before the **−P** argument.

Any options after **−−** are not parsed by CMake, but they are still included in the set of **CMAKE_ARGV<n>** variables passed to the script (including the **−−** itself).

RUN A COMMAND-LINE TOOL

CMake provides builtin command−line tools through the signature

```
cmake -E <command> [<options>]
```

Run **cmake −E** or **cmake −E help** for a summary of commands. Available commands are:

capabilities
Report cmake capabilities in JSON format. The output is a JSON object with the following keys:

version
A JSON object with version information. Keys are:

string The full version string as displayed by cmake **−−version**.

major The major version number in integer form.

minor The minor version number in integer form.

patch The patch level in integer form.

suffix The cmake version suffix string.

isDirty A bool that is set if the cmake build is from a dirty tree.

generators
A list available generators. Each generator is a JSON object with the following keys:

name A string containing the name of the generator.

toolsetSupport
true if the generator supports toolsets and **false** otherwise.

platformSupport
true if the generator supports platforms and **false** otherwise.

supportedPlatforms
New in version 3.21.

Optional member that may be present when the generator supports platform specification via **CMAKE_GENERATOR_PLATFORM** (**−A ...**). The value is a list of platforms known to be supported.

extraGenerators
A list of strings with all the extra generators compatible with the generator.

fileApi Optional member that is present when the **cmake−file−api(7)** is available. The value is a JSON object with one member:

requests
A JSON array containing zero or more supported file−api requests. Each request is a JSON object with members:

kind Specifies one of the supported file−api object kinds.

version
A JSON array whose elements are each a JSON object containing **major** and **minor** members specifying non−negative integer version components.

serverMode
true if cmake supports server−mode and **false** otherwise. Always false since CMake 3.20.

cat <files>...
Concatenate files and print on the standard output.

chdir <dir> <cmd> [<arg>...]
Change the current working directory and run a command.

compare_files [−−ignore−eol] <file1> <file2>
Check if **<file1>** is same as **<file2>**. If files are the same, then returns **0**, if not it returns **1**. In case of invalid arguments, it returns 2. The **−−ignore−eol** option implies line−wise comparison and ignores LF/CRLF differences.

copy <file>... <destination>
Copy files to **<destination>** (either file or directory). If multiple files are specified, the **<destination>** must be directory and it must exist. Wildcards are not supported. **copy** does follo w symlinks. That means it does not copy symlinks, but the files or directories it point to.

copy_directory <dir>... <destination>
Copy content of **<dir>...** directories to **<destination>** directory. If**<destination>** directory does not exist it will be created. **copy_directory** does follow symlinks.

copy_if_different <file>... <destination>
Copy files to **<destination>** (either file or directory) if they have changed. If multiple files are specified, the **<destination>** must be directory and it must exist. **copy_if_different** does follow symlinks.

create_symlink <old> <new>
Create a symbolic link **<new>** naming **<old>**.

NOTE:
Path to where **<new>** symbolic link will be created has to exist beforehand.

create_hardlink <old> <new>
Create a hard link **<new>** naming **<old>**.

NOTE:
Path to where **<new>** hard link will be created has to exist beforehand. **<old>** has to e xist beforehand.

echo [<string>...]
Displays arguments as text.

echo_append [<string>...]
Displays arguments as text but no new line.

env [−−unset=NAME]... [NAME=VALUE]... COMMAND [ARG]...
Run command in a modified environment.

environment
Display the current environment variables.

false Do nothing, with an exit code of 1.

make_directory <dir>...
Create **<dir>** directories. If necessary, create parent directories too. If a directory already exists it will be silently ignored.

md5sum <file>...
Create MD5 checksum of files in **md5sum** compatible format:

```
351abe79cd3800b38cdfb25d45015a15  file1.txt
052f86c15bbde68af55c7f7b340ab639  file2.txt
```

sha1sum <file>...
Create SHA1 checksum of files in **sha1sum** compatible format:

```
4bb7932a29e6f73c97bb9272f2bdc393122f86e0  file1.txt
1df4c8f318665f9a5f2ed38f55adadb7ef9f559c  file2.txt
```

sha224sum <file>...
Create SHA224 checksum of files in **sha224sum** compatible format:

```
b9b9346bc8437bbda630b0b7ddfc5ea9ca157546dbbf4c613192f930  file1.txt
6dfbe55f4d2edc5fe5c9197bca51ceaaf824e48eba0cc453088aee24  file2.txt
```

sha256sum <file>...
Create SHA256 checksum of files in **sha256sum** compatible format:

```
76713b23615d31680afeb0e9efe94d47d3d4229191198bb46d7485f9cb191acc  file1.t
15b682ead6c12dedb1baf91231e1e89cfc7974b3787c1e2e01b986bffadae0ea  file2.t
```

sha384sum <file>...
Create SHA384 checksum of files in **sha384sum** compatible format:

```
acc049fedc091a22f5f2ce39a43b9057fd93c910e9afd76a6411a28a8f2b8a12c73d7129e
668ddeb108710d271ee21c0f3acbd6a7517e2b78f9181c6a2ff3b8943af92b0195dcb7cce
```

sha512sum <file>...
Create SHA512 checksum of files in **sha512sum** compatible format:

```
2a78d7a6c5328cfb1467c63beac8ff21794213901eaadafd48e7800289afbc08e5fb3e86a
7a0b54896fe5e70cca6dd643ad6f672614b189bf26f8153061c4d219474b05dad08c4e729
```

remove [−f] <file>...
Deprecated since version 3.17.

Remove the file(s). The planned behavior was that if any of the listed files already do not exist, the command returns a non−zero exit code, but no message is logged. The **−f** option changes the behavior to return a zero exit code (i.e. success) in such situations instead. **r emove** does not follow symlinks. That means it remove only symlinks and not files it point to.

The implementation was buggy and always returned 0. It cannot be fixed without breaking backwards compatibility. Use **rm** instead.

remove_directory <dir>...
Deprecated since version 3.17.

Remove **<dir>** directories and their contents. If a directory does not exist it will be silently ignored. If**<dir>** is a symlink to a directory , just the symlink will be removed. Use**rm** instead.

rename <oldname> <newname>
Rename a file or directory (on one volume). If file with the **<newname>** name already exists, then it will be silently replaced.

rm [−rRf] <file> <dir>...
Remove the files **<file>** or directories **dir**. Use**−r** or **−R** to remo ve directories and their contents recursively. If any of the listed files/directories do not exist, the command returns a non−zero exit code, but no message is logged. The **−f** option changes the behavior to return a zero exit code (i.e. success) in such situations instead.

server Launch **cmake−server(7)** mode.

sleep <number>...
Sleep for given number of seconds.

tar [cxt][vf][zjJ] file.tar [<options>] [−−] [<pathname>...]
Create or extract a tar or zip archive. Options are:

c Create a new archive containing the specified files. If used, the **<pathname>...** argument is mandatory.

x Extract to disk from the archive. The**<pathname>...** ar gument could be used to extract only selected files or directories. When extracting selected files or directories, you must provide their exact names including the path, as printed by list (**−t**).

t List archive contents. The **<pathname>...** argument could be used to list only selected files or directories.

v Produce verbose output.

z Compress the resulting archive with gzip.

j Compress the resulting archive with bzip2.

J Compress the resulting archive with XZ.

−−zstd Compress the resulting archive with Zstandard.

−−files−from=<file>
Read file names from the given file, one per line. Blank lines are ignored. Lines may not start in − except for **−−add−file=<name>** to add files whose names start in −.

−−format=<format>
Specify the format of the archive to be created. Supported formats are: **7zip**, **gnutar**, **pax**, **paxr** (restricted pax, default), and **zip**.

−−mtime=<date>
Specify modification time recorded in tarball entries.

−− Stop interpreting options and treat all remaining arguments as file names, even if they start with −.

time <command> [<args>...]
Run command and display elapsed time.

touch <file>...
Creates **<file>** if file do not exist. If**<file>** e xists, it is changing **<file>** access and modification times.

touch_nocreate <file>...
Touch a file if it exists but do not create it. If a file does not exist it will be silently ignored.

true Do nothing, with an exit code of 0.

Windows−specific Command−Line Tools
The following **cmake −E** commands are available only on Windows:

delete_regv <key>
Delete Windows registry value.

env_vs8_wince <sdkname>
Displays a batch file which sets the environment for the provided Windows CE SDK installed in VS2005.

env_vs9_wince <sdkname>
Displays a batch file which sets the environment for the provided Windows CE SDK installed in VS2008.

write_regv <key> <value>
Write Windows registry value.

RUN THE FIND-PACKAGE TOOL

CMake provides a pkg−config like helper for Makefile−based projects:

```
cmake --find-package [<options>]
```

It searches a package using **find_package()** and prints the resulting flags to stdout. This can be used instead of pkg−config to find installed libraries in plain Makefile−based projects or in autoconf−based projects (via **share/aclocal/cmake.m4**).

NOTE:
This mode is not well−supported due to some technical limitations. It is kept for compatibility but should not be used in new projects.

VIEW HELP

To print selected pages from the CMake documentation, use

```
cmake --help[-<topic>]
```

with one of the following options:

−−help,−help,−usage,−h,−H,/?
Print usage information and exit.

Usage describes the basic command line interface and its options.

−−version,−version,/V [<f>]
Show program name/version banner and exit.

If a file is specified, the version is written into it. The help is printed to a named <f>ile if given.

−−help−full [<f>]
Print all help manuals and exit.

All manuals are printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−manual <man> [<f>]
Print one help manual and exit.

The specified manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−manual−list [<f>]
List help manuals available and exit.

The list contains all manuals for which help may be obtained by using the **−−help−manual** option followed by a manual name. The help is printed to a named <f>ile if given.

−−help−command <cmd> [<f>]
Print help for one command and exit.

The **cmake−commands(7)** manual entry for **<cmd>** is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−command−list [<f>]
List commands with help available and exit.

The list contains all commands for which help may be obtained by using the **−−help−command** option followed by a command name. The help is printed to a named <f>ile if given.

−−help−commands [<f>]
Print cmake−commands manual and exit.

The **cmake−commands(7)** manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−module <mod> [<f>]
Print help for one module and exit.

The **cmake−modules(7)** manual entry for **<mod>** is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−module−list [<f>]
List modules with help available and exit.

The list contains all modules for which help may be obtained by using the **−−help−module** option followed by a module name. The help is printed to a named <f>ile if given.

−−help−modules [<f>]
Print cmake−modules manual and exit.

The **cmake−modules(7)** manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−policy <cmp> [<f>]
Print help for one policy and exit.

The **cmake−policies(7)** manual entry for **<cmp>** is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−policy−list [<f>]
List policies with help available and exit.

The list contains all policies for which help may be obtained by using the **−−help−policy** option followed by a policy name. The help is printed to a named <f>ile if given.

−−help−policies [<f>]
Print cmake−policies manual and exit.

The **cmake−policies(7)** manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−property <prop> [<f>]
Print help for one property and exit.

The **cmake−properties(7)** manual entries for **<prop>** are printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−property−list [<f>]
List properties with help available and exit.

The list contains all properties for which help may be obtained by using the **−−help−property** option followed by a property name. The help is printed to a named <f>ile if given.

−−help−properties [<f>]
Print cmake−properties manual and exit.

The **cmake−properties(7)** manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−variable <var> [<f>]
Print help for one variable and exit.

The **cmake−variables(7)** manual entry for **<var>** is printed in a human−readable text format. The help is printed to a named <f>ile if given.

−−help−variable−list [<f>]
List variables with help available and exit.

The list contains all variables for which help may be obtained by using the **−−help−variable** option followed by a variable name. The help is printed to a named <f>ile if given.

−−help−variables [<f>]
Print cmake−variables manual and exit.

The **cmake−variables(7)** manual is printed in a human−readable text format. The help is printed to a named <f>ile if given.

To view the presets available for a project, use

```
cmake <source-dir> --list-presets
```

SEE ALSO

The following resources are available to get help using CMake:

Home Page
https://cmake.org

The primary starting point for learning about CMake.

Online Documentation and Community Resources
https://cmake.org/documentation

Links to available documentation and community resources may be found on this web page.

Discourse Forum
https://discourse.cmake.org

The Discourse Forum hosts discussion and questions about CMake.

COPYRIGHT