WIKIPEDIA

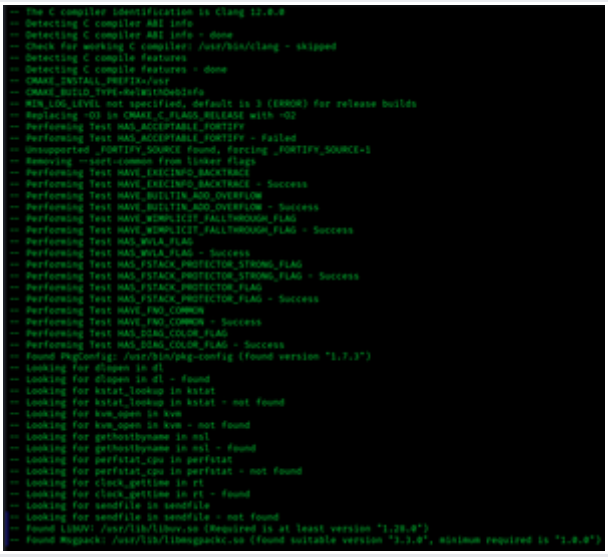
CMake

In software development, **CMake** is cross-platform free and open-source software for build automation, testing, packaging and installation of software by using a compiler-independent method.[4] CMake is not a build system but rather it generates another system's build files. It supports directory hierarchies and applications that depend on multiple libraries. It is used in conjunction with native build environments such as Make, Qt Creator, Ninja, Android Studio, Apple's Xcode, and Microsoft Visual Studio. It has minimal dependencies, requiring only a C++ compiler on its own build system.

CMake is distributed as open-source software under permissive New BSD license.[5]

Contents

CMake



Developer(s)	Andy Cedilnik, Bill Hoffman, Brad King, Ken Martin, Alexander Neundorf
Initial release	2000
Stable release	3.22.1[1] / 7 December 2021
Preview release	3.21.0-rc3[2] / 8 July 2021
Repository	gitlab.kitware.com/cmake/cmake (https://gitlab.kitware.com/cmake/cmake)
Written in	C, C++[3]
Operating system	Cross-platform
Type	Software development tools
License	New BSD
Website	cmake.org (https://cmake.org/)

History

CMake development began in 1999 in response to the need for a cross-platform build environment for the Insight Segmentation and Registration Toolkit.[6] The project is funded by the United States National Library of Medicine as part of the Visible Human Project. It was partially inspired by pcmaker, which was made by Ken Martin and other developers to support the Visualization Toolkit (VTK). At Kitware, Bill Hoffman blended components of pcmaker with his own ideas, striving to mimic the functionality of Unix configure scripts. CMake was first implemented in 2000 and further developed in 2001.

Continued development and improvements were fueled by the incorporation of CMake into developers' own systems, including the VXL Project, the CABLE[7] features added by Brad King, and GE Corporate R&D for support of DART. Additional features were created when VTK transitioned to CMake for its build environment and for supporting ParaView.

Version 3.0 was released in June 2014.[8] It has been described as the beginning of "Modern CMake".[9] Experts now advise to avoid variables in favor of *targets* and *properties*.[10] The commands `add_compile_options`, `include_directories`, `link_directories`, `link_libraries` that were at the core of CMake 2 should now be replaced by target-specific commands.

Features

A key feature is the ability to (optionally) place compiler outputs (such as object files) outside the source tree. This enables multiple builds from the same source tree and cross-compilation. It also keeps the source tree separate ensuring that removing a build directory will not remove the source files. The users aren't protected from removing the first one by an accident though.

Flexible project structure

CMake can locate system-wide and user-specified executables, files, and libraries. These locations are stored in a cache, which can then be tailored before generating the target build files. The cache can be edited with a graphical editor, which is shipped with CMake.

Complicated directory hierarchies and applications that rely on several libraries are well supported by CMake. For instance, CMake is able to accommodate a project that has multiple toolkits, or libraries that each have multiple directories. In addition, CMake can work with projects that require executables to be created before generating code to be compiled for the final application. Its open-source, extensible design allows CMake to be adapted as necessary for specific projects.[11]

IDEs configuration support

CMake can generate project files for several popular IDEs, such as Microsoft Visual Studio, Xcode, and Eclipse CDT. It can also produce build scripts for MSBuild or NMake on Windows; Unix Make on Unix-like platforms such as Linux, macOS, and Cygwin; and Ninja on both Windows and Unix-like platforms.

Build process

The build of a program or library with CMake is a two-stage process. First, standard build files are created (generated) from configuration files (CMakeLists.txt) which are written in CMake language. Then the platform's native build tools (native toolchain) are used for actual building of programs.[11][12]

The build files are configured depending on used generator (e.g. *Unix Makefiles* for make). Advanced users can also create and incorporate additional makefile generators to support their specific compiler and OS needs. Generated files are typically placed (by using `cmake`'s flag) into a folder outside of the sources one (out of source build), e.g., `build/`.

Each build project in turn contains a `CMakeCache.txt` file and `CMakeFiles` directory in every (sub-)directory of the projects (happened to be included by `add_subdirectory(...)` command earlier) helping to avoid or speed up regeneration stage once it's run over again.

Once the Makefile (or alternative) has been generated, build behavior can be fine-tuned via target properties (since version 3.1) or via `CMAKE_...`-prefixed global variables . The latter is discouraged for target-only configurations because variables are also used to configure CMake itself and to set up initial defaults.[10]

Types of build targets

Depending on CMakeLists.txt configuration the build files may be either executables, libraries (e.g. `libxyz`, `xyz.dll` etc.), object file libraries or pseudo-targets (including aliases). CMake can produce object files that can be linked against by executable binaries/libraries avoiding dynamic (run-time) linking and using static (compile-time) one instead. This enables flexibility in configuration of various optimizations.[13]

Build dependencies may be determined automatically.

Precompiled headers

It's possible to generate precompiled headers by using CMake since 3.6 version.[14]

Language

CMakeLists.txt

CMake has a relatively simple interpreted, imperative scripting language. It supports variables, string manipulation methods, arrays, function/macro declarations, and module inclusion (importing). CMake Language commands (or directives) are read by `cmake` from a file named `CMakeLists.txt`. This file specifies the source files and build parameters, which cmake will place in the project's build specification (such as a Makefile). Additionally, `.cmake`-suffixed files can contain scripts used by cmake.[15]

To generate a project's build files, one invokes `cmake` in terminal and specifies the directory which contains `CMakeLists.txt`. This file contains one or more commands in the form of `COMMAND(argument ...)`.

Command syntax

The arguments of the commands are whitespace-separated and can include keywords to separate groups of arguments. Commands can take a keywords, for instance, in the command `SET_SOURCE_FILE_PROPERTIES(source_file ... COMPILE_FLAGS compiler_option ...)` the keyword is `COMPILE_FLAGS`. It can serve as a delimiter between list of source files and some other options.[16]

Examples of commands that cmake includes to specify targets and its dependencies (to be built by the native toolchain) and which serve as starting point of the CMakeLists.txt:[17][18][19]

- `add_executable(...)` — declares an executable binary target with sources (depend on language chosen) to be built
- `add_library(...)` — the same but for a library
- `target_link_libraries(...)` — adds dependencies etc.

JSON strings

Cmake supports extracting values into variables from the JSON-data strings (since version 3.19).[20]

Internals

The executable programs CMake, CPack, and CTest are written in the C++ programming language.

Much of CMake's functionality is implemented in modules that are written in the CMake language.[21]

Since release 3.0, CMake's documentation uses reStructuredText markup. HTML pages and man pages are generated by the Sphinx documentation generator.

Modules & Tools

CMake ships with numerous `.cmake` modules and tools. These facilitate work such as finding dependencies (both built-in and external, e.g. `FindXYZ` modules), testing the toolchain environment and executables, packaging releases (`CPack` module and `cpack` command), and managing dependencies on external projects (`ExternalProject` module):[22][23]

- **ctest** — is used for target testing commands specified by CMakeLists.txt
- **ccmake** and **cmake-gui** — tweaks and updates configuration variables intended for the native build system
- **cpack** — helps to package software

CPack

CPack is a packaging system for software distributions. It is tightly integrated with CMake but can function without it. [24][25]

It can be used to generate:

- Linux RPM, deb, and gzip packages (for both binaries and source code).
- NSIS files (for Microsoft Windows).
- macOS packages.

Adopters

Open Source

Software built by using CMake includes: MySQL, Boost (C++ libraries), KDE/KDE Plasma 5 — Desktop Environment for Linux-based systems, KiCAD, FreeCAD, Webkit and Blender 3D graphics editor.[26]

Scientific tools

Software used by the ATLAS experiment is built by using CMake. The software itself is written in C/C++ and Python.[27]

Examples

Hello World

The following source code files demonstrate how to build a simple hello world program written in C++ by using CMake. The source files are placed in a `src/` directory.

```cpp
// src/Hello_world.cc
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
}
```

```
# src/CMakeLists.txt
cmake_minimum_required(VERSION 3.10)

# set the project name
project("Hello World")

# specify the executable and corresponding source file
add_executable(hello "Hello_world.cc")
```

bash script to run CMake on a Linux system. This example assumes that the script will be kept next to the `src/` folder:

```
#!/usr/bin/env bash
# Place this file next to src/ folder

cmake -S src/ -B build/     # Generate configs from the src/ folder into a build/ one
cmake --build build/        # Start building inside the build/ folder
./build/hello               # Run the compiled program.  Outputs "Hello, world!"
```

See also

- List of build automation software § Build script generation
- configure script
- Monorepo
- Qmake

References

1. "CMake 3.22.1 available for download" (https://blog.kitware.com/cmake-3-22-1-available-for-download/).
2. "CMake 3.21.0-rc3 is ready for testing" (https://blog.kitware.com/cmake-3-21-0-rc3-is-ready-for-testing/). 8 July 2021.
3. "The CMake Open Source Project on OpenHub" (https://www.openhub.net/p/cmake). OpenHub. Retrieved 2016-04-09.
4. "CMake" (https://cmake.org/).
5. "Licenses · master · CMake / CMake" (https://gitlab.kitware.com/cmake/cmake/-/tree/master/Licenses). *GitLab*. Retrieved 2020-11-13.
6. "FLOSS Weekly 111: CMake" (http://twit.tv/floss111). *podcast*. TWiT Network. Retrieved 27 February 2011.
7. "The CABLE" (https://web.archive.org/web/20130619224333/http://public.kitware.com/Cable/HTML/Index.html). Archived from the original (http://public.kitware.com/Cable/HTML/Index.html) on 2013-06-19. Retrieved 2010-11-10.
8. Maynard, Robert (June 10, 2014). "[CMake] [ANNOUNCE] CMake 3.0.0 Released" (https://cmake.org/pipermail/cmake/2014-June/057793.html).
9. "Effective Modern CMake" (https://gist.github.com/mbinna/c61dbb39bca0e4fb7d1f73b0d66a4fd1). *Gist*.
10. https://github.com/boostcon/cppnow_presentations_2017/blob/master/05-19-2017_friday/effective_cmake__daniel_pfeifer__cppnow_05-19-2017.pdf, https://gist.github.com/mbinna/c61dbb39bca0e4fb7d1f73b0d66a4fd1
11. Neundorf, Alexander (2006-06-21). "Why the KDE project switched to CMake—and how" (https://lwn.net/Articles/188693/). LWN.net.
12. "cmake-toolchains(7) — CMake 3.19.0-rc2 Documentation" (https://cmake.org/cmake/help/latest/manual/cmake-toolchains.7.html). *cmake.org*. Retrieved 2020-10-29.
13. "cmake-buildsystem(7) — CMake 3.19.0-rc3 Documentation" (https://cmake.org/cmake/help/latest/manual/cmake-buildsystem.7.html#object-libraries). *cmake.org*. Retrieved 2020-11-14.
14. "target_precompile_headers — CMake 3.21.20210925-gb818e3c Documentation" (https://cmake.org/cmake/help/git-stage/command/target_precompile_headers.html). *cmake.org*. Retrieved 2021-09-25.
15. "cmake-language(7) — CMake 3.19.0-rc2 Documentation" (https://cmake.org/cmake/help/latest/manual/cmake-language.7.html). *cmake.org*. Retrieved 2020-10-29.
16. Cedilnik, Andrej (2003-10-30). "Cross-Platform Software Development Using CMake Software" (https://www.linuxjournal.com/article/6700). Linux Journal. Retrieved 2021-01-29.
17. "add_executable — CMake 3.19.0-rc1 Documentation" (https://cmake.org/cmake/help/latest/command/add_executable.html). *cmake.org*. Retrieved 2020-10-25.
18. "add_library — CMake 3.19.0-rc1 Documentation" (https://cmake.org/cmake/help/latest/command/add_library.html). *cmake.org*. Retrieved 2020-10-25.
19. "target_link_directories — CMake 3.20.2 Documentation" (https://cmake.org/cmake/help/latest/command/target_link_directories.html). *cmake.org*. Retrieved 2021-05-10.
20. "CMake 3.19 Release Notes — CMake 3.19.7 Documentation" (https://cmake.org/cmake/help/v3.19/release/3.19.html). *cmake.org*. Retrieved 2021-03-15.
21. "cmake-language(7) — CMake 3.19.0-rc1 Documentation" (https://cmake.org/cmake/help/latest/manual/cmake-language.7.html). *cmake.org*. Retrieved 2020-10-25.
22. "cmake-modules(7) — CMake 3.14.7 Documentation" (https://cmake.org/cmake/help/v3.14/manual/cmake-modules.7.html). *cmake.org*. Retrieved 2020-10-24.
23. "ExternalProject — CMake 3.14.7 Documentation" (https://cmake.org/cmake/help/v3.14/module/ExternalProject.html). *cmake.org*. Retrieved 2020-10-24.

24. "Packaging With CPack" (https://gitlab.kitware.com/cmake/community/-/wikis/doc/cpack/Packaging-With-CPack). *CMake Community Wiki*.
25. `cpack(1) (https://www.mankier.com/1/cpack)` – Linux General Commands Manual
26. "Building Blender - Blender Developer Wiki" (https://wiki.blender.org/wiki/Building_Blender). *wiki.blender.org*. Retrieved 2021-07-01.
27. Elmsheuser, J; Krasznahorkay, A; Obreshkov, E; Undrus, A (2017). "Large Scale Software Building with CMake in ATLAS" (https://cds.cern.ch/record/2243765/files/ATL-SOFT-PROC-2017-033.pdf) (PDF). CERN.

External links

- Official website (https://cmake.org/)
- CMake (https://github.com/Kitware/CMake) on GitHub
- C++Now 2017: Daniel Pfeifer "Effective CMake" (https://www.youtube.com/watch?v=bsXLMQ6WgIk) on YouTube