CMake

About CMake

CMake is an extensible, open-source system that manages the build process in an operating system and in a compiler-independent manner. Unlike many cross-platform systems, CMake is designed to be used in conjunction with the native build environment. Simple configuration files placed in each source directory (called CMakeLists.txt files) are used to generate standard build files (e.g., makefiles on Unix and projects/workspaces in Windows MSVC) which are used in the usual way. CMake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. CMake supports in-place and out-of-place builds, and can therefore support multiple builds from a single source tree. CMake also supports static and dynamic library builds. Another nice feature of CMake is that it generates a cache file that is designed to be used with a graphical editor. For example, when CMake runs, it locates files, libraries, and executables, and may encounter optional build directives. This information is gathered into the cache, which may be changed by the user prior to the generation of the native build files.

CMake is designed to support complex directory hierarchies and applications dependent on several libraries. For example, CMake supports projects consisting of multiple toolkits (i.e., libraries), where each toolkit might contain several directories, and the application depends on the toolkits plus additional code. CMake can also handle situations where executables must be built in order to generate code that is then compiled and linked into a final application. Because CMake is open source, and has a simple, extensible design, CMake can be extended as necessary to support new features. Using CMake is simple. The build process is controlled by creating one or more CMakeLists.txt files in each directory (including subdirectories) that make up a project. Each CMakeLists.txt consists of one or more commands. Each command has the form COMMAND (args...) where COMMAND is the name of the command, and args is a white-space separated list of arguments. CMake provides many pre-defined commands, but if you need to, you can add your own commands. In addition, the advanced user can add other makefile generators for a particular compiler/OS combination. (While Unix and MSVC++ is supported currently, other developers are adding other compiler/OS support.) You may wish to study the examples page to see more details.

Origins

CMake was created in response to the need for a powerful, cross-platform build environment for the Insight Segmentation and Registration Toolkit (ITK) funded by NLM

as part of the Visible Human Project. It was influenced by an earlier system called **pcmaker** created by Ken Martin and other developers to support the Visualization Toolkit (VTK) open source 3D graphics and visualization system. To create CMake, Bill Hoffman at Kitware incorporated some key ideas from pcmaker, and added many more of his own, with the thought to adopt some of the functionality of the Unix **configure** tool. The initial CMake implementation was mid-2000, with accelerated development occurring in early 2001. Many improvements were due to the influences of other developers incorporating CMake into their own systems. For example, the VXL software community adopted CMake as their build environment, contributing many essential features. Brad King added several features in order to support the CABLE, and automated wrapping environment and GCC-XML, and GE Corporate R&D required support of their testing infrastructure (DART). Other features were added to support the transition of VTK's build environment to CMake, and to support ParaView, a parallel visualization system to support the Advanced Computing Lab at Los Alamos National Laboratory.

Learn More

You can learn CMake from the Wiki, the online Documentation, and Examples. The Mailing lists are also very helpful. The Mastering CMake book is also a good place to learn CMake.