A Short Introduction to the Pcynlitx Nova Automatic C++ Build System Generator and its Implementation Details

The automatic C++ build system generator-configurator project includes several meta-programs and subsystems.

The project web site and GitHub repositories listed below.

https://www.pcynlitx.com/

https://github.com/Erkam-Murat-Bozkurt/pcynlitx.nova

Unfortunately, I couldn't find an opportunity in order to prepare sufficient enough documentation for the fully automatic C++ build system generator. Because I have developed these software systems by myself and I did not have sufficient enough time for documentation for every part of my projects. However, I am becoming impatient in order to share this study.

I have named this system as “Pcynlitx Nova” on the project web site and this C++ automatic build system generator reads the source code of a C++ git repository and extracts C++ project dependency information. Then, the platform builds a C++ build system without requiring any configuration or coding. In this process, the software developer only performs selections about the directory locations of the source files and the options for the compiler if they exist. The C++ build systems can be constructed in two forms. The first one is CMAKE list file creation and the other one is typical GNU-Make files and operating system native scripts in a directory hierarchy.

The platform can be used from the command line or from its GUI.

In fact, automatic build system construction is performed in several steps. Although the actual construction process is more complex, I have listed some steps following on the automatic build system construction in order to give an overall look about the system.

- The user enters preferences from a GUI panel and a descriptor file (or a project file, I am not sure which is better naming) is constructed automatically.
- The preferences are read by a subsystem (I have named this sub-system as description processing and/or reading system)
- The list of files existing on the project repository and their paths are received from the git version control system.
- A meta-program determines the type of the files existing on the repository in terms of source file relation. In other words, the meta-program determines whether the files are source files or other kinds of file.
- The source files are read line by line and its header declarations are determined (In addition to this, modules will be added in the future. I have started from the headers for backward compatibility)

- From the header file declarations, the files which are parts of the git repository are extracted. In addition, the header files which are not standard headers are determined. The dependencies are converted proper data structures.
- For CMAKE construction, each source file is considered as a separate CMAKE target and CMakeList.txt files and some CMAKE configuration files with .make extension are written based on directory hierarchy. When the build system construction is performed based on GNU-Make files, the make files and scripts are written by a meta-program.
- Finally, the main CMakeLists.txt file is constructed based on the collected information about the compiler, files and directory locations ext.

In the kernel of the platform, the code runs in parallel (the computing intensive parts are multithreaded). Therefore, CMAKE configuration files and/or GNU-Make files can be written automatically in a reasonable time interval for nontrivial real C++ repositories. In my opinion, this system can be considered as an automatic build system generator or it can be considered as a translator from one build system to the others (for instance CMAKE to Basel or opposite) Because, this system can construct automatically a build system for a mature, nontrivial C++ project from bottom to top without having previous config file or code.

Have a nice day.

Thanks for everything and giving time.

Erkam Murat Bozkurt.