Instigate

Parallel Systems Development

Release Notes

Instigate Build System 1.2.8

Table of contents

1. Introduction

This document provides an overview of Instigate Build system.

Instigate Build System project which provides set of makefiles and scripts to compile, test, generate documentations and create installation packages for SW/HW projects. The documentation is available in src/doc directory. Please refer to README in the top directory for more information.

2. Available in current release

The current release supports the multiple languages, GUI toolkits, and platforms.

The library should be extended to support also compilation / simulation for Verilog HDL, VHDL.

2.1. Known issues

- **Packaging**
  - There is need to add "make package" target to be able to create the package.
- **Testing**
  - There is need to update / enhance the testing mechanism.
- **Dependencies among the projects**
  - There is need to update / enhance the mechanism which provides the dependencies among the projects(e.g. user should be able to provide the dependencies easily).
- **Review**
  - There is need to add “make review” target which will replace all the tabs in the sources with the white-spaces and checks the existence of vim settings line in all source files.
- **QT 4.5 is not supported**
  - The build environment supports only the projects build created with 4.6 QT version.

2.2. Reporting bug or issue

Please report found bugs, documentation issues to Instigate Open source team <opensource@instigate.am>

3. System Information

The project is developed under Linux environment and tested on Cygwin/Mingw/Windows. Please refer to the information provided below for the used tools.

3.1. Tools used in the test-suite

The following tools are used in the test-suite:

- GNU bash
- GNU make
- GCC g++
- Doxygen
- Pkg-config
- Patchelf

3.2. Cygwin / Mingw / Windows

The makefiles has been verified on Cygwin (CYGWIN_NT-5.1) and Mingw running under Windows XP (may work also for the other Windows distributions).

3.3. Installation

The project will be installed by execution of the "make install" command. By default the project will be installed with the following path:

- /opt/instigate/os3/

There is need to change the install_path parameter value in top-level makefile to be able to change the default path for installation.

install_path := /opt/instigate/os3/$(os3_version)/

4. Usage

The structure of Instigate Makefiles contains two types of makefiles. The first is the main makefile which is placed under root directory of the application. Near the makefile should be created "src" folder with the projects. The second type of the makefiles should be placed under each of the projects. Each of makefiles contains specific variables and the project is building based on this parameters. The variables specified in the main makefile can be modified in the local makefiles.

In order to use the Instigate Build System, one needs to include the main.mk in main makefile.

The list of variables specified in the main makefile follows:

- project_name - the name of the project
- project_version - the version of the project
- os3_path - the path of Instigate OS3
- os3_version - the version of Instigate OS3
- project_root - the absolute path to the directory where the current project is located
- projects - the list of the projects which are should be built

One needs to specify preconditions variable for checking them before building the project. See the example below:

```
preconditions := gcc doxygen ar ln
library_preconditions := gdkmm-2.4
```

- test_type variable specifies the type of tests to be run. The possible values are listed below:
  - regression_tests - includes the tests to check the basic principles of the application. Should be called after each modification to check the regression of application.
  - continuous_tests - tests be run after each the modification.
  - nightly_tests - tests to be run after working day
  - weekly_tests - tests to be run after working week
- install_path - defined the path where the package will be installed. See Installation section.

Depending on the build type (debug or release), dbg_package and opt_package variables are used respectively to specify which directories need be installed.
See the examples below :

- export dbg_package := src doc lib bin inc
- export opt_package := lib bin doc inc

See example below for defining dependencies between the projects:

src/project1 : src/project2 src/project3 src/project4

The list of variables specified in the projects makefiles follows:

- cpp_files - the source files under the project.
- public_headers - the header files under the project, which are supposed to be used by other projects.
- qt_headers - header files of Qt project.
- moc_flags - flags to apply during the creation of moc files.
- qt_ui_files - Qt ui files
- uic_flags - cflags of Qt ui compiling
- qrc_files – Qt resources files
- compiler_flags - flags for compiler (can be updated/overwritten for the project).
- linker_flags - linker flags for compiler (can be updated/overwritten for the project).
- libs - the dependency libraries.
- lib_version - version of library, if not specified the project version is used

The exe.mk should be included to generate the executable file.

Two types of libraries can be generated for each project - dynamic (shared objects) and static (archives).

The dynamic.mk should be included to build shared library.

The static.mk should be included to build the static library.

If you want to build for specific target (i386 or x86_64) then include lib32.mk or lib64.mk. Including lib32_64.mk will compile for both targets. lib.mk will compile static or shared libraries for the current platform.

