Ceylan's HOWTO

HOW-TO

Contact: about (dash) howtos (at) esperide (dot) com

Creation date: Wednesday, November 17, 2021

Lastly updated: Sunday, November 21, 2021

Version: 0.0.1

Status: In progress

Dedication: Users of these HOWTOs

Abstract: The role of these `HOW-TOs` is, akin to a cookbook, to share a collection of (technical) recipes ("how-to do this task?) regarding various topics.

These elements are part of the Ceylan umbrella project.

The latest version of this documentation is to be found at the official Ceylan-HOWTOs website (`http://howtos.esperide.org`).

Table of Contents

Build Tools

Organisation: Copyright (C) 2021-2021 Olivier Boudeville

Contact: about (dash) howtos (at) esperide (dot) com

Creation date: Saturday, November 20, 2021

Lastly updated: Sunday, November 21, 2021

Table of Contents

Purpose of Build Tools

A build tool allows to automate all kinds of tasks, by **applying rules and tracking dependencies**: not only compiling, linking, etc. applications, but also checking them, generating their documentation, running and debugging them, etc.

Choice

Often build tools are tied to some programming languages (ex: Maven for Java, Rebar3 for Erlang, etc.).

Some tools are more generic by nature, like late GNU autotools, or Cmake, GNU make, etc.

For most uses, our personal preference goes to the latter. Notably all our Erlang-based developments, starting from Ceylan-Myriad, are based on GNU make.

GNU make

We recommend the reading of this essential source for reference purpose, notably the section about The Two Flavors of Variables.

Taking our Erlang developments as an example, their base, first layer, Ceylan-Myriad, relies on build facilities that are designed to be also reused and further adapted / specialised / parametrised in turn by all layers above in the stack (ex: Ceylan-WOOPER).

For that, Myriad defines three top-level makefiles:

- base build-related *variables* (settings) in GNUmakevars.inc, providing defaults that can be overridden by upper layers
- *automatic rules*, in GNUmakerules-automatic.inc, able to operate generically on patterns, typically based on file extensions
- *explicit rules*, in GNUmakerules-explicit.inc, for all specific named make targets (ex: `all`, `clean`)

Each layer references its specialisation of these three elements (and the ones of all layers below) in its own GNUmakesettings.inc file, which is the only element that each per-directory `GNUmakefile` file will have to include.

Such a system allows defining (build-time and runtime) settings and rules once for all, while remaining flexible and enabling individual makefiles to be

minimalistic: beside said include, they just have to list which of their subdirectories the build should traverse (thanks to the `MODULES_DIRS` variable, see example).

See Also

One may refer to the development section of Ceylan-Hull, or go back to the Ceylan-HOWTOs main page.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, use the contact address at the top of this document).

Ending Word

Hoping that these Ceylan-HOWTOs may be of help!