Table of Contents

Generic Makefile for CMT projects.....1
- History.....1
- Features.....1
- How to get it.....1
- Usage.....1
  - all (default).....2
  - <package>.....2
  - clean.....2
  - purge.....3
  - tests (LHCb-specific).....3
  - docs (LHCb-specific).....3

Generic Makefile for CMT projects

History

CMT is a tool that moves away from the de-facto standards in the Open Source and Linux communities, making the usage of standard tools (like eclipse) difficult to use. In some cases the tools are configurable enough to allow, with some efforts, to enable them to build our software.

Instead of configuring different tools to understand CMT, we can take another route: add a layer of wrapping around CMT that adheres to the santards mentioned above, for example by a `Makefile` that embed calls to the `cmt` program.

I started to write such a Makefile avoiding as much as possible LHCb-specific conventions to have a single file that could be used in any CMT project to simply call "make" "make clean" and have the project built in the correct way (you should know that the safest way to build a CMT project iis a call to `cmt broadcast cmt make` from the "main" package of the project). Along the way I discovered that the Makefile I was writing allowed to build a project using several processes in a way that has never been possible even combining `tbroadcast` and `cmt make -j`.

After some time, the generic Makefile was included in the project LbScripts (in LbUtils) to simplify its deployment. Another Makefile was then added to the package LbConfiguration, this one using the one in LbUtils and extending it with LHCb-specific features.

By adding a few POSIX tools it has been possible to adapt the Makefile to work on Windows too.

It is now used routinely in the nightly builds of LHCb.

Features

You may ask why at all you should use this Makefile if you know already `cmt broadcast cmt make`?

- it is simple to use
- it can parallelize the build at both the package and the sources levels
- it works for both local projects and complete ones
- it works in the same way on Linux, MacOSX and Windows
- it can run the tests and can build the doxygen documentation in local projects

How to get it

You may not have noticed, but you already have it. It is automatically added to each new local project created with `SetupProject --build-env` (or the shortcut `setenvProject`) and to each project checked out with `getpack --project`.

Actually, the Makefile created points simply to the one in the version of LbScripts you are using, so that you will automatically pick up the new features whenever a new release of LbScripts is put in production.

Usage

The Makefile contains a list of targets to perform the actions with something like

```
make [target]
```

If no target is specified, the default behavior is to build the target `all`.

In the following sections the targets are described.

all (default)

Default target used to build all the packages in the current directory taking into account their dependencies. All the groups defined in the packages will be built.

It is, in principle, equivalent to a call to

```
cmt broadcast cmt make all_groups
```

from the `cmt` directory of the container package of a project (<Project>Sys), but doesn't need that you call `cmt config` beforehand and that you check out the container package.

If a package fails to build, by default, the build is stopped, but the behavior can be changed using the option `Package_failure_policy` on the command line. The possible values are:

- **stop**: (default) stop as soon as possible after the error
- **skip**: do not build the other groups of the problematic package, but go on with the other packages
- **ignore**: ignore the errors and try to go on with the other groups and packages (useful in combination with the switch `-k`)

For example, to try to build as much as possible use:

```
make -k Package_failure_policy=ignore
```

The `--jobs` option (AKA `-j`) of make can be used very efficiently. If possible, more than one package will be built at the same time, and within the same package more than one source file will be compiled, never exceeding (in total) the maximum number of jobs specified as argument of `-j`. The option `-j` can be effectively used in combination with the option `-l` (see `info make` for details).

<package>

Specifying the name of one of the local packages as target, that package and all the packages it depends on are built.

It is possible to trigger the build only of one package by adding the special option `PKG_ONLY=1` on the command line of make, e.g.

```
make PKG_ONLY=1 Tools/CondDBUI
```

clean

Remove all the files generated by a build step and the directory `InstallArea`.

It is equivalent to

```
cmt broadcast cmt make clean
rm -rf InstallArea
```

The advantage is the is can be run in parallel, cleaning all the packages at the same time.

purge

A bit more aggressive than `clean`, it removes also the cached dependencies between the packages and the Makefiles created by CMT int the `cmt` directories, thus forcing a new call to `cmt config` during the next build.

It is mandatory to call it when a package is removed from the current project.

tests (LHCb-specific)

Depends on the target `all` and run the tests of the project.

It requires that the container package of the project has been checked out.

With this target, building and running the tests of project is as easy as

```
getpack -Pr Gaudi v21r6
cd GAUDI/GAUDI_v21r6
make tests -j 4
```

docs (LHCb-specific)

Builds the doxygen documentation of the project (works with local projects too).

It requires the container package to be present in the working directory.

It simplifies the testing and fixing of the doxygen comment of single packages.

This target is available since LbScripts v4r4.

-- MarcoClemencic - 20-Nov-2009

This topic: LHCb > LbScriptMakefile
Topic revision: r2 - 2009-11-20 - MarcoClemencic