ckon Documentation

Release 0.3

Patrick Huck

September 23, 2015

Contents

CHAPTER 1

Introduction

ckon is a C++ program/tool which automatically takes care of compilation, dictionary generation and linking of programs and libraries developed for data analyses within the CERN ROOT analysis framework. This includes parsing include headers to figure out which libraries the main programs need to be linked to. It uses automake/autoconf to be platform independent and GNU install compliant. In addition, m4 macros are automatically downloaded and the according compiler flags included based on a list of boost libraries provided in the config file. For the purpose of YAML database usage, a m4 macro can be downloaded during setup to link against the yaml-cpp library.

Authors and Contributors

Patrick Huck (@tschaume)
invaluable contributions: Hiroshi Masui
Reference Talk (LBNL, 2011/11/14)

License & Project Homepage

ckon is published under MIT License.
Find the project page at http://tschaume.github.com/ckon

Software Requirements

- m4/1.4.6
- autoconf/2.68
- automake/1.11.4
- libtool/2.4
- boost/1.50
- libcurl/7.27.0

CHAPTER 2

Installation

- clone *ckon* via `git clone git@github.com:tschaume/ckon.git`
- install via `cd ckon; ./installCkon <install-path>`
  - replace `<install-path>` with an install path in your $PATH
  - see `./installCkon -h` for help
- see `./configure --help` for configure options in case something goes wrong

CHAPTER 3

Usage

3.1 Generic Options

Shown below are the generic command line options which can be given to *ckon*.:

```
Generic Options:
 -h [ --help ]          show this help
 -v [ --verbose ]       verbose output
 -j arg                 call make w/ -j <#cores>
 --ckon_cmd arg         setup | clean | install
```

The long option `--ckon_cmd` is implemented as optional positional option to run the setup, clean all compilation products (i.e. `make clean`) and globally install libraries and programs (i.e. `make install`):

- `ckon setup`: run the setup
- `ckon`: compile
- `ckon clean`: make clean
- `ckon install`: make install
- `ckon dry`: only generates Makefiles, no compilation

3.2 Setup

`ckon setup` generates the files *configure.ac* and *.autom4te.cfg* (both autoconf specific, no need for modifications) as well as *ckon.cfg*. Modify the latter to resemble your directory structure and linker options. Simply remove the lines/options you don't need, thus using the default options.

3.3 Configuration

The following options can be set on the command line or preferably in *ckon.cfg*. Optionally, a file named `ckonignore` with a list of strings to be ignored during the build process, can be created in the working directory. Wildcards are not supported (yet). Instead each path currently processed by *ckon* will be checked against the strings/lines in `ckonignore`. If one of the strings in `ckonignore` is contained in the path, the path is ignored/skipped.:

```
Configuration:
  -s [ --suffix ] arg    add suffix + in LinkDef.h (bool)
  -y [ --yaml ] arg      use yaml
  --ckon.src_dir arg     source dir
  --ckon.exclSuffix arg  no + suffix
  --ckon.NoRootCint arg  no dictionary
  --ckon.prog_subdir arg progs subdir
  --ckon.build_dir arg   build dir
  --ckon.install_dir arg install dir
  --ckon.cppflags arg    add CPPFLAGS
  --ckon.boost arg       boost libraries

In addition, unregistered options of the form
ldadd.prog_name are allowed to use for adding
LDFLAGS to the linker of specific programs. The
given string/value is added verbatim in LDADD.
Unregistered options are only allowed in ckon.cfg
```

The unregistered option group `ldadd` is allowed. For instance, link the programs *genCharmContrib* and *dedxCut* versus Pythia6 and RooFit, respectively, by adding the following to *ckon.cfg*.:

```
[ldadd]
genCharmContrib=-lPhysics -lEG -lEGPythia6  # link pythia
dedxCut=-lRooFit -lRooFitCore -lMinuit      # link roofit
```

`ckon.boost` is set during `ckon setup` to use and link against specific boost libraries. Try not to run rootcint (`ckon.NoRootCint`) on the library if compilation fails.

Note: ckon version 0.4 now allows for the automatic download of a `yaml.m4` macro during `ckon setup` to link against the yaml-cpp library. Please submit an issue if the macro doesn't find the library after you installed it. This added functionality shouldn't break anything if you choose not to use YAML during `ckon setup`.

Warning: For the recursive header scan to work, make sure that all include directives for C++ and ROOT headers are enclosed in `<...>`! Only your local/private headers should be enclosed in `"..."`. Otherwise *ckon* will fail reporting a `basic_string::_S_create` error.

3.4 Typical Directory Structure

Put header and source files for each library into a separate folder in `ckon.src_dir`. Running *ckon* should automagically take the right action for the current status of your build directory (no need to run `ckon clean` before re-compilation). Makefiles and LinkDef's are generated automatically based on the contents and timestamps in the `ckon.src_dir` directory.

A typical directory structure could look as follows - using the current defaults for illustration purposes.:

```
StRoot/
    ElectronPid/
        BetaPanels.cxx
        BetaPanels.h
        PureSampleAnalysis.cxx
        PureSampleAnalysis.h
        SigmaElFitsMaker.cxx
        SigmaElFitsMaker.h
        SigmaElFitsPlotter.cxx
        SigmaElFitsPlotter.h
```

```
        SigmaElFitsUtils.cxx
        SigmaElFitsUtils.h
        programs/
            README
            beta3sig.cc
            dedxCut.cc
            nsigparamsGP.cc
            pureSamp.cc
    StBadRdosDb/
        StBadRdosDb.cxx
        StBadRdosDb.h
        database/
            dbfiles
            genAll.sh
            genBadRdosDb.pl
        macros/
            testStBadRdosDb.C
    YamlCfgReader/
        YamlCfgReader.cxx
        YamlCfgReader.h
        config.yml
...
```

CHAPTER 4

Index and Search

- genindex
- modindex
- search