Under heavy development. Please mail mcr38@cornell.edu with any questions or concerns.

1 Introduction

This build system is based on the tool "cmake" version 2.8.8.
All syntax in build description files mentioned below conforms to the rules for cmake CMakeList.txt files as found in the cmake documentation.

2 Prerequisites

One must have their bash shell environment set up via the ACC environment setup procedure described here: https://wiki.lepp.cornell.edu/lepp/bin/view/ACC/ACL/DevelopmentEnvironmentSetup

3 Overview

The build system will compile C, C++, and Fortran (77,90,95,2003) source code into object code and link into libraries and/or executables (programs).

To control how a project's source code is built one composes or edits an existing file named CMakeLists.txt. This file must appear in a project directory in order for the build system to recognize that directory as project that can be built.

The system operates within some directory organization requirements. The base directory wherein one or more project directories are located will be referred to as `<DIR>`.
The project directory in question will be referred to as `<PROJ_DIR>` = `<DIR>/<PROJECT>`.

- Within `<PROJ_DIR>` there may be one or more subdirectories containing one or more source code files in any of the languages mentioned above.
- Within `<PROJ_DIR>` there may be one or more subdirectories containing header files necessary for compilation.

4 Building Code

4.1 General Options

Some options are available for controlling the build process.
Prebuild action: This allows for the user to specify a command to run immediately prior to each build.

The prebuild action target allows for the user of the build system to request that a particular action is taken prior to building their project. This allows one to run a preprocessing or code generation tool whose product is required for the build process. The user must specify the path of a single command to run as a string in the variable PREBUILD_ACTION. If this variable is present, the build system will run the specified command in a shell prior to starting the build process. The command cannot have any arguments but it may generate console output along with doing other tasks. The output it generates will show up as part of the output produced during the build process.

Example:

```
set (PREBUILD_ACTION ./generate_some_code.sh)
```

The mpmnet project uses this mechanism and can be consulted as an example.

4.2 Libraries

Provided a CmakeLists.txt build description file exists in the project directory that you wish to build, invoking the command `mk` will build a "production" binary which is suitable for everyday running. The command `mkd` will build a "debug" binary that contains debugging symbols for use in an interactive debuuger.

The object and other support files are kept in a separate build tree automatically created within the project directory called production or debug, depending on the build type requested.

`mk clean` will clean the object files and binaries of the production build type and `mkd clean` will do the same for the debug build type.

To control the building of a library, create or copy a file named `CmakeLists.txt` into `<PROJ_DIR>`. An example `CmakeLists.txt` file is shown and annotated here.
The gray text is boilerplate and must appear verbatim. The remaining sections are meant to be modified by the user to configure the details of the build. This example is used to build the `libcesrv.a` static library file and to allow for building the `cesrv_cl` program

```
set(LIBNAME cesrv)
cmake_minimum_required(VERSION $ENV{ACC_CMAKE_VERSION})

set(INC_DIRS
  ../include
  ../CesrBPM/include
  include
)

set(SRC_DIRS
  code
)

set(EXE_SPECS cesrv.cmake)

include($ENV{ACC_BUILD_SYSTEM}/Master.cmake)
```

INC_DIRS holds the list of directories in <PROJ_DIR> to search for include files.
SRC_DIRS holds the list of directories in <PROJ_DIR> that hold C, C++, and/or Fortran source code files to compile.
DEPS holds the list of in-house libraries that the software being built depends upon. This ensures that the necessary code in the project gets rebuilt in the event a local dependency library is modified.
EXE_SPECS holds the list of build specifications needed for producing one or more programs. See below for details.

4.3 Programs

To control the building of an executable, create or copy a file named after the program to produce into <PROJ_DIR>. In this example, the supplementary build description file is called cesrv.cmake which will be used to produce the executable file cesrv_cl.

```
set (EXENAME cesrv_cl)
set (SRC_FILES
  program/cesrv_cl.f90
)

set (LINK_LIBS
  cesrv
  mpm_utils
  nonlin_bpm
  cesr_utils
  bmad
  sim_utils
  cbpmfio
  CesrBPM
  mpmnet
  recipes_f-90_LEPP
  cbi_net
  pgplot
  xsif
  forest
  curses
)
```

EXENAME is the name of the program file to be built.
SRC_FILES is a list of file paths (relative to <PROJ_DIR>) of the source files containing the program's main routine and other functionality that is not to be included in a library.
LINK_LIBS is the list of additional libraries, if any, that provide functionality to the program being built.

To actually create the executable upon invocation of the mk command, a session variable can be set to in the user's environment called ACC_BUILD_EXES. Several affirmative values of this variable will be honored as per the cmake documentation. A short list of values that will result in the intended behavior is "ON", "TRUE", "Y", "YES", "1". This allows the creation of all executables for which <executable>.cmake files have been created and for which references are present in the SET (EXE_SPECS <spec1> <spec2>...) line.

Alternatively, if the ACC_BUILD_EXES variable is not set affirmatively, individual executables can be created by using the name of their target after the mk[mkd] command. Program target names have an "-exe" appended to them.
I.e. mk <program_nam

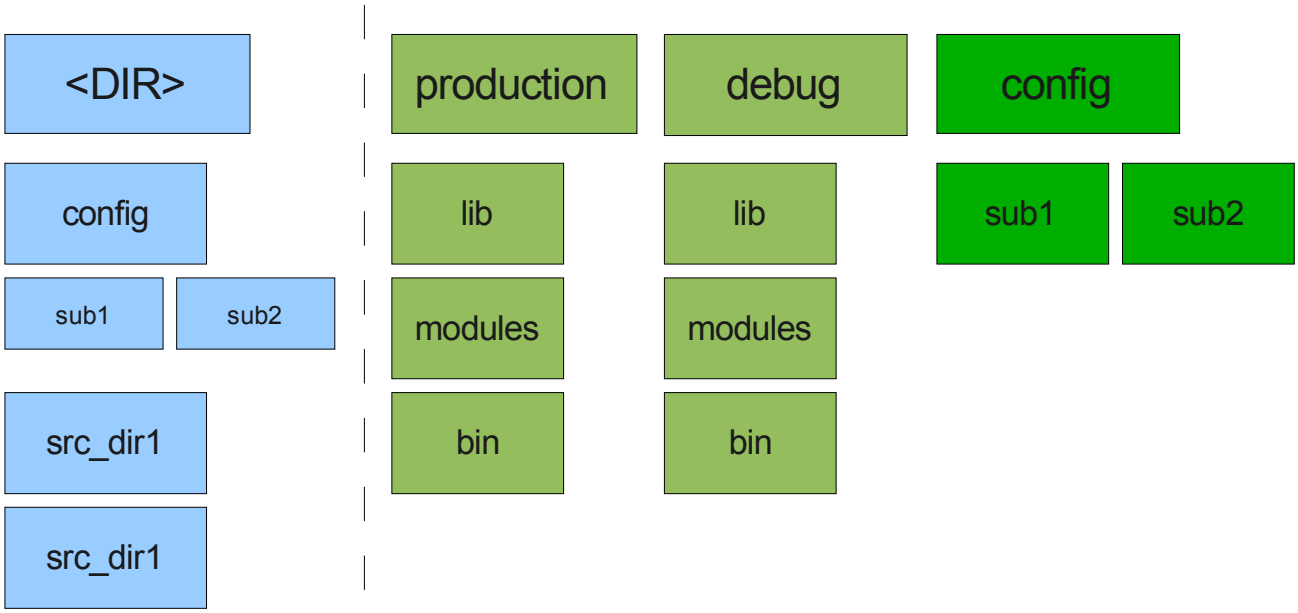
4.4 Library Availability

Short linker-style library names may be specified in the LINK_LIBS assignment of executable build descriptions. Libraries with various origins may be specified in this way. Libraries in bold are always available and need not be specified by the user in their LINK_LIBS list.

Library Origin	Libraries	Predefined Search Locations
System Libraries	**`X11, pthread, stdc++, readline`**	`/lib64, /usr/lib64`
3rd-party Packages (Maintained by Accelerator Group)	`Pgplot, forest, gsl, lapack, xsif, ...`	`${ACC_RELEASE_DIR}/packages/lib` `${ACC_RELEASE_DIR}/packages/root/lib`
Accelerator Physics Libraries (Actively developed by Accelerator Group)	`recipes_f-90_LEPP, mpmnet, cbi_net, bmad, ...`	`../production[debug]/lib` `${ACC_RELEASE_DIR}/lib`

4.5 Build Products

The build process operates on the project directory (blue, left of dotted line) and produces one or more output directories with varying internal structure (green, right of dotted line).



Library/Module and Include File Search Orders:

1. Local build product directories (`../production[debug]/lib`, `../production[debug]/modules`)
2. Local build project include directories (i.e. `../<project>/include`)
3. Pre-built libraries release tree `$ACC_RELEASE_DIR/lib` and release project-specific include directories
4. 3rd-party packages directories `$ACC_RELEASE_DIR/packages/lib` and packages-specific include directories
5. System (OS) directories

Generally, searches start local and proceed to more and more distant/lower level locations. This allows for locally built version of various libraries to be used preferentially over pre-built libraries if necessary.

5. Summary of Commands and Environment Variables That Control Build Behavior

Command	Purpose
`mk`	Build "production" binaries/executables
`mkd`	Build "debug" binaries/executables - (-g options employed, debugging symbols)
`mk[mkd] <program>-exe`	Build a single program that has a <program>.cmake reference in the CmakeLists.txt file. Will also build the local library, if one is defined and the program depends upon it.

Environment Variable	Value(s)	Purpose
`ACC_BUILD_EXES`	`Y, YES, T, TRUE, 1`	If exported in environment, `mk[mkd]` will build all programs specified in `EXE_SPECS` list.
`ACC_ENABLE_SHARED`	`Y, YES, T, TRUE, 1`	If exported in environment **-AND-** if `set (CREATE_SHARED true)` is present in `CmakeLists.txt` , a shared-object (.so) version of the library will be created.