File Map Document for Software Sassena v1.4.1

High Level Organization

The main directory contains a few subdirectories which provide the structural framework for code maintenance and compilation. It usually contains:

build.env
builds
cmake
CMakeLists.txt
include
src
tests
tools
vendor

build.env contains sample bash scripts which can be used to initialize the cmake build environment. They are named by the machine name for which they are written. These files are highly system dependent and are not necessarily accurate, since systems continuously get upgraded. However, they can be used to bootstrap the compilation environment more efficiently.

build is the default name for the out of source builds you may perform. The convention used here is that the most recent version is build within the “head” subdirectory. Other builds may refer to a specific tag/version, e.g. v1.2.1.

cmake contains custom module files for the compilation in the subdirectory “modules” and the compilation logic structured through separate cmake include files, as will be discussed in the cmake section.

CMakeLists.txt is the entry point for the cmake compilation kit. When cmake is executing with an absolute or relative path it will search for this file for initializing the compilation environment.

include contains the interface and header section for all the source code files of the project.

src contains all the implementation code for the software.

tests is the directory where single file execution units should reside. In version v1.2.1 the tests are mostly broken and extensive work is needed to revise this section and make it functional. Once operational unit tests should provide an alternative way of testing the software through testing separate components. The build instructions for building the tests have yet to be incorporated into the cmake instruction set.

tools contains source files for small binaries for single purpose tasks (e.g. normalization). However, as in the case of *tests*, the directory is not well maintained.

vendor contains code which is provided from other sources and are distributed together with sassena for convenience.

cmake directory

This directory contains the instruction set for cmake. The main entry point for cmake (CMakeLists.txt) resides in the root directory and refers to files within the cmake subdirectory for further instructions. Currently the CMakeLists.txt file checks for the definition of a static flag (-D STATIC=1) and then either includes *CMakeLists.txt.static* or *CMakeLists.txt.shared*. The files are:

CMakeLists.txt.depends contains instructions for including external library dependencies into the compilation environment.

CMakeLists.txt.intern contains instructions for building the sub-libraries of the sassena software.

CMakeLists.txt.executables contains build target instructions and performs the linking of any binaries with the sassena sub-libraries.

CMakeLists.txt.packages is included but not used at the moment. It eventually will provide a mechanism for cross-compiling software packages.

CMakeLists.txt.version contains instructions for incorporating version information into the software through automatically generated C++ header files (SassenaConfig.hpp).

include and src directory

In general, files in the include directory are usually purely used to define the C++ class interface and files in the src directory are used to implement it. The software is strongly compartmentalized, which means that the sub-directories represent certain functional levels of the software, which can either be independent or rely on each other. For example, the LOG module provides various singleton classes used for console output which are used by various other modules to communicate information to the user. The CONTROLIO module for example includes the LOG and the MATH module, which themselves are only dependent on external libraries. The dependency graph can be inspected by studying the TARGET_LINK_LIBRARIES in the file *CMakeLists.intern*.

The following modules exist and use the specified subdirectories to implement the C++ classes:

Module Name	Subdirectories
MATH	math
LOG	log
CONTROLIO	control, io
SAMPLE	sample
MPI	mpi
REPORT	report
DECOMPOSITION	decomposition
SERVICES	services
STAGER	stager
SCATTER_DEVICES	scatter_devices

The main directory in the src folder is dedicated to hold the single file entry points for any binary executables. They should only take charge of initializing the software environment and executing the various sections of the modules. The entry point for the software sassena is *sassena.cpp* in the src/main folder.

Additionally every module includes the single file set incude/common.hpp and src/common.cpp, which provides a mechanism to set type information across all modules.

MATH

The MATH module provides functions to implement mathematical routines (element wise squaring of an array, autocorrelation, reduction), type classes for coordinate systems, and other trigonometric operations.

LOG

The LOG module contains singleton classes which are used to provide the user with formatted Information, Warnings and Errors.

CONTROLIO

The CONTROLIO module consists of two sub modules.
The CONTROL part contains classes which define singleton classes used to initialize the software wide database and parameters. This module is used to map the input coming from the files db.xml and scatter.xml (default names) and initialize anything else with default values and behavior. The control flow of the software uses these classes extensively, which is why CONTROLIO is frequently included by other modules.

The IO part contains classes used to interface file formats, such as XDR data files and XML.

SAMPLE

The SAMPLE module contains classes which initialize the trajectory data and structural information.

MPI

The MPI module contains wrapper functions to facilitate the broadcasts of data structures.

REPORT

The REPORT module contains analysis tools which operate on the timer information aggregated during the software execution and presents this information at the end of the calculation to the user.

DECOMPOSITION

The DECOMPOSITION module contains the partitioning logic. The software will use it to make an intelligent decision on which partition size to use.

SERVICES

The SERVICES module contains additional internal applications which implement asynchronous behavior. Two are the monitoring service, which is used to query for the progress of a calculation and present the information to the user, and the file writer service, which is used to send results asynchronously to the main node.

STAGER

The STAGER module implements the routines for staging the trajectory data. It uses the SAMPLE module to initialize the trajectory and then places the data intelligently on the available partitions.

SCATTER_DEVICES

The SCATTER_DEVICES module contains a hierarchy of classes which implement the runtime logic for computing the scattering functions.