gsutils

Generated by Doxygen 1.9.1

Chapter 1

Main Page

1.0.1 GreenSocs Basic SystemC utility library

This is the GreenSocs basic utilities library. It contains utility functions for CCI and simple logging functions.

The GreenSocs CCI libraries allows two options for setting configuration parameters

`--gs_luafile <FILE.lua>` this option will read the lua file to set parameters.

`--param path.to.param=<value>` this option will allow individual parameters to be set.

NOTE, order is important, the last option on the command line to set a parameter will take preference.

This library includes a Configurable Broker (gs::ConfigurableBroker) which provides additional functionality. Each broker can be configured separately, and has a parameter itself for the configuration file to read. This is `lua_file`. Hence

```
--param path.to.module.lua_file="\"/host/path/to/lua/file""
```

Note that a string parameter must be quoted.

The lua file read by the ConfigurableBroker has relative paths - this means that in the example above the `path.to.module` portion of the absolute path should not appear in the (local) configuration file. (Hence changes in the hierarchy will not need changes to the configuration file).

1.1 GreenSocs Build and make system

1.2 How to build

This project may be built using cmake

```
cmake -B BUILD;cd BUILD; make -j
```

cmake version 3.14 or newer is required. This can be downloaded and used as follows

```
curl -L
 https://github.com/Kitware/CMake/releases/download/v3.20.0-rc4/cmake-3.20.0-rc4-linux-x86_64.tar.gz
 | tar -zxf -
./cmake-3.20.0-rc4-linux-x86_64/bin/cmake
```

1.2.1 details

This project uses CPM https://github.com/cpm-cmake/CPM.cmake in order to find, and/or download missing components. In order to find locally installed SystemC, you may use the standards SystemC environment variables: SYSTEMC_HOME and CCI_HOME. CPM will use the standard CMAKE find_package mechanism to find installed packages https://cmake.org/cmake/help/latest/command/find_package.html To specify a specific package location use <package>_ROOT CPM will also search along the CMAKE_MODULE_PATH

Sometimes it is convenient to have your own sources used, in this case, use the CPM_<package>_SOURCE_DIR. Hence you may wish to use your own copy of SystemC

NB, CMake holds a cache of compiled modules in ~/.cmake/ Sometimes this can confuse builds. If you seem to be picking up the wrong version of a module, then it may be in this cache. It is perfectly safe to delete it.

1.2.1.1 Common CMake options

CMAKE_INSTALL_PREFIX : Install directory for the package and binaries. CMAKE_BUILD_TYPE : DEBUG or RELEASE

The library assumes the use of C++14, and is compatible with SystemC versions from SystemC 2.3.1a.

1.2.2 More documentation

More documentation, including doxygen generated API documentation can be found in the /docs directory.

1.2.3 Using yaml for configuration

If you would prefer to use yaml as a configuration language, lyaml provides a link. This can be downloaded from https://github.com/gvvaughan/lyaml

The following lua code will load "conf.yaml".

```
local lyaml   = require "lyaml"
function readAll(file)
    local f = assert(io.open(file, "rb"))
    local content = f:read("*all")
    f:close()
    return content
end
print "Loading conf.yaml"
yamldata=readAll("conf.yaml")
ytab=lyaml.load(yamldata)
for k,v in pairs(ytab) do
    _G[k]=v
end
yamldata=nil
ytab=nil
```

1.2.4 Using gs::ConfigurableBroker

The broker will self register in the SystemC CCI hierarchy. All brokers have a parameter `lua_file` which will be read and used to configure parameters held within the broker. This file is read at the *local* level, and paths are *relative* to the location where the ConfigurableBroker is instanced.

These brokers can be used as global brokers.

The `gs::ConfigurableBroker` can be instanced in 3 ways:

1. `ConfigurableBroker()` This will instance a 'Private broker' and will hide **ALL** parameters held within this broker.

   A local `lua_file` can be read and will set parameters in the private broker. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`).

2. `ConfigurableBroker({{"key1","value1"},{"key2","value2")...})` This will instance a broker that sets and hides the listed keys. All other keys are passed through (exported). Hence the broker is 'invisible' for parameters that are not listed. This is specifically useful for structural parameters.

   It is also possible to instance a 'pass through' broker using `ConfigurationBroker({{}})`. This is useful to provide a *local* configuration broker than can, for instance, read a local configuration file.

   A local `lua_file` can be read and will set parameters in the private broker (exported or not). This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`). The `lua_file` will be read **AFTER** the construction key-value list and hence can be used to over-right default values in the code.

3. `ConfigurableBroker(argc, argv)` This will instance a broker that is typically a global broker. The argc/argv values should come from the command line. The command line will be parsed to find:

   `> -p, --param path.to.param=<value>` this option will allow individual parameters to be set.

   `> -l, --gs_luafile <FILE.lua>` this option will read the lua file to set parameters. Similar functionality can be achieved using –param lua_file="<FILE.lua>".

   A `{{key,value}}` list can also be provided, otherwise it is assumed to be empty. Such a list will set parameter values within this broker. These values will be read and used **BEFORE** the command line is read.

   Finally **AFTER** the command line is read, if the `lua_file` parameter has been set, the configuration file that it indicates will also be read. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(argc, argv, false)`). The `lua_file` will be read **AFTER** the construction key-value list, and after the command like, so it can be used to over-right default values in either.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

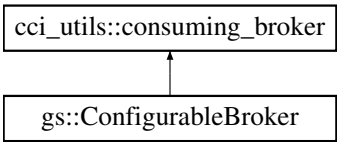
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 gs::ConfigurableBroker Class Reference

Inheritance diagram for gs::ConfigurableBroker:



Public Member Functions

- **ConfigurableBroker** (bool load_conf_file=true)
- **ConfigurableBroker** (std::initializer_list< cci_name_value_pair > list, bool load_conf_file=true)
- **ConfigurableBroker** (const int argc, const char *const *argv, std::initializer_list< cci_name_value_pair > list={{}}, bool load_conf_file=true)
- std::string **relname** (const std::string &n) const
- std::string **relname** (const char *n) const
- cci_originator **get_value_origin** (const std::string &parname) const
- bool **has_preset_value** (const std::string &parname) const
- cci_value **get_preset_cci_value** (const std::string &parname) const
- void **lock_preset_value** (const std::string &parname)
- cci_value **get_cci_value** (const std::string &parname) const
- void **add_param** (cci_param_if *par)
- void **remove_param** (cci_param_if *par)
- void **set_preset_cci_value** (const std::string &parname, const cci_value &cci_value, const cci_originator &originator)
- cci_param_untyped_handle **get_param_handle** (const std::string &parname, const cci_originator &originator) const
- std::vector< cci_param_untyped_handle > **get_param_handles** (const cci_originator &originator) const
- bool **is_global_broker** () const

Public Attributes

- std::set< std::string > **expose**
- cci_param< std::string > **conf_file**

The documentation for this class was generated from the following file:

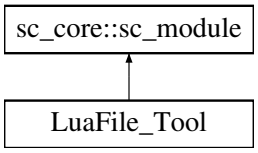
- /Users/mark/work/libgs/tmp/libgsutils/include/greensocs/gsutils/cciutils.h

4.2 LuaFile_Tool Class Reference

Tool which reads a Lua configuration file and sets parameters.

`#include <luafile_tool.h>`

Inheritance diagram for LuaFile_Tool:



Public Member Functions

- LuaFile_Tool (sc_core::sc_module_name name, std::string _orig_name="")
  Constructor.
- int config (const char *config_file)
  Makes the configuration.
- void parseCommandLine (const int argc, const char *const *argv)
  Parses the command line and extracts the luafile option.

Protected Member Functions

- void parseCommandLineWithGetOpt (const int argc, const char *const *argv)
  Parses the command line with getopt and extracts the luafile option.

4.2.1 Detailed Description

Tool which reads a Lua configuration file and sets parameters.

Lua Config File Tool which reads a configuration file and uses the Tool_GCnf_Api to set the parameters during intitialize-mode.

One instance can be used to read and configure several lua config files.

The usage of this Tool:

- instantiate one object
- call config(filename)

4.2.2 Member Function Documentation

4.2.2.1 config()

```
int LuaFile_Tool::config (
            const char * config_file )  [inline]
```

Makes the configuration.

Configure parameters from a lua file.

May be called several times with several configuration files

Example usage:

```
int sc_main(int argc, char *argv[]) {
  LuaFile_Tool luareader;
  luareader.config("file.lua");
  luareader.config("other_file.lua");
}
```

4.2.2.2 parseCommandLine()

```
void LuaFile_Tool::parseCommandLine (
            const int argc,
            const char *const * argv )  [inline]
```

Parses the command line and extracts the luafile option.

Throws a CommandLineException.

Parameters

argc	The argc of main(...).
argv	The argv of main(...).

4.2.2.3 parseCommandLineWithGetOpt()

```
void LuaFile_Tool::parseCommandLineWithGetOpt (
            const int argc,
            const char *const * argv )  [inline], [protected]
```

Parses the command line with getopt and extracts the luafile option.

Throws a CommandLineException.

Parameters

argc	The argc of main(...).
argv	The argv of main(...).

The documentation for this class was generated from the following file:

- /Users/mark/work/libgs/tmp/libgsutils/include/greensocs/gsutils/luafile_tool.h

Index