{{lastupdated_at}} by {{lastupdated_by}}

GammaLib directory structure

This page explains the directory structure of GammaLib. Note that this description applies to the directory that is obtained by cloning the Git repository. For a source code release, Python wrapper files will be present, and also some documentation will be shipped in pdf format. The directory structure of a source code release will thus differ in details.

The directory structure is shown here as it would appear using the ls -la command on a Linux machine. Note that file sizes may differ for your actual copy of the source code as the software evolves.

GammaLib directory

The gammalib directory after cloning from the GammaLib Git repository will have the following content:

```
drwxr-xr-x   13 jurgen  staff    442 12 oct 21:35 .git           <= Git information
-rw-r--r--    1 jurgen  staff    218 12 oct 21:35 AUTHORS
-rw-r--r--    1 jurgen  staff  35146 12 oct 21:35 COPYING
-rw-r--r--    1 jurgen  staff   6358 12 oct 21:35 ChangeLog
-rw-r--r--    1 jurgen  staff  11397 12 oct 21:35 INSTALL
-rw-r--r--    1 jurgen  staff   6139 12 oct 21:35 Makefile.am     <- automake input file
-rw-r--r--    1 jurgen  staff   2770 12 oct 21:35 NEWS
-rw-r--r--    1 jurgen  staff  11606 12 oct 21:35 README
-rwxr-xr-x    1 jurgen  staff    228 12 oct 21:35 autogen.sh      <- autotools generation script
-rw-r--r--    1 jurgen  staff  52654 12 oct 21:35 configure.ac    <- autoconf input file
drwxr-xr-x    9 jurgen  staff    306 12 oct 21:35 dev            <= code development utilities
drwxr-xr-x    8 jurgen  staff    272 12 oct 21:35 doc            <= GammaLib documentation
drwxr-xr-x    3 jurgen  staff    102 12 oct 21:35 examples       <= GammaLib usage examples
-rw-r--r--    1 jurgen  staff    397 12 oct 21:35 gammalib.pc.in  <- pkgconfig input file
drwxr-xr-x  129 jurgen  staff   4386 12 oct 21:35 include        <= include files for instrument independent code
drwxr-xr-x    6 jurgen  staff    204 12 oct 21:35 inst           <= instrument dependent code
drwxr-xr-x   12 jurgen  staff    408 12 oct 21:35 m4             <= autoconf macros
drwxr-xr-x  128 jurgen  staff   4352 12 oct 21:35 pyext          <= Python binding definition files (for swig)
drwxr-xr-x   18 jurgen  staff    612 12 oct 21:35 src            <= instrument independent code
drwxr-xr-x   52 jurgen  staff   1768 12 oct 21:35 test           <= code for unit testing
-rwxr-xr-x    1 jurgen  staff   1418 12 oct 21:35 testall.sh      <- extended test script
```

GammaLib uses autotools for configuration (see below). There are two central input files that will be used by autotools: Makefile.am will be used to generate Makefile, which is the script invoked by the make command. configure.ac will be used to generate configure, which is the script that is called for configuration of the software prior to compilation.

GammaLib supports the pkg-config helper system. The file gammalib.pc.in will be used to generate gammalib.pc using the autotools.

In the following, the subdirectories of the gammalib directory are described.

dev

The dev directory contains are couple of scripts and files that are useful for GammaLib development and GammaLib release. Most developers will not need to use one of these scripts. Below the content of the dev directory:

```
-rw-r--r--  1 jurgen  staff    925 12 oct 21:35 license_new.txt      <- New license text (used by replace_license.py)
-rw-r--r--  1 jurgen  staff    308 12 oct 21:35 license_old.txt      <- Old license text (used by replace_license.py)
-rwxr-xr-x  1 jurgen  staff   5122 12 oct 21:35 macosx_package.sh    <- Create a GammaLib / ctools binary for packaging
-rwxr-xr-x  1 jurgen  staff  13654 12 oct 21:35 macosx_testconf.sh   <- Tests GammaLib under various Mac OS X configurations
-rwxr-xr-x  1 jurgen  staff  12016 12 oct 21:35 macosx_testpkg.sh    <- Tests GammaLib / ctools Mac OS X package prior to release
-rwxr-xr-x  1 jurgen  staff   2997 12 oct 21:35 release_gammalib.sh  <- Create a GammaLib release
-rwxr-xr-x  1 jurgen  staff   2586 12 oct 21:35 replace_license.py   <- Replace license text in all source and include files
```

doc

The doc directory contains the GammaLib documentation. Here the content of the directory:

```
-rw-r--r--   1 jurgen  staff  57183 12 oct 21:35 Doxyfile  <- Configuration file for generation of Doxygen documentation
drwxr-xr-x   8 jurgen  staff    272 12 oct 21:35 dev       <= Development documentation
drwxr-xr-x  14 jurgen  staff    476 12 oct 21:35 html      <= Web documentation
drwxr-xr-x   3 jurgen  staff    102 12 oct 21:35 um        <= User Manual
```

GammaLib uses Doxygen to generate the reference documentation. The file Doxyfile contains the Doxygen configuration that is applied for this generation.

The dev directory contains all documentation that is relevant for code development. It contains a number of subdirectories, one for each development document (except of the tn directory which contains all technical notes of the project). Here the actual content of dev:

```
drwxr-xr-x  4 jurgen  staff  136 12 oct 21:35 coding  <= Coding and Design Conventions
drwxr-xr-x  7 jurgen  staff  238 12 oct 21:35 inst    <= Instrument specific interface
drwxr-xr-x  5 jurgen  staff  170 12 oct 21:35 maths   <= Mathematical Implementation
drwxr-xr-x  6 jurgen  staff  204 12 oct 21:35 sdd     <= Software Design Description
drwxr-xr-x  4 jurgen  staff  136 12 oct 21:35 srs     <= Software Requirement Specification
drwxr-xr-x  4 jurgen  staff  136 12 oct 21:35 tn      <= Technical Notes
```

The html directory contains all Web documentation of GammaLib. The Web documentation is published at the following sourceforge site: http://gammalib.sourceforge.net/

The um directory contains the User Manual for GammaLib.

Note that much of the documentation is not up to date. The only document that can be considered reasonably complete are the "Coding and Design Conventions".

examples

This directory contains example code that illustrates the usage of GammaLib. So far, only a single example named readmodel exists. To compile this example, step into the examples/readmodel directory and type

```
$ make
```

Note that GammaLib needs to be installed prior to compilation. Also, the Makefile assumes that GammaLib has been installed in the path /usr/local/gamma. If GammaLib is installed in a different location, please adjust the Makefile.

include

The include directory contains all instrument independent header files for GammaLib. The directory is flat, i.e. no structuration into modules exists.

To add a new class to GammaLib, simply add a header file to this directory. Add the name of the header file also to the Makefile.am script, and add a

```
#include "GClass.hpp"
```

directive to GammaLib.hpp to include the new header in the GammaLib master header file (in this example we supposed that the new header file is named GClass.hpp).

inst

This directory contains the instrument specific source code. There is one directory per instrument. The Makefile.am gathers information for all subdirectories, and it is one of the files that needs to be adapted when a new instrument should be added.

A typical instrument specific directory has the following structure (here the example of cta):

```
-rw-r--r--  1 jurgen  staff  2732 12 oct 21:35 Makefile.am  <- automake input file
drwxr-xr-x  15 jurgen  staff   510 12 oct 21:35 caldb        <= calibration data
drwxr-xr-x  21 jurgen  staff   714 12 oct 21:35 include      <= include files
drwxr-xr-x  20 jurgen  staff   680 12 oct 21:35 pyext        <= Python binding definition files (for swig)
drwxr-xr-x  24 jurgen  staff   816 12 oct 21:35 src          <= source code
drwxr-xr-x  18 jurgen  staff   612 12 oct 21:35 test         <= code for unit testing
```

The Makefile.am is the input file that will be used by automake to generate a Makefile. This file needs to be modified if code is added to the interface.

The structure of the caldb directory depends strongly on the instrument. The data contained in this directory will be used for unit testing, and will be installed together with the software.

The include directory is a flat directory, containing all header files for the instrument dependent code. It's structure is similar to the include directory of the instrument independent code. Instead of the GammaLib.hpp file (which gathers all headers for the instrument independent code), a file gathering all headers for the instrument dependent code will exist (for CTA this file is named GCTALib.hpp).

The pyext directory is a flat directory, containing all interface files that will be processed by swig to create the Python wrapper files. There is one interface file for each class. All interface files a gathered in a summary file (for CTA this file is named cta.i). If a class interface should be added it is sufficient to add the respective file to the summary file.

The src directory is a flat directory, containing all source code.

The test directory contains all code and data needed for unit testing. In addition, it may contain an (arbitrary) number of test scripts and programs that are useful for testing the functionalities of the instrument specific interface. Here the example for the cta interface:

```
-rw-r--r--  1 jurgen  staff    982 12 oct 21:35 README
drwxr-xr-x  14 jurgen  staff    476 12 oct 21:35 data
-rwxr-xr-x  1 jurgen  staff   2338 12 oct 21:35 example_binned_ml_fit.py
-rwxr-xr-x  1 jurgen  staff   1672 12 oct 21:35 example_make_model.py
-rwxr-xr-x  1 jurgen  staff   3889 12 oct 21:35 example_sim_photons.py
-rw-r--r--  1 jurgen  staff  20474 12 oct 21:35 test_CTA.cpp
-rw-r--r--  1 jurgen  staff   3868 12 oct 21:35 test_CTA.hpp
-rwxr-xr-x  1 jurgen  staff   1578 12 oct 21:35 test_CTA.py
-rwxr-xr-x  1 jurgen  staff   1998 12 oct 21:35 test_gauss.py
-rwxr-xr-x  1 jurgen  staff   5056 12 oct 21:35 test_irf_offset.py
-rwxr-xr-x  1 jurgen  staff   1561 12 oct 21:35 test_irf_trafo.py
-rwxr-xr-x  1 jurgen  staff   6408 12 oct 21:35 test_model.py
-rwxr-xr-x  1 jurgen  staff   3388 12 oct 21:35 test_npred_computation.py
-rwxr-xr-x  1 jurgen  staff   3492 12 oct 21:35 test_npred_intergation.py
-rwxr-xr-x  1 jurgen  staff   2393 12 oct 21:35 test_radial_acceptance.py
-rwxr-xr-x  1 jurgen  staff   1944 12 oct 21:35 test_response_table.py
```

The README file contains more information about the various test files that exist in this directory. The data directory contains data needed for unit testing. The files test_CTA.cpp and test_CTA.hpp are used for unit testing of the C++ code, the file test_CTA.py is used for unit testing of the Python interface.

m4

This directory contains autoconf macros. Some of the macros are actually used by the configure script during testing of the system configuration, other macros are still there for legacy.

In case you need to add a test to the configure script that makes use of a non-standard macro, please add the macro to this folder to make sure that it will be available during the configuration step.

pyext

The pyext directory contains all interface files that will be processed by swig to create the Python wrapper files. There is one interface file for each class. All these files are directly located in the pyext directory.

The Python interface is organized into modules, mainly to reduce the size of the Python wrapper code. The definitions of the modules are found in the gammalib subdirectory, which has the following content:

```
-rw-r--r--  1 jurgen  staff  2194 12 oct 21:35 app.i
-rw-r--r--  1 jurgen  staff  3024 12 oct 21:35 fits.i
-rw-r--r--  1 jurgen  staff  2234 12 oct 21:35 linalg.i
-rw-r--r--  1 jurgen  staff  3101 12 oct 21:35 model.i
-rw-r--r--  1 jurgen  staff  2209 12 oct 21:35 numerics.i
-rw-r--r--  1 jurgen  staff  2543 12 oct 21:35 obs.i
-rw-r--r--  1 jurgen  staff  2225 12 oct 21:35 opt.i
-rw-r--r--  1 jurgen  staff  2348 12 oct 21:35 sky.i
-rw-r--r--  1 jurgen  staff  2070 12 oct 21:35 support.i
-rw-r--r--  1 jurgen  staff  2186 12 oct 21:35 test.i
-rw-r--r--  1 jurgen  staff  2298 12 oct 21:35 xml.i
```

Each of these files is a summary file for one module. By means of %include directives, each summary file includes the interface files that should be gathered into the module. If a new class should be added to a module you have to add an %include directive to the respective module. Please make sure that a given interface file is not included in several modules.

The pyext directory contains a Makefile.am that defines how the modules will be built. The files needs to be modified when a new module (or a new instrument) should be added. If a new class is added to an existing module, no modification is required.

The directory also contains a file setup.py.in that is used for building and installing the Python wrappers. This file will be converted into a file setup.py by the configuration step. The files needs to be modified when a new module (or a new instrument) should be added. If a new class is added to an existing module, no modification is required.

src

The src directory is where all instrument independent source code lives. The directory has the following content:

```
-rw-r--r--   1 jurgen  staff  2074 12 oct 21:35 Makefile.am
drwxr-xr-x   8 jurgen  staff   272 12 oct 21:35 app
drwxr-xr-x  36 jurgen  staff  1224 12 oct 21:35 fits
-rwxr-xr-x   1 jurgen  staff   814 12 oct 21:35 gammalib-init.csh
-rwxr-xr-x   1 jurgen  staff   793 12 oct 21:35 gammalib-init.sh
-rw-r--r--   1 jurgen  staff  5636 12 oct 21:35 gammalib-setup.in
drwxr-xr-x  13 jurgen  staff   442 12 oct 21:35 linalg
drwxr-xr-x  35 jurgen  staff  1190 12 oct 21:35 model
drwxr-xr-x   8 jurgen  staff   272 12 oct 21:35 numerics
drwxr-xr-x  25 jurgen  staff   850 12 oct 21:35 obs
drwxr-xr-x   7 jurgen  staff   238 12 oct 21:35 opt
drwxr-xr-x  15 jurgen  staff   510 12 oct 21:35 sky
drwxr-xr-x   8 jurgen  staff   272 12 oct 21:35 support
drwxr-xr-x   4 jurgen  staff   136 12 oct 21:35 template
drwxr-xr-x   7 jurgen  staff   238 12 oct 21:35 test
drwxr-xr-x  12 jurgen  staff   408 12 oct 21:35 xml
```

Each subdirectory presents a specific module. As example, the content of the app subdirectory is

```
-rw-r--r--  1 jurgen  staff  20078 12 oct 21:35 GApplication.cpp
-rw-r--r--  1 jurgen  staff   8383 12 oct 21:35 GException_app.cpp
-rw-r--r--  1 jurgen  staff  26817 12 oct 21:35 GLog.cpp
-rw-r--r--  1 jurgen  staff  32253 12 oct 21:35 GPar.cpp
-rw-r--r--  1 jurgen  staff  39490 12 oct 21:35 GPars.cpp
-rw-r--r--  1 jurgen  staff    499 12 oct 21:35 Makefile.am
```

These directories contain the code for all GammaLib classes. Typically, there is one file per class. The Makefile.am in these subdirectories gather the names of all files that need be compiled. If a new file is added to the subdirectory, Makefile.am needs to be updated.

Often, the subdirectories contain a file named GException_xxx.cpp, where xxx is the name of the module. This file implements the exception classes that are relevant for this module. The definition of these exception classes are found in the file GException.hpp, situated in the gammalib/include directory.

The src directory also contains a Makefile.am which gathers informations about all modules and instrument interfaces that are present. If a module is added ot if a new instrument interface is added, this file needs to be modified.

Finally, the src directory contains the scripts gammalib-init.csh, gammalib-init.sh, and gammalib-setup.in. These scripts will be installed in the bin directory of the GammaLib installation, and will serve to configure the GammaLib environment. If modifications to the environment are required, the file gammalib-setup.in should be changed.

test

The test directory contains all code and data that are needed for unit testing. In addition, it may contain any kind of script that is useful for testing some functionalities of GammaLib. Here is a selected list of files or folders that are present in the directory:

```
-rw-r--r--   1 jurgen  staff   5513 12 oct 21:35 Makefile.am           <- automake input file
-rw-r--r--   1 jurgen  staff    755 12 oct 21:35 README
drwxr-xr-x   7 jurgen  staff    238 12 oct 21:35 data                  <= test data directory
drwxr-xr-x   3 jurgen  staff    102 12 oct 21:35 reports               <= test reports directory
-rw-r--r--   1 jurgen  staff   4960 12 oct 21:35 test_GApplication.cpp  <- source code for C++ module test
-rw-r--r--   1 jurgen  staff   2454 12 oct 21:35 test_GApplication.hpp  <- header file for C++ module test
-rwxr-xr-x   1 jurgen  staff   4115 12 oct 21:35 test_GApplication.py   <- script for Python module test
drwxr-xr-x  13 jurgen  staff    442 12 oct 21:35 testinst              <= Instrument for testing of instrument interface
```

The Makefile.am file is the input file from which automake will generate a Makefile. This Makefile will be executed when make check is invoked. If a new module or a new instrument is added, this file needs to be modified.

For each module there is a C++ source code file, a C++ header file and a Python script. These files will be used during make check to perform the C++ and Python unit tests for a given module.

The testinst directory implements a test instrument that is used to test the instrument interface. The test instrument is realized by header files only, hence no compilation is performed. To include the test instrument in any unit test, the testinst/GTestLib.hpp file needs to be included in the test. See test_GObservation.cpp for an example.