{{lastupdated_at}} by {{lastupdated_by}}

Contributing to ctools

This page explains how you can contribute to the development of the ctools library.

Prerequisits

You will need the following software installed on your system to contribute to the ctools development:

- C++ compiler (e.g. GNU gcc)
- Git
- autoconf
- automake
- libtool
- cfitsio
- Python
- swig (version 2.0.4 or later)

Optionally, it is useful to have:

- doxygen
- sphinx

If some of the software is not yet installed on your system, it is very likely that you can install it through your system's package manager. On Mac OS X, you may use the Homebrew or MacPorts package managers. Make sure that you install the development packages of cfitsio, Python and readline as they provide the header files that are required for compilation.

Getting the source code

ctools uses Git for version control.

To learn how Git is used within the ctools project, please familiarize yourself with the [[Git workflow]].

The central ctools Git repository can be found at https://cta-gitlab.irap.omp.eu/ctools/ctools.git. A mirror of the central repository is available on GitHub at https://github.com/ctools/ctools. Both repositories are read-only, and are accessed using the https protocol. Both on Gitlab and GitHub you may fork the ctools repository and develop your code using this fork (read [[Git workflow]] to learn how).

The default branch from which you should start your software development is the devel branch. **devel is ctools's trunk.** The command

```
$ git clone https://cta-gitlab.irap.omp.eu/ctools/ctools.git
```

will automatically clone this branch from the central ctools Git repository.

Software developments are done in feature branches. When the development is finished, issue a pull request so that the feature branch gets merged into devel. Merging is done by the ctools integration manager.

Preparing ctools for configuration

After cloning ctools (see above) you will find a directory called ctools. Before this directory can be used for development, we have to prepare it for configuration. **If you're not familiar with the autotools, please read this section carefully so that you get the big picture.**

Step into the ctools directory and prepare for configuration by typing

```
$ cd ctools
$ ./autogen.sh
```

Configuring ctools

There is a single command to configure ctools:

```
$ ./configure
```

Note that configure is a script that has been generated previously by the autoconf step. A typical output of the configuration step is provided in the file attachment:configure.out.

Compiling ctools

ctools is compiled by typing

```
$ make
```

Running unit tests

Introduction

The ctools unit test suite is run using

```
$ make check
```

Running a subset of the unit tests

To be written

Debugging unit tests

To be written

Debugging Python unit tests

To be written

Installing ctools

You have to execute the additional make install step and source the ctools-init.sh setup file so that you'll actually use the new version of ctools:

```
$ export CTOOLS=<wherever you want>
$ ./configure --prefix=$CTOOLS
$ make install
$ source $CTOOLS/bin/ctools-init.sh
```

Profiling ctools

To be written.

Adding a new ctool to ctools

To be written.

Adding a new cscript to ctools

To be written.