ctools - Feature #572

Make ctools help pages available via fhelp

10/15/2012 08:25 PM - Deil Christoph

Status:	Closed	**Start date:**	10/15/2012
Priority:	Normal	**Due date:**	
Assigned To:	Deil Christoph	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

All other ftools I know (HEASOFT, CIAO, FERMI) provide a help page for every tool that is available via

```
fhelp ftool_name
```

This sometimes opens a txt file with less or an html file in my browser and generally I find fhelp pages pretty useful.

I can see that there html help files are available for ctools in ctools/doc/html/doc

- Are these hand-written html pages or are they generated from some other text file?
- Is it possible to install them so that fhelp will find them?

History

#1 - 10/15/2012 08:48 PM - Knödlseder Jürgen

The html files are hand written.

I think you asked already in the past for the fhelp thing. I remember that I looked into this, but I'm not sure what the conclusion was. I think to remember that fhelp hardcodes the paths where it looks for the help file, so the only solution was to write another tool ... but maybe there is also another solution. I think it is sufficient to look up the fhelp script ...

#2 - 06/29/2015 05:03 PM - Deil Christoph

- Target version set to 1.0.0

I'm putting this for 1.0 ... having an easy way to print the help and list the parameters for each command line tool is very important in day-to-day usage.

I think `ctlike --help` is best ... that's how all Unix tools work.

`cshelp ctlike` would be nice too ... I could implement this and also add an option so the user can list all tools and which version of the software is installed.

The FTOOLS way would be to call `fhelp` or `fthelp`, which I think sometimes opens up a txt file in the console and sometimes an html file in the browser.

Jürgen, how do you want to do this for ctools?
I could work on this this week.

#3 - 06/29/2015 09:55 PM - Knödlseder Jürgen

Have you tried man ctlike?

#4 - 06/30/2015 12:05 AM - Deil Christoph

Yes, I had tried man ctlike ... the manpage isn't there for me:

```
$ man ctlike
No manual entry for ctlike
$ find /usr/local/ -name '*ctlike*'
/usr/local//gamma/bin/ctlike
/usr/local//gamma/share/ctools/swig/ctlike.i
/usr/local//gamma/syspfiles/ctlike.par
```

?

#5 - 06/30/2015 10:11 AM - Mayer Michael

man ctlike actually works for me

#6 - 06/30/2015 11:19 AM - Knödlseder Jürgen

- Assigned To set to Deil Christoph

Decided to use rst files on input, convert them into plain text files and install them in shared area.

Implement in GApplication the --help (and/or -h) option.

#7 - 06/30/2015 12:44 PM - Kelley-Hoskins Nathan

- File clean.rst.file.py added

I've written a python script (see attached) that should clean up rst files for easier human readability. I tested it on the ones in $CTOOLS/doc/source/reference_manual, and it works pretty well. See the result:

```
.. _ctlike:

ctlike
======

Performs binned and unbinned maximum likelihood analysis of CTA data.

Synopsis
--------

Determines source model parameters, such as flux, spectral index, source
position, and source extent from a maximum likelihood analysis of CTA data.
The analysis can be done in a binned or an unbinned formulation of the
log-likelihood function.
For binned analysis, a counts cube produced by :doc:`ctbin` is required.
For unbinned analysis, an event list processed by :doc:`ctselect` is required.
Based on the input file format, ctlike automatically selects between binned
and unbinned maximum likelihood analysis.
```

General parameters

``inobs [file]``
  Input event list, counts cube or observation definition XML file.

``inmodel [string]``
  Source model input XML file.

``expcube [file]``
  Exposure cube file (only needed for stacked analysis).

``psfcube [file]``
  PSF cube file (only needed for stacked analysis).

``bkgcube [file]``
  Background cube file (only needed for stacked analysis).

``caldb [string]``
  Calibration database.

``irf [string]``
  Instrument response function.

``outmodel [string]``
  Source model result XML file with values and uncertainties updated by
  the maximum likelihood fit.

``(stat = POISSON) [string]``
  Fitting statistics (POISSON or GAUSSIAN; only affects binned analysis).

``(edisp = no) [boolean]``
  Applies energy dispersion to response computation.

``(refit = no) [boolean]``
  Performs refitting of solution after initial fit.

Standard parameters

``(chatter = 2) [integer]``
  Verbosity of the executable:
   chatter = 0: no information will be logged

   chatter = 1: only errors will be logged

   chatter = 2: errors and actions will be logged

   chatter = 3: report about the task execution

   chatter = 4: detailed report about the task execution

``(clobber = yes) [boolean]``
  Specifies whether an existing output counts cube should be overwritten.

``(debug = no) [boolean]``
  Enables debug mode. In debug mode the executable will dump any log file output to the console.

``(mode = ql) [string]``
  Mode of automatic parameters (default is "ql", i.e. "query and learn").

``(logfile = ctlike.log) [string]``
  Name of log file.

Related tools

None

After:

ctlike
======

Performs binned and unbinned maximum likelihood analysis of CTA data.

Synopsis

Determines source model parameters, such as flux, spectral index, source
position, and source extent from a maximum likelihood analysis of CTA data.
The analysis can be done in a binned or an unbinned formulation of the
log-likelihood function.
For binned analysis, a counts cube produced by :doc:`ctbin` is required.
For unbinned analysis, an event list processed by :doc:`ctselect` is required.
Based on the input file format, ctlike automatically selects between binned
and unbinned maximum likelihood analysis.

General parameters
----

inobs [file]
  Input event list, counts cube or observation definition XML file.

inmodel [string]
  Source model input XML file.

expcube [file]
  Exposure cube file (only needed for stacked analysis).

psfcube [file]
  PSF cube file (only needed for stacked analysis).

bkgcube [file]
  Background cube file (only needed for stacked analysis).

caldb [string]
  Calibration database.

irf [string]
  Instrument response function.

outmodel [string]
  Source model result XML file with values and uncertainties updated by
  the maximum likelihood fit.

(stat = POISSON) [string]
  Fitting statistics (POISSON or GAUSSIAN; only affects binned analysis).

(edisp = no) [boolean]
  Applies energy dispersion to response computation.

(refit = no) [boolean]
  Performs refitting of solution after initial fit.

Standard parameters

(chatter = 2) [integer]
  Verbosity of the executable:
   chatter = 0: no information will be logged

   chatter = 1: only errors will be logged

   chatter = 2: errors and actions will be logged

   chatter = 3: report about the task execution

   chatter = 4: detailed report about the task execution

(clobber = yes) [boolean]
  Specifies whether an existing output counts cube should be overwritten.

(debug = no) [boolean]
  Enables debug mode. In debug mode the executable will dump any log file output to the console.

(mode = ql) [string]
    Mode of automatic parameters (default is "ql", i.e. "query and learn").

(logfile = ctlike.log) [string]
    Name of log file.

Related tools

None

Usage:

$ clean.rst.file.py input_filename output_filename

#8 - 10/28/2015 12:08 PM - Knödlseder Jürgen

- Status changed from New to Pull request

- % Done changed from 0 to 100

I added a need_help() method to GApplication that catches the --help option that is specified on the command line (it is for the moment assumed that --help is the only option that is specified on the command line).

On the ctools side, I added some code to the ctool base class that handles the case that the need_help() returns true (i.e. that the --help option has been specified on the command line). Help files named <tool>.txt and installed in $CTOOLS/share/help will be displayed to the console.

I added a target to the Makefile that generates the <tool>.txt files for all tools and scripts from the .rst files during the install process. I reimplemented clean.rst.file.py using sed so that the file translation is directly done at the Makefile level without requiring an external Python script.

The code is ready to go into devel.

#9 - 10/28/2015 12:25 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

Merged into devel.

Files

clean.rst.file.py	1.02 KB	06/30/2015	Kelley-Hoskins Nathan