


# Help Generation for ROOT Related Commands

By Elie Khairallah



# Types of help

## Two ways to get help for the root commands

- Man pages (*man root.1*) 
- Command line option help (*root -h*)



```
[macbookepsft:python epsftss$ root -h
```

```
usage: root [-b B] [-x X] [-e E] [-n N] [-t T] [-q Q] [-l L] [--config CONFIG]
          [--memstat MEMSTAT] [-h HELP] [--notebook NOTEBOOK] [--web WEB]
          dir
```

### OPTIONS:

-b	Run in batch mode without graphics
-x	Exit on exceptions
-e	Execute the command passed between single quotes
-n	Do not execute logon and logoff macros as specified in .rootrc
-t	Enable thread-safety and implicit multi-threading (IMT)
-q	Exit after processing command line macro files
-l	Do not show splash screen
--config	print ./configure options
--memstat	run with memory usage monitoring
-h, -?, --help	Show summary of options
--notebook	Execute ROOT notebook
--web	Display graphics in a web browser
dir	if dir is a valid directory cd to it before executing

```
ROOT(1)
NAME
  root a Interpreter of C++ for the ROOT framework
SYNOPSIS
  [datafile ...] [macrofile ...] macrofile.C macrofile.C(arguments ...) macrofile.C("string arguments") macrofile.C+ macrofile.C++ macrofile.C+g macrofile.C+g
DESCRIPTION
  ROOT's Object-Oriented Technologies.
  root is an interactive interpreter of C++ code. It uses the ROOT framework. For more information on ROOT, please refer to
  An extensive Users Guide is available from that site (see below).
OPTIONS
  -? Show summary of options.
  -h Show summary of options.
  -b Run in batch mode without graphics.
  -x Exit on exceptions.
  -e Execute the command passed between single quotes
  -n Do not execute logon and logoff macros as specified in .rootrc
  -t Enable thread-safety and implicit multi-threading (IMT)
  -q Exit after processing command line macro files
  -l Do not show splash screen
  --config print ./configure options
  --memstat run with memory usage monitoring
  --help Show summary of options.
  --notebook Execute ROOT notebook.
  --web Display graphics in a web browser.
dir if dir is a valid directory cd to it before executing
Data files are opened and accessible in root as _file#, _file# ...
Macro files are either interpreted (filename provided alone) or compiled (with + added to the filename). Afterwards the function with the same name as the filename (without extension) in the macro file is
executed (eg. void macro() in a file macro.C).
Compilation is done on-demand if the macro is newer than a previously generated shared library. Two plus signs (macro.C++) force a recompilation. Adding Q after a plus results in an optimised build, adding
g adds debug symbols (eg. macro.C+g).
Macro files, data files and _g expressions are processed in the order in which they are provided. If a macro relies on the presence of _file#, then root should be called with root data.root macro.C.
SEE ALSO
  rootcling(1), cling(1), root-config(1), root(1), hroot(1), s2root(1)
For extensive documentation on the ROOT system, see
A Users Guide is available at
The classes of ROOT are all documented by the automatic documentation system, and is available at
Questions and support are provided in the root forum bugs can be reported on jira and pull requests can be submitted on github
FILES
  <etcdir>/system.rootrc
    System-wide configuration file. <etcdir> either $ROOTSYS, or something like /etc/root
  <libdir>/#
    ROOT C++ class libraries. <libdir> is either $ROOTSYS/lib or something like /usr/lib/root.
  <incdir>/#
    The header files for the ROOT C++ class libraries. <incdir> is either $ROOTSYS/include or something like /usr/include/root.
  -/.rootrc, ./rootrc
    User configuration file
ORIGINAL AUTHORS
  The ROOT team (see web page above):
  Rene Brun and Fons Rademakers
COPYRIGHT
  This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published
  by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
  This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
  or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Founda-
  tion, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
AUTHORS
  This manual page was written by Christian Holm Christensen <holm@mbi.dk>, for the Debian GNU/Linux system (but may be used by others)..
  Version 6
  ROOT(1)
```

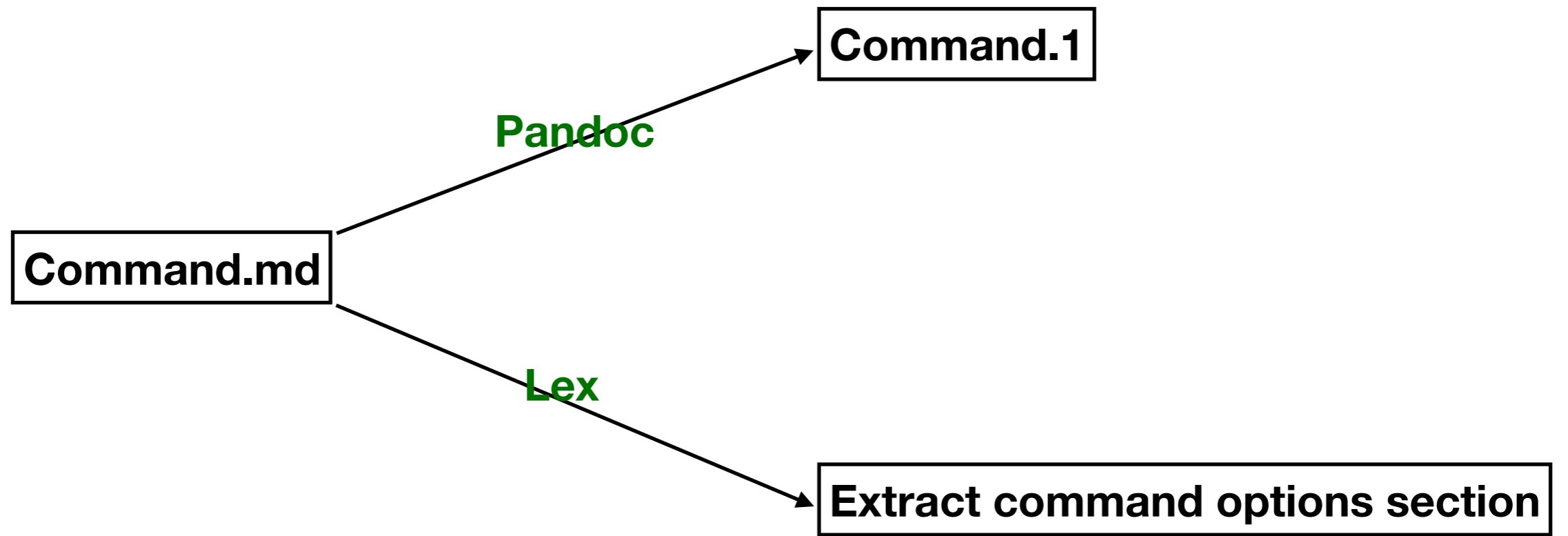
# Problems with static implementation

## Two main problems

- **Man page and usage become out of sync**
- **Lack of pattern**
  - **Usage as big text with examples**
  - **Usage as series of option name and help message**
  - **Usage written in header**
  - **Usage as output stream commands**
  - **Usage handled by argparse module**



# First Approach



# Output and Problems

Simple! And we had the desired output on both the man page and usage level

but

...

...

...

- Lex is not cross platform compatible
- Additional features provided by modules like argparse should be manually reproduced

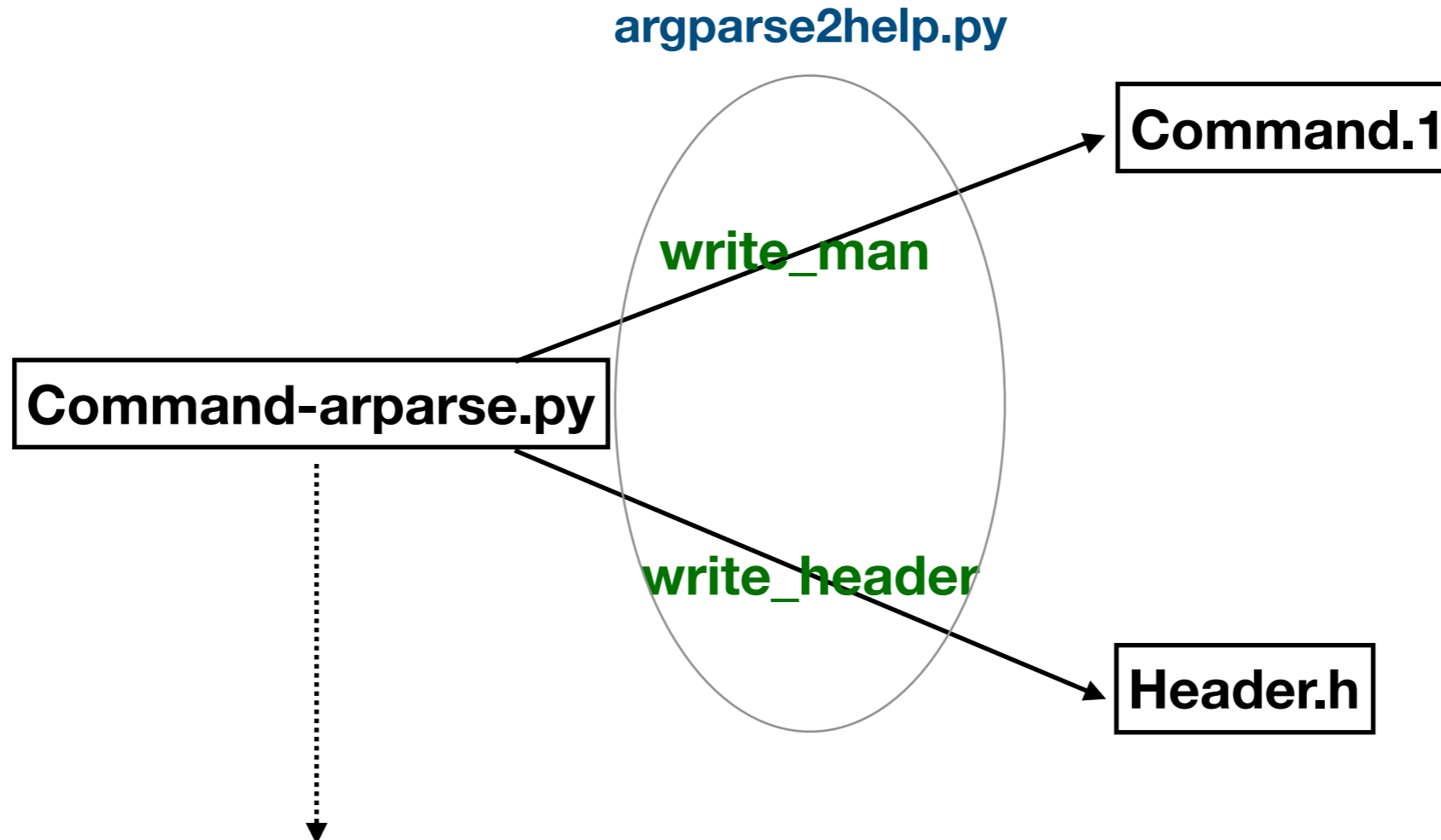
```
1  %{  
2  #include <iostream>  
3  #include <string>  
4  }  
5  %x OPTIONS  
6  %%  
7  ^"# OPTIONS\n\n" BEGIN OPTIONS;  
8  <OPTIONS>^"*" printf(" ");  
9  <OPTIONS>"*" ;  
10 <OPTIONS>"\n\n" printf("\n");  
11 <OPTIONS>^" BEGIN 0;  
12 <OPTIONS>"\" ;  
13 . ;  
14 "\n" ;  
15 %%  
16 int yywrap()  
17 {  
18 return 1;  
19 }  
20 int main()  
21 {yylex();}
```

Lex Program

```
[macbookepsft:week1 epsftss$ ./a.out < ./root.md  
-? Show summary of options.  
-h Show summary of options.  
-b Run in batch mode without graphics.  
-x Exit on exceptions.  
-e Execute the command passed between single quotes  
-n Do not execute logon and logoff macros as specified in .rootrc  
-t Enable thread-safety and implicit multi-threading (IMT)  
-q Exit after processing command line macro files  
-l Do not show splash screen  
-config print ./configure options  
-memstat run with memory usage monitoring  
--help Show summary of options.  
--notebook Execute ROOT notebook.  
--web Display graphics in a web browser.  
dir if dir is a valid directory cd to it before executing
```

Lex Output

# New Approach



```
1 import argparse
2
3 def get_argparse():
4     parser = argparse.ArgumentParser(add_help=False, prog='root',
5     description="""ROOTs Object-Oriented Technologies.\n
6     root is an interactive interpreter of C++ code. It uses the ROOT framework. For more information on ROOT, please refer to\n
7     An extensive Users Guide is available from that site (see below).
8     """)
9     parser.add_argument('-b', help='Run in batch mode without graphics')
10    parser.add_argument('-x', help='Exit on exceptions')
11    parser.add_argument('-e', help='Execute the command passed between single quotes')
12    parser.add_argument('-n', help='Do not execute logon and logoff macros as specified in .rootrc')
13    parser.add_argument('-t', help='Enable thread-safety and implicit multi-threading (IMT)')
14    parser.add_argument('-q', help='Exit after processing command line macro files')
15    parser.add_argument('-l', help='Do not show splash screen')
16    parser.add_argument('-config', help='print ./configure options')
17    parser.add_argument('-memstat', help='run with memory usage monitoring')
18    parser.add_argument('-h','-?', '--help', help='Show summary of options')
19    parser.add_argument('--notebook', help='Execute ROOT notebook')
20    parser.add_argument('--web', help='Display graphics in a web browser')
21    parser.add_argument('--dir', help='if dir is a valid directory cd to it before executing')
22    return parser
```

For the commands written in c++ we generate both header and manual.

For the python file, the usage is handled by argparse directly, we need to generate the man page only.

```
5 def getLongest():-
6     longestSize = 0-
7     for arg in listArgs:-
8         if (len(arg.option_strings)==0):-
9             size = len(arg.dest)-
10            else:-
11                size = len(", ".join(arg.option_strings))-
12            longestSize = max(longestSize, size)-
13            return longestSize-
14
15 def write_header(parser, fileName):-
16     longestSize = getLongest()-
17     file = open(fileName, "w+")-
18     splitPath = sys.argv[2].split("/")-
19     file.write("#ifndef ROOT_{}\n".format(splitPath[len(splitPath)-1].partition(".")[0]))-
20     file.write("#define ROOT_{}\n".format(splitPath[len(splitPath)-1].partition(".")[0]))-
21     file.write("constexpr static const char kCommandLineOptionsHelp[] = R\"RAW(\n")-
22     file.write(parser.format_usage() + "\n")-
23     file.write("OPTIONS:\n")-
24     for arg in listArgs:-
25         options = ""-
26         help = arg.help-
27         if (len(arg.option_strings)==0):-
28             listOptions = [arg.dest]-
29         else:-
30             listOptions = arg.option_strings-
31         options = ", ".join(listOptions)-
32         spaces = " " * (12 + longestSize - len(options))-
33         if help != None:-
34             help = help.replace("\n", "\n·{}".format(" " * (len(options) + spaces)))-
35             file.write("·{}{}\n".format(options, spaces, help))-
36         else:-
37             file.write("·{}\n".format(options))-
38     file.write(")RAW\";\n")-
39     file.write("#endif\n")-
40     file.close()-
```

Function generating the header file

```
42 def write_man(parser, fileName):-
43     file = open(fileName, "w+")-
44     file.write(".TH·{}·1·\n".format(parser.prog))-
45     file.write(".SH·SYNOPSIS\n")-
46     file.write(parser.format_usage() + "\n")-
47     file.write(".SH·DESCRIPTION\n")-
48     file.write(parser.description + "\n")-
49     file.write(".SH·OPTIONS\n")-
50     for arg in listArgs:-
51         options = ""-
52         help = arg.help-
53         if (len(arg.option_strings)==0):-
54             listOptions = [arg.dest]-
55         else:-
56             listOptions = arg.option_strings-
57         options = "\·".join(listOptions)-
58         if help != None:-
59             file.write(".IP·{}\n".format(options))-
60             file.write(help.replace("\n", "\n.IP\n") + "\n")-
61         else:-
62             file.write(".IP·{}\n\n".format(options))-
63     file.close()-
```

Function generating the man pages

# Commands

C++

**\_root**  
**\_hadd**  
**\_rootcling**  
**\_hist2workspace**

**\_rootls**  
**\_rootbrowse**  
**\_rooteventselector**

Python

**\_rootmv**  
**\_rootprint**  
**\_rootrm**  
**\_rootslmtree**  
**\_rootmkdir**  
**\_rootcp**  
**\_rootdrawtree**



# Modifications

c++ commands



replace old help code by the display of the raw string in header

Python command



split old code into two, one being the argparse python file

```
1 #!/usr/bin/env python@-
2 -
3 # ROOT command line tools: rootls@-
4 # Author: Julien Ripoche@-
5 # Mail: julien.ripoche@u-psud.fr@-
6 # Date: 20/08/15@-
7 -
8 """Command line to dump ROOT files contents to terminal"""@-
9 -
10 import cmdLineUtils@-
11 import sys@-
12 import importlib@-
13 -
14 -
15 def execute():@-
16 -
17     i = importlib.import_module("rootls-argparse")@-
18     parser = i.get_argparse()@-
19     # Put arguments in shape@-
20     sourceList, optDict = cmdLineUtils.getSourceListOptDict(parser)@-
21 -
22     # Process rootLs@-
23     return cmdLineUtils.rootLs(sourceList, oneColumn=optDict["oneColumn"], @-
24     longListing=optDict["longListing"], treeListing=optDict["treeListing"])@-
25 -
26 sys.exit(execute())@-
```

New rootls.py

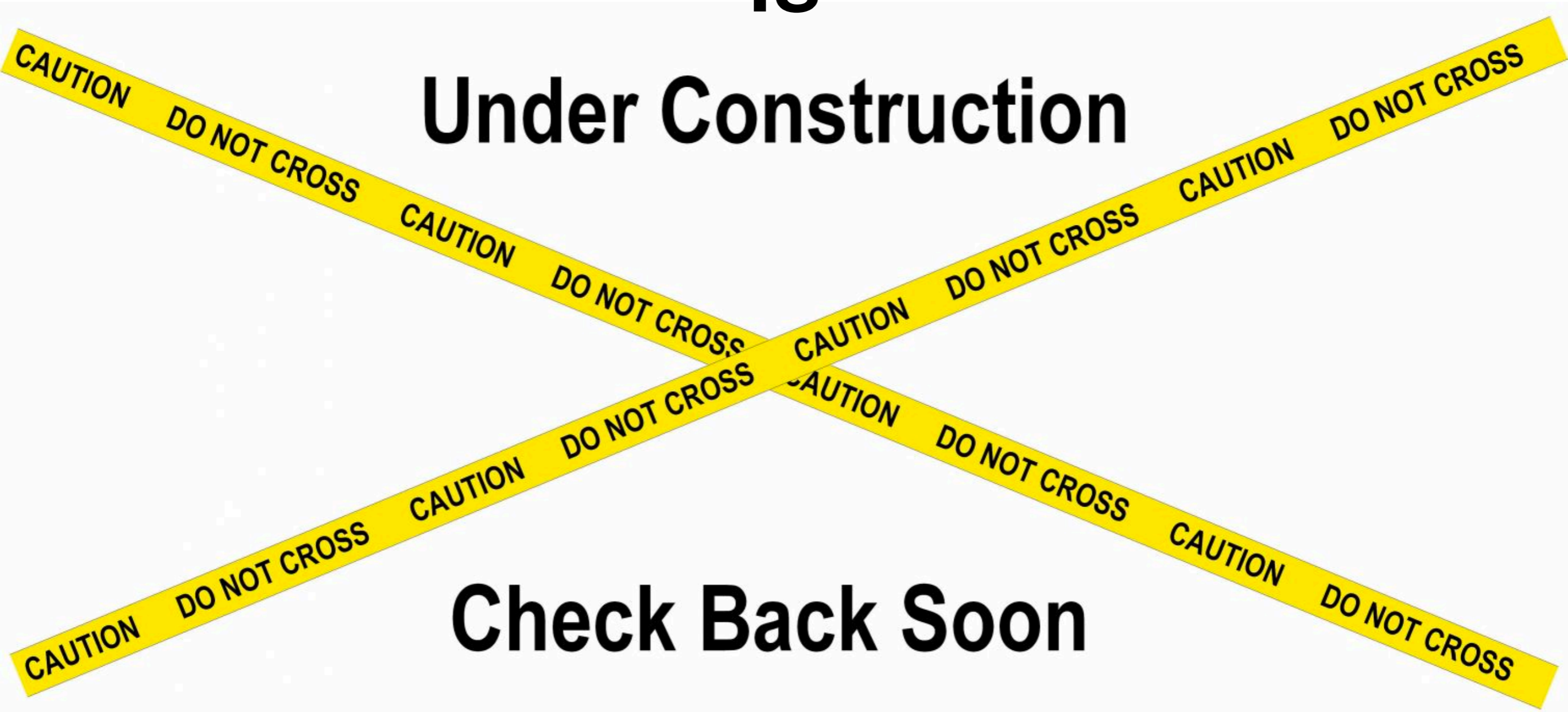
```
1 import argparse@-
2 import cmdLineUtils@-
3 -
4 # Help strings@-
5 description = "Display ROOT files contents in the terminal."@-
6 -
7 ONE_HELP = "Print content in one column"@-
8 LONG_PRINT_HELP = "Use a long listing format."@-
9 TREE_PRINT_HELP = "Print tree recursively and use a long listing format."@-
10 -
11 EPILOG = """Examples:@-
12 rootls example.root@-
13     . Display contents of the ROOT file 'example.root'.@-
14 -
15 rootls example.root:dir@-
16     . Display contents of the directory 'dir' from the ROOT file 'example.root'.@-
17 -
18 rootls example.root:*@-
19     . Display contents of the ROOT file 'example.root' and his subdirectories.@-
20 -
21 rootls file1.root file2.root@-
22     . Display contents of ROOT files 'file1.root' and 'file2.root'.@-
23 -
24 rootls *.root@-
25     . Display contents of ROOT files whose name ends with '.root'.@-
26 -
27 rootls -1 example.root@-
28     . Display contents of the ROOT file 'example.root' in one column.@-
29 -
30 rootls -l example.root@-
31     . Display contents of the ROOT file 'example.root' and use a long listing format.@-
32 -
33 rootls -t example.root@-
34     . Display contents of the ROOT file 'example.root', use a long listing format and print trees recursively.@-
35 """@-
36 -
37 def get_argparse():@-
38     parser = cmdLineUtils.getParserFile(description, EPILOG)@-
39     parser.prog = 'rootls'@-
40     -
41     parser.add_argument("-1", "--oneColumn", help=ONE_HELP, action="store_true")@-
42     parser.add_argument("-l", "--longListing", help=LONG_PRINT_HELP, action="store_true")@-
43     parser.add_argument("-t", "--treeListing", help=TREE_PRINT_HELP, action="store_true")@-
44     return parser@-
```

rootls-argparse.py

**CMAKE  
Is**

**Under Construction**

**Check Back Soon**



Bedankt, nanni, nandri, bayarlalaa, kiitos, dankie, dhanyavad, faafetai lava, rahmat, Баярлалаа, спасибо, mersi, 謝謝, ngiyabonga, tesekkür ederim, tapadh leat, xвала, asante, manana, obrigada, tenki, moichchakeram, mamnun, chokrane, murakoze, dзякую, djiere dieuf, go raibh maith agat, trugarez, dakujem, merce, мерси, shukriya, takk, arigatô, grazi, paldies, grazzi, mahalo, matondo, misaotra, dank je, welalin tack, barka, kria ora, mersu, vinaka, spas, благодарам, akun, dankon aciü, sulpáy, taiku, go raibh maith agat, sukriya, kop khun krap, taiaku, grazie, arigatô, takk, dakujem, trugarez, danjert, rahmet, tanemirt, najis tuke, didi, madioba, sagolun, mési, dekuji, sobodi, obrigado, enkosi, bayarlalaa, gracie, hvala, maururu, koszonom, chnorakaloutioun, gratias ago, gracies, namsi, kam sah hamnida, rahmat, tомаке хныбад, 감사합니다, xiexie, eucharistw, diolch, dhanyavadagalu, mersi