.help TTree

Desislava Nikolaeva Kalaydjieva[*]*, Supervisors: Olivier Couet, Axel Naumann*
CERN, Meyrin, Switzerland

Abstract
Improvement of the command ".help" in the ROOT software framework is needed. The process of the development of ".help scopeName" and ".help scopeName::member" are discussed. The command will be part of the next ROOT release v6.20 and it will offer a significant improvement for the users of ROOT.

Keywords
ROOT; ".help", reference guide, Doxygen, scopeName, member.

1 Project goal

ROOT is a software framework used by many scientists all over the world. It gives the user a great variety of tools and provides all the functionalities needed to deal with big data processing and scientific analysis. This variety determines the need for a detailed reference guide [1] and a help command for the interactive interface. The idea behind the project is to make the already existing command ".help" more useful by connecting it to the online reference guide. Currently, the ".help" command displays some useful information at the ROOT prompt. The extended command allows direct access from the ROOT prompt to the online reference guide generated by Doxygen [2] - a standard tool for generating documentation from annotated C++ sources. The original project was to introduce this new functionality for classes documentation only, but after reverse engineering of the Doxygen code was performed, it was possible to extend the functionality to methods defined in classes, enumeration, enumerators, namespaces, functions defined in namespaces, structs and data members.

Figure 1: The web page to be loaded if the user types ".help TTree" in the ROOT prompt.

2 .help scopeName

The original idea of the project consisted of extending the ".help" command to ".help className", Figure 1. Later, it was realized that namespaces and structs could be also included in the command in a similar way. The typical URL for a scope (class/namespace/struct) consists of five parts - two static ones (the

[*]Master student at Sofia University "St. Kliment Ohridski". Email: dnkalaydjieva@gmail.com.

beginning and the end of the URL), a part for the version of ROOT, a part for the type of the scope and one for the scope name, Figure 2. Three main tasks can be deduced from here:

1. Extraction of the scope name.
2. Building the URL.
3. Opening the URL in a web browser.

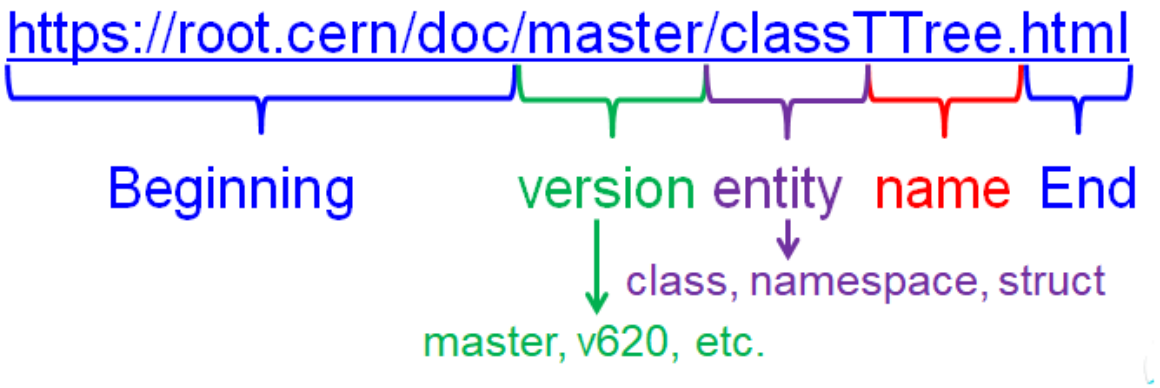

Figure 2: The URL structure for the class TTree.

2.1 Extraction of the scope name

The scope name has to be extracted from the command line. We save the command line into a TString. TStrings can be easily manipulated: cut, replace, compare, strip, preprend or append. First, we have to make sure that the user wants to call the extended command. We do that by checking if the command line starts with ".help " or ".? " with the function "BeginsWith", defined in TString. If this is not the case, an error message will appear in the ROOT prompt. Otherwise, we will have to remove the first six/three symbols and strip the command to extract the scope name.

2.2 Building the URL

As we already discussed, the URL consists of five parts. We need to find all of them and simply append them to build the URL. It is essential for the user to choose the most suitable version of the online reference guide. Thus we should build the URL based on the running ROOT version - which can be found thanks to "GetGitBranch" command. It could be "master", "v620" or any future version of ROOT. The scope type could be a class, a namespace or a struct. Currently we switch between the cases using enumerators (kUrlForClass, kUrlForNameSpace and kUrlForStruct). The fourth part of the URL consists of the scope's name that the user is looking for. We extract this name from the command line and modify it with respect to Doxygen's requirements - replace all "::" with "_1_1" and "_" with "__". The building of the URL is implemented in a new function called "UrlGenerator".

2.3 Opening the URL in a web browser

With the help of Sergey Linev, Axel Naumann and Bertrand Bellenot, a new function "OpenInBrowser" has been created. The function generates a command that opens a web browser for the Doxygen URL. It takes into account the operating system: it works for Windows, Linux and MacOS. The command is executed with the help of the shell command Exec. In the case of Linux operating system, there is also a check if the DISPLAY is set (with the shell command Getenv("DISPLAY")). If not, the user won't be able to open a web browser on that computer and a warning message will appear to let him know. Moreover the URL will be displayed in the prompt.

3 .help scopeName::member

The next step to improve the command is to extend it to ".help scopeName::member". The user will be able to search for a certain member from a scope. The URL to be loaded in a web browser is shown on Figure 3. The URL can be divided into two parts. The first part is for the scope and we already know how

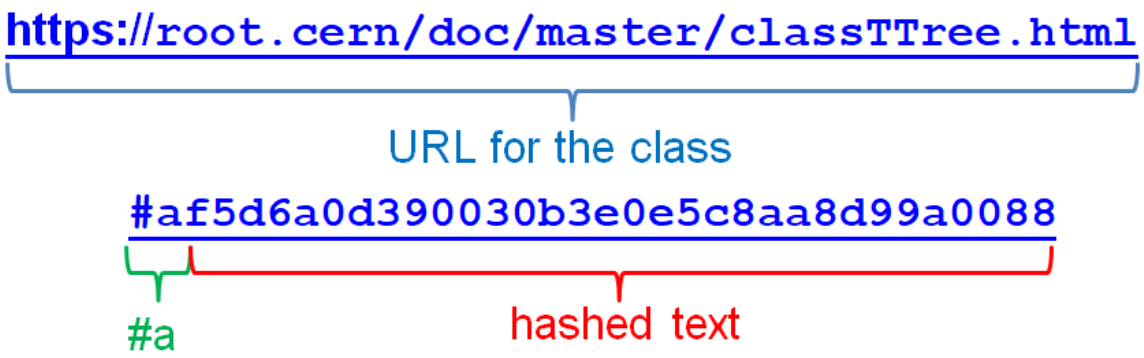

Figure 3: The URL structure for the method Branch form the class TTree.

to build it. The second part always starts with "#a" and then a hashed text is appended. To build such a URL we need to understand what is hidden behind the hashed text. In order to do that we have to perform Doxygen reverse engineering. With the help of Olivier Couet, the code of Doxygen was modified and debugged. The analysis shows that MD5 [3] hash algorithm is being used. MD5 is a message-digest algorithm, which produces a 128-bit hash value from initial text with different length. The use of MD5 assures that the second part of the URL will always have the same length for different members. The initial text differs for different members - methods, structors, enumeration, etc.

3.1 Methods and functions

The hashed text for member functions (methods in classes and functions in namespaces), defined in a scope, consists of "Return type ClassName::MemberNameMemberName(Member arguments)". We need to collect all needed parts in order to generate the hashed text for the URL. From the information given by the user in the command line we can extract the scope name and the member name. Thanks to the function "GetUnqualifiedName" even complicated cases like "ROOT::Fit::BinData::Range()" can be handled as the member is always the last to be written. The return type and the arguments of the member can be found by the use of ROOT's reflection data. These are functions giving information about the code of ROOT itself. "GetSignature()" and "GetReturnTypeName()" are used to retrieve the needed information. It is essential to understand that due to the process of hashing, it is required to have a very precise syntax for the initial text. Even a single missed symbol can lead to a different URL. The Doxygen syntax requirements will be discussed later.

3.2 Structors

The main difference between the build of the hashed text for structors (constructors/destructors) and methods is that Doxygen requires no return type for structors. ROOT's reflection data functions give us the class itself as a return type for a structor. Thus we have to take into account if the member is a method or a structor to build the correct URL. The user is searching for the constructor of the class if the name of the member and the name of the scope are the same. If the member name is "~scopeName", the user is looking for the destructor. We use enumerators to switch between the possible cases (kURLforMethod, kURLforStructor).

3.3 Enumeration and enumerators

Enumerations and structors have similar generation of the URL. No return type is appended and the hashed text is "ClassName::EnumerationEnumeration". The enumerator case is more complex. The URL consists of three parts as shown on Figure 4.

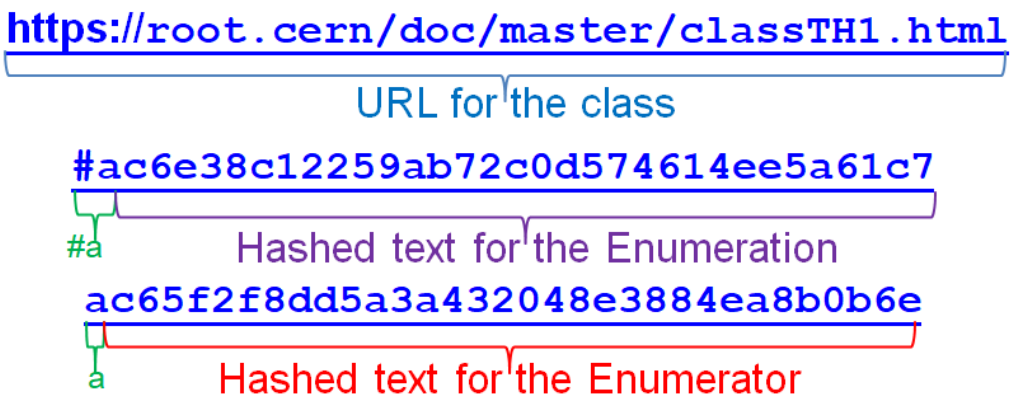

Figure 4: The URL structure for the enumerator kNormal from TH1 class.

The first part is for the scope and we generate it with "UrlGenerator". Then we append "#a" and a MD5 hashed text for the enumeration ("ScopeName::EnumerationEnumeration"). The last part of the URL is the symbol "a" and the MD5 hashed text for the enumerator, consisting of "@ ScopeName::EnumeratorEnumerator". We have to make sure the enumeration is not anonymous, thus Doxygen requires different syntax in this case: "scopeName::@№@№". Better investigation of the numbering scheme is needed. These features are implemented in a function, called "GetUrlForDataMember". Except for enumerators, the function generates also URLs for all other data members. Their URL is similar to the URL for methods. The hashed text consists of "Type ClassName::DataMemberNameDataMemberName(arguments)". We get the type with the help of "GetFullTypeName()".

4 Doxygen syntax requirements

Once we have collected all the needed information (return type, arguments, names) it is easy to generate a TString with the initial text for hashing. It is important to note that the text Doxygen requires is different from what ROOT reflection data gives us for the return type and the arguments. An example of that is the return type of "GetFriend" from "TTree" class. The return type given by ROOT is "TTree*", but Doxygen requires "TTree *". The spacing around the symbol "*" is different. Some other rules were found, after reverse engineering of the Doxygen code:

- Most of the special symbols require right alignment ("&" -> " &").
- The equal sign must be stripped.
- "std::" must be prepended to all stdlib classes.
- The scope name must be removed from enumeration.
- If the function is defined as inline, "*" in the return type has left alignment.
- If the function is inline and virtual, the word "virtual" has to be prepended to the return type.
- Not inline functions have right alignment of "*" in the return type.
- "constexpr" must be added in constexpr functions in namespaces.
- If the function is inherited from a base class, we need to use the name of the base class instead of the current class for the first part of the URL.

5 Results

The extended command ".help scopeName" will allow for direct access from the ROOT prompt to the online reference guide for all scopes (even for ones with complicated names like "ROOT::Fit::BinData"). In its current state, the command performs well for the majority of the methods. Is is not properly working for only a very small number of methods, due to the lack of information for the Doxygen requirements in those cases (anonymous enumeration) or due to some bugs in ROOT or Cling. However in those cases the URL for the scope will be opened in a web browser and the user will be able to search for the method in the web page. In the case of mistake from the user's side (wrong name of the method), the URL for the class will be displayed at the prompt. He will be able to open it and search by hand for the certain method or fix his mistake and open the URL for the method directly.

During the development of the command, some changes were made in ROOT. Ability to query whether function is defined as inline (kIsInlined by Enric Tejedor) or as constexpr (kIsConstExpr by Axel Naumann) were added due to the requirements of Doxygen. Some bugs in ROOT were found and fixed (by Axel Naumann) - for example there were missing variable arguments given by the ROOT reflection data. While testing the command, some methods with missing documentation in the reference guide were found. The documentation of the class "TClass" was not showing due to a small mistake in the documentation generation code, which was fixed by Olivier Couet. Of course, there are some possible future extensions (templates, global functions) and improvements (function overloads). However, the current state of the command has sufficient performance, Table 1.

Table 1: Testing the availability of the command for members of different scopes.

Scope	Available members	Unavailable members
TH1 (class)	302	24
TMath (namespace)	143	3

6 Conclusion

The existing ".help" command has been extended to allow for direct access to the Doxygen online reference guide. It was required for classes, but extended for methods, namespaces, enumerators, etc. The new feature will be part of the next ROOT release v6.20. The implementation of the improved ".help" command offers a significant usability improvement for the vast majority of high energy physicists.

Acknowledgements

During my time as summer student I have gained a lot of useful knowledge and fully enjoyed being a part of the ROOT team. I learned how to develop a new feature in the context of a large, existing C++ framework, to interact with all relevant parties, to collect user feedback, to debug and reverse engineer. I really enjoyed spending time on the computing side of physics. Thus I would like to acknowledge the support of my supervisors Olivier Couet and Axel Naumann. I am very thankful for their help and guidance throughout my project development. I would like to thank also Bertrand Bellenot, Enric Tejedor, Sergey Linev and the ROOT team in general. Thank you for being so welcoming, kind and always ready to help.

Bibliography

[1] https://root.cern/doc/master/index.html

[2] http://www.doxygen.nl/

[3] https://en.wikipedia.org/wiki/MD5