

EUROPEAN CENTRE FOR NUCLEAR RESEARCH, CERN

# ROOT Reference Documentation

by

Fuakye Eric Gyabeng

A Report submitted to CERN as part of the 2017 Summer Student Programme

in the  
EP-SFT Department

August 2017

EUROPEAN CENTRE FOR NUCLEAR RESEARCH, CERN

## *Abstract*

EP-SFT Department

by [Fuakye Eric Gyabeng](#)

A ROOT Reference Documentation has been implemented to generate all the lists of libraries needed for each ROOT class. Doxygen has no option to generate or add the lists of libraries for each ROOT class. Therefore shell scripting and a basic C++ program was employed to import the lists of libraries needed by each ROOT class.

# Contents

<b>Abstract</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
1.1 General Overview of ROOT . . . . .	1
1.1.1 Getting Started With ROOT . . . . .	2
1.1.2 Starting ROOT . . . . .	2
1.2 Motivation . . . . .	2
<b>2 Methodology</b>	<b>3</b>
<b>3 Results and Discussions</b>	<b>5</b>
<b>4 Conclusion</b>	<b>7</b>
 <b>Bibliography</b>	 <b>10</b>

# Chapter 1

## Introduction

### 1.1 General Overview of ROOT

In recent times, the ROOT software which is described as an object oriented program and library has helped many scientists all over the world in analyzing their data especially in areas related to physics research. ROOT was originally built for particle physics data analysis and it contains many features related to this field, but it is also used in other applications such as astronomy and data mining.

Also, ROOT is a free, open-source framework which is based on C++. This tool was developed at the European Organization for Nuclear Research, which is popularly known as CERN where physicists and engineers are probing the fundamental structure of the universe. It is interesting to know that at CERN, the world's largest and most complex scientific instruments are usually used by these scientists to study the basic constituents of matter - the fundamental particles.

The need to address the challenges faced by the experimental high-energy physics community was a motivating factor to the development of ROOT. [1] This is because scientists in the experimental high-energy physics community usually produce and analyze vast amounts of very complex data. For instance, the LHC experiments at CERN usually generates high amount of data (over 1,000 terabytes of data ) per year. [1] Therefore for users of ROOT and those who would like to use it for analyzing their data in the present or the future to do so effectively, there is the need to use and follow the Root Reference Guide. As the name suggests, the Root Reference Guide is an interpretation guide which is developed to give assistance to people using it. Hence, this project seeks to improve the ROOT Reference Guide developed at CERN by adding an essential feature.

### 1.1.1 Getting Started With ROOT

In order to start learning ROOT, the first thing to do is to have it installed on your system. Developers of ROOT has made it simpler in such a way that, it can be installed on most operating systems such as WINDOWS, LINUX, MAC and others.

Under the GNU Lesser General Public License[2] ROOT can be downloaded successfully from the ROOT download page[3]. The ROOT download page provides the installation instructions together with user's guide, complete class reference and lessons. A ROOT user forum [4] is also available to find answers to most challenging questions.

### 1.1.2 Starting ROOT

In Unix/Linux environment, ROOT can be started by issuing the following command from the command line:

```
$ROOTSYS/bin/root
```

\$ROOTSYS is the environment variable pointing to the directory where ROOT was installed. If the directory containing the ROOT executable is already in the system path, then one just needs to type "root" from command line to start ROOT. The ROOT logo should pop up regardless of the the operating system (as shown in Figure 1).



FIGURE 1.1: The Root logo

## 1.2 Motivation

The key motivation behind this project is the use of Linux shell scripts as well as basic C++ programming to add new functionality in implementing the ROOT Reference Documentation. This project also basically aimed at generating the various lists of libraries for each class in ROOT since Doxygen has no option in generating the the lists of libraries automatically.

## Chapter 2

# Methodology

This chapter emphasized on the methods and procedures that were deployed to include the various libraries needed by each class in the ROOT Reference Documentation. The methodology session highlighted on the possible techniques especially the programming codes used. Below is a detailed description of how the various lists of libraries were generated for each ROOT class.

- **ROOT installation**

The first step needed to build the documentation successfully is to have ROOT installed and running on any computer system. The installation of ROOT has been discussed earlier in the first chapter.

- **Run Cmake**

CMake is an open-source, cross-platform family of tools designed to build, test and package software. [5] CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice.

- **Modify the Doxyfile**

Basically, a Doxyfile describes the settings to be used by the documentation system. Here the key settings made in the Doxyfile was just setting the "COLLABORATION\_GRAPH = YES".

- **Run C++ codes and Shell Scripts**

C++ is an imperative procedural language whereas a shell script is a text file that contains a sequence of commands for a UNIX-based operating system. Basic C++ program and shell scripts were used to build the documentation. These codes can be seen at the appendix section.

- **Modify and Run the Makefile**

A Makefile consists mainly of variable definitions and rules which are instructions for remaking files or performing some other actions. Rules also consist of targets, dependencies and commands. The purpose of make is to simplify the process of rebuilding a program (i.e. a binary executable) from its source code, and to ensure that only source code that has been modified gets recompiled.

## Chapter 3

# Results and Discussions

In this project, the idea of modifying the Doxyfile as well as the Makefile and writing a basic C++ program with shell scripts has been used to implement the Root Reference Documentation used at CERN. This idea was used to basically include or add the various lists of libraries needed by each ROOT class. Although this idea was used, the concept of also modifying the filter.cxx which doxygen produces could also be used. Doxygen has no option to automatically generate the lists of libraries needed by each class in ROOT. Hence, shell scripts were written to include these lists of libraries.

Knowing the libraries needed by each class in ROOT helps to know more about the functionality in each class. For instance, the Math libraries in ROOT provides and supports a coherent set of mathematical and statistical functions. There are many classes in ROOT. Each class is governed by a set of libraries. Examples of ROOT classes include TArc, TH1, TNtuple, TArrayC etc. Images of the various lists of libraries needed by their corresponding ROOT classes were generated using dot files. The results of the libraries obtained are shown below for only four classes (ie. TArc, TH1, TNtuple, TArrayC). The remaining ones can be found in the documentation generated.



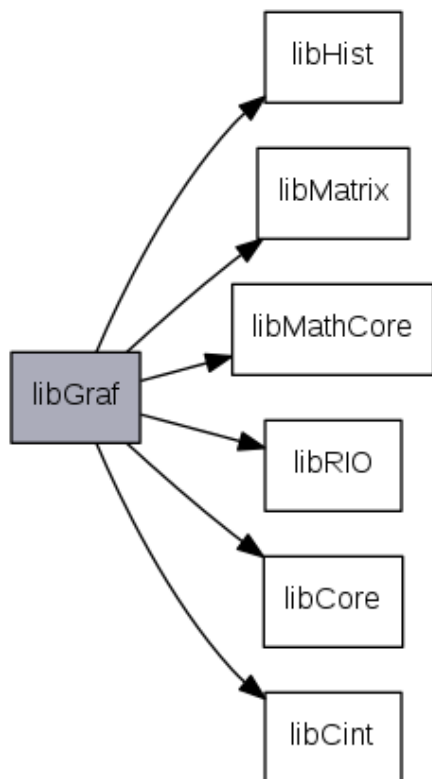


FIGURE 3.1: lists of libraries for TArc

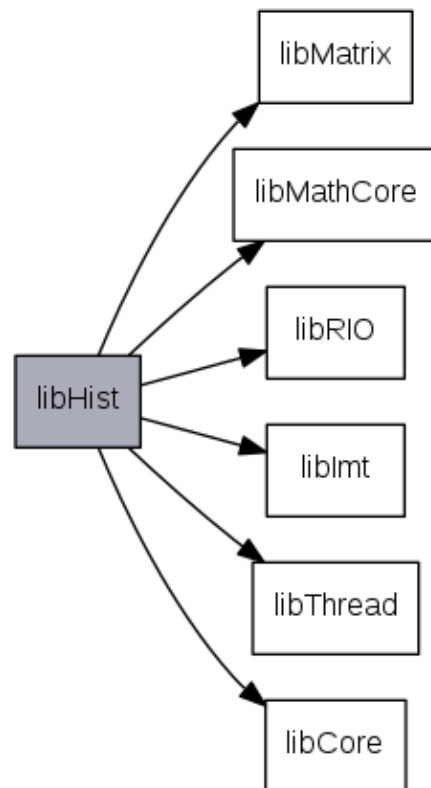


FIGURE 3.2: lists of libraries for TH1

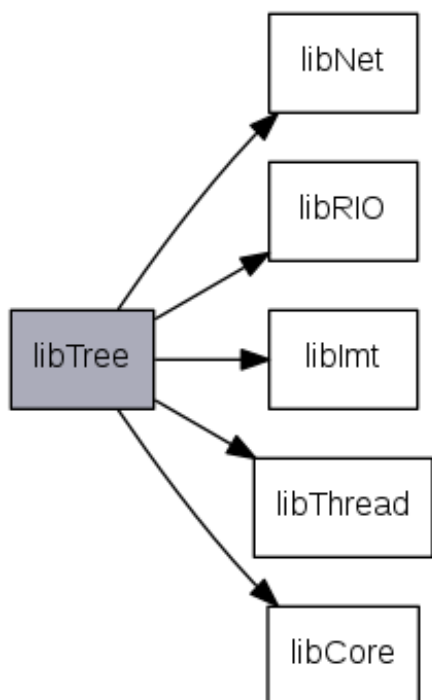


FIGURE 3.3: lists of libraries for TNtuple



FIGURE 3.4: lists of libraries for TArrayC

## Chapter 4

# Conclusion

The computations was successfully carried out and the outcome threw more light on the lists of various libraries for each class in ROOT. Generally, it is known in object-oriented programming that a class library is a collection of pre-written classes or coded templates, any of which can be specified and used by a programmer when developing an application program. The use of class libraries in ROOT makes users know the several functionalities needed for each ROOT class. Hence, it can be concluded that the use of these new techniques (writing a C++ program and shell scripts) has helped in generating the various lists of libraries needed for each ROOT class.

# Appendix

## C++ code and Shell Scripts for the Documentation

### 1. libs.C

---

```
void libs(TString classname)
{
    const char *libname;

    // Find in which library classname sits
    libname = gInterpreter->GetClassSharedLibs(classname.Data());

    if(!libname) return;

    // Print the library name in a external file
    TString mainlib = libname;
    mainlib.ReplaceAll(".so ", "");
    FILE *f = fopen("mainlib.dot", "w");
    fprintf(f, "    mainlib [label=%s];\n", mainlib.Data());
    fclose(f);

    // List of libraries used by libname via ldd on linux and otool on Mac

    gSystem->Exec(Form("$DOXYGEN_LDD $ROOTSYS/lib/%s | grep -v %s >
    libslist.dot", libname, libname));
}
```

---

### 2. listlib.sh

---

```
#!/bin/sh

# Finding the system we are running

export DOXYGEN_LDD="ldd"

OS=`uname `

case "$OS" in
    "Linux") export DOXYGEN_LDD="ldd"
    ;;
    "Darwin") export DOXYGEN_LDD="otool -L"
    ;;
```

```

esac

# Transform collaboration diagram into list of libraries

echo '#!/bin/sh' > listofclass.sh
echo '' >> listofclass.sh
grep -s "Collaboration diagram for" ${DOXYGEN_OUTPUT_DIRECTORY}/html/classT*.html | sed -e
"s/.html:.*$//" | sed -e "s/^.*html\\/class\\.\\.\\makelibs.sh /" >> listofclass.sh

chmod +x ./listofclass.sh
./listofclass.sh

```

---

### 3. makelibs.sh

---

```

#!/bin/bash

HTMLPATH=${DOXYGEN_OUTPUT_DIRECTORY}/html

sed -i'.back' -e 's/Collaboration diagram for /Libraries for /g' $HTMLPATH/class$1.html
rm $HTMLPATH/class$1.html.back

# Picture name containing the "coll graph"
PICNAME=$HTMLPATH/"class"$1"__coll__graph.svg"

# Find the libraries for the class $1
root -l -b -q "libs.C(\"$1\")"

sed -i'.back' -e "s/\\.so.*$/\";/\" libslist.dot
rm libslist.dot.back

# Generate the dot file describing the graph for libraries
echo "digraph G {" > libraries.dot
echo "    rankdir=LR;" >> libraries.dot
echo "    node [shape=box, fontname=Arial];" >> libraries.dot
cat mainlib.dot >> libraries.dot;
cat libslist.dot | grep -v "\.dylib" \
    | grep lib[A-Z] | sed -e "s/\\(.*\\)\\(lib.*;\\)$/    mainlib->\"\\2/\" \
    >> libraries.dot
echo "    mainlib [shape=box, fillcolor=\"\\#ABACBA\\", style=filled];" >> libraries.dot
echo "}" >> libraries.dot

# Generate the SVG image of the graph
dot -Tsvg libraries.dot -o $PICNAME

# Make sure the picture size in the html file the same as the svg
PICSIZE=$(grep "svg width" $PICNAME | sed -e "s/<svg //" '
sed -i'.back' -e "s/\\(.*__coll__graph.svg\\)\\( width.*\\)>\\)\\(.*$\\)/\\1 $PICSIZE>\\3/"
$HTMLPATH/class$1.html
rm $HTMLPATH/class$1.html.back

```

---

# Bibliography

- [1] ACAS Ravi Kumar and Arun Tripathi. Root: A data analysis and data mining tool from cern. In *E-Forum Committee*, page 90.
- [2] <https://root.cern.ch/root/License.html>, . Accessed: 31.07.2017.
- [3] <https://root.cern.ch/downloading-root>, . Accessed: 31.07.2017.
- [4] <https://root-forum.cern.ch/>, . Accessed: 31.07.2017.
- [5] Bill Hoffman, David Cole, and John Vines. Software process for rapid development of hpc software using cmake. In *DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2009*, pages 378–382. IEEE, 2009.