JOINT INSTITUTE FOR NUCLEAR RESEARCH
Veksler and Baldin Laboratory of High Energy Physics

FINAL REPORT ON THE SUMMER STUDENT PROGRAM

Development of software for analysis of femtoscopic correlation of K+K- pairs for pPb collisions in ALICE experiment

Supervisor:
Krystian Rosłon

Student:
Aleksander Szpakiewicz-Szatan,
Poland, *aleksander.szsz@gmail.com,*
Warsaw University of Technology

Participation period:
July 01 – August 25

Dubna, 2018

Contents

1 Abstract

This report consists of two main chapters.

Its first chapter revolves around Therminator2 software setup. In this chapter it is briefly explained what Therminator2 software is and what it depends on. Moreover it explains what probles it causes when user tries to set it up manually. Next solution of those problems is described and later there is brief summary of effect of my work.

The second chapter concerns Aligenqa software. In that chapter it is explained what Aligenqa is, what difficulties did its setup cause and how to use it. Later it is explained what did I do to its code within my task and how it lead me to modification of AliEn software. In the and of this chapter there is brief summary.

Next there is short conclusion to my internship. Later there is list of referenced materials. Following is acknowledgments section in which I thank people. In the and I attach source code of my installer script (with included comments).

2 Therminator2 setup

One of my tasks was preparation of working environment for other people. Many scientists need program for particle collision simulation. Program Therminator2 fulfils that need. However manual installation of this program is much more complicated (and time consuming) than setup of typical software. This is the reason why script for automatic setup must have been developed. The most popular (supported by Therminator2 and ROOT) operating system amoung the end users is Debian based Linux distro Ubuntu. That is why it was chosen as main operating system supported by installation script. As Ubuntu is based on Debian, it was very easy to modify script to support both systems.

2.1 What is Therminator2

Therminator2 (THERMal hevy IoN genrATOR 2) is program developed by Polish team consisting of Mikolaj Chojnacki, Adam Kisiel, Wojciech Florkowski and Wojciech Broniowski. Therminator2 is Monte Carlo event generator for studies of statistical of particles in relativistic heavy-ion collisions.

Therminator was originally written in order to perform hadronic freeze-out on simple boost-invariant hypersurfaces (such as Cracow single-freeze-out model or Blast-Wave model). However it was later refitted into more versatile tool for heavy-ion experiments. It may be adapted to analyze experimental data, model detectors or estimate results of experiments[1].

The most feasible way of providing that functionality was usage of CERN's ROOT framework, which is main dependency of Therminator2 (apart from C++ compiler).

2.2 Main depencency - ROOT

ROOT is a scientific software framework developed with modular approach. ROOT framework consists of mathematical libraries that are highly optimized for parallel processing of numerical data. Moreover ROOT libraries contain functions for data generation with various statistical distributions that may be used for simulation of wide variety of models. Furthermore ROOT software features include usage of standarized, tree-based compressed binary format which allows for efficient data storage. All of those features may be easily accessed by external programs as ROOT (while written in C++) supplies user with interfaces for various programming languages[3].

2.3 Problems related to Therminator2 setup

Because the last stable version of Therminator2 was published in 2011 the code became outdated. This leads to problems when using it in conjunction with modern hardware and software. Some parts of Therminator2 code became deprecated and later strictly forbidden by more recent C++ revisions (standards C++2011 and C++2017). Moreover standard C++ namespace *std* was in developer's enviroment used automatically (which is not the

case for unmodified environments). The documented requirements of Therminator2 are only ROOT software suite and C++ compiler, what is not complete list of it's requirements.

2.4 Solution to problems

Script was written in bash language (I used its manual[2] when I needed it), because it is supported by target OS text shell. When it executes script checks if user has root privilages (which are required to realise its task). Later it detects on what OS (distro and version) it is ran. If it is supported (ir is Debian 9 or Ubuntu 14, 16, 17 or 18) the program proceedes. If the used distro is correct, but version is not not supported (such as Debian 8 or Ubuntu 15) it informs user about it and suggests either usage of supported version or modification of script. So it does when distro is not supported (i.e. ARCH based distros such as Manjaro) or it cannot detect OS.

After determining the operating system script chooses the right version of ROOT to download (for Ubuntu precompiled binaries exist, Debian user is forced to do timeconsuming compilation). Next script downloads dependencies that are mostly required libraries and tools needed for proper compilation of ROOT (in case of Debian system) and Therminator2. Then script downloads proper version of ROOT to temporary directory (as shown on fig. 1). After download is finished if it is source installation (Debian) script compiles ROOT's code (after waring user about how time-consuming this process is and suggesting doing something else in meantime). After that (on Debian) or instead of that (on Ubuntu) the ready binaries are installed systemwide.

```
ROOT and Therminator2 have some dependencies. First let's update your database.
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Hit:2 http://pl.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://pl.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:4 http://archive.canonical.com/ubuntu bionic InRelease
Get:5 http://pl.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://pl.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [246 kB]
Get:7 http://pl.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [229 kB]
Get:8 http://pl.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [93.2 kB]
Get:9 http://pl.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [131 kB]
Get:10 http://pl.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [131 kB]
Get:11 http://pl.archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [58.6 kB]
Fetched 1,135 kB in 2s (491 kB/s)
Reading package lists... Done
You need C++ compiler, cmake, git and some minor utilities and need some dev libraries (with headers etc):
Reading package lists... Done
Building dependency tree
Reading state information... Done
cmake is already the newest version (3.10.2-1ubuntu2).
dpkg-dev is already the newest version (1.19.0.5ubuntu2).
g++ is already the newest version (4:7.3.0-3ubuntu2).
gcc is already the newest version (4:7.3.0-3ubuntu2).
libjpeg-dev is already the newest version (8c-2ubuntu8).
libx11-dev is already the newest version (2:1.6.4-3).
libxext-dev is already the newest version (2:1.3.3-1).
libxft-dev is already the newest version (2.3.2-1).
libxpm-dev is already the newest version (1:3.5.12-1).
python is already the newest version (2.7.15~rc1-1).
python-dev is already the newest version (2.7.15~rc1-1).
libtbb-dev is already the newest version (2017~U7-8).
binutils is already the newest version (2.30-20ubuntu2~18.04).
git is already the newest version (1:2.17.1-1ubuntu0.1).
libpng-dev is already the newest version (1.6.34-1ubuntu0.18.04.1).
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
=========================================
Downloading ROOT version 6.14.00
--2018-07-19 18:18:41--  https://root.cern.ch/download/root_v6.14.00.Linux-ubuntu18-x86_64-gcc7.3.tar.gz
```

Figure 1: Terminal output of running setup script.

When script finishes setting ROOT up it downloads Therminator2 source code. When it is ready it updates it's code (by inclusion of now mandatory line *using namespace std;*[1] in therm2_events.cxx file). Moreover it corrects order of linker flags in Therminator2's makefile, as they would fail to find ROOT libraries properly. It also removes code that creates Therminator2's documentation in LateX format (as it's support doesn't work corretly and it may still provide hypertext version of it). Next the script compiles Therminator2 (what is much quicker than ROOT compilation) and informs user how to run it.

When script finishes setting everything up, it removes ROOT's temporary files (as they take a lot of space up). Therminator2's source code is intentionally left intact, as user may need to check how it works.

If user uses command line switch *–manual*, the script downloads doxygen software and later compiles hypertext documentation for Therminator2. Other supported switches are: *-h* which prints usage information and *–list-supported-systems* that informs user which operating systems are supported.

In case of failure at any step, script informs user about it. Then it returns unique exit code, that may be used by external bash script (if end user needs further authomatization) to react properly. For programmer's convenience all of used exit codes are defined at beginning of script.

2.5 Summary

The script (after initial testing) was supplied to Alexey Aparin's group of students. For them I have prepared and presented a short presentation (for about 45 minutes) in which I explained them the informations about usage of my script and basic principals of Therminator2 usage. Alexey's group used my script and it prepared ROOT and Therminator2 environments for them with requirement of minimal effort on their behalf. It allowed his students to concentrate on their own tasks. They did not need to follow complicated manuals or debug outdated Therminator2's code. Their working environments were ready to use.

[1] This line informs compiler that code refers to standard library instructions and constant values without usage of *std::* prefix (i.e. *cout* instead of *std::cout*).

3 Implementing new particles to Aligenqa

Second subtask I was assigned to do was to enhance Aligenqa. My task was to make it possible for it to include in it's report output files more types of particles.

3.1 What is Aligenqa

Aligenqa is a tool for preparation of reports from experimental data.Originally it was based on tools developed by Christian Bourjau within the HMTF[2] and was later adapted as general generator-level QA in ALICE[**??**]. The tool itself is written in Python language and uses AliRoot macros.

3.2 Setup of environment

Before I was able to work on Aligenqa I needed to fulfill it's requirements. While ROOT and Therminator software required only free, widely available libraries and tools, Aligenqa was not that simple. In order to be able to use Aligenqa I needed to setup AliRoot and AliPhysics environment. Accessing their setup manuals on its own required CERN account. After CERN granted me access to their resources I was able to continue.

Setup of those two frameworks require user to have particular operating system (supported are only Ubuntu, CentOS, Mac OSX and Fedora). Because of this I have set virtual machine with Ubuntu up (as alternative solution with usage Docker[3] was resulting in errors).

Next I started setting up environment while following the manual[5]. While setting up AliPhysics dependencies I found out, that in order to download some of them I needed access to CERN's git repository. My account did not have privilage of accessing it. I needed to request access.

After being granted access to it I was able to continue. I spent few days on downloading the sourcecode (because of problems with Internet connection). When all dependencies were fullfilled I was able to build AliEn (ALIce ENvironment) software and use it.

While working with AliEn software I was forced to rebuild it as ROOT 6 is not fully backwards-compatible with ROOT5 (originaly I have built my environment as based on newest ROOT version, what lead me to programs not executing macros properly).

3.3 Usage of Aligenqa

In order to use Aligenqa user needs to have active AliEn session. Within that session user may use Aligenqa to download simulation data from CERN server (after authenticating with CERN creditentials) and to prepare reports out of this data. Aligenqa takes as input simulation data in root format, then it looks for specific data inside root file trees and than

[2]One of AliPhysics modules

[3]Docker is software that allows one Linux based distro pretend to be another. Simply it is leightweight alternative for virtualization (as it only virtualizes parts of code).

segregates it into intermediary root file from which it prepares summary - a pdf file that consists of segregated charts (which is shown in fig. 2).

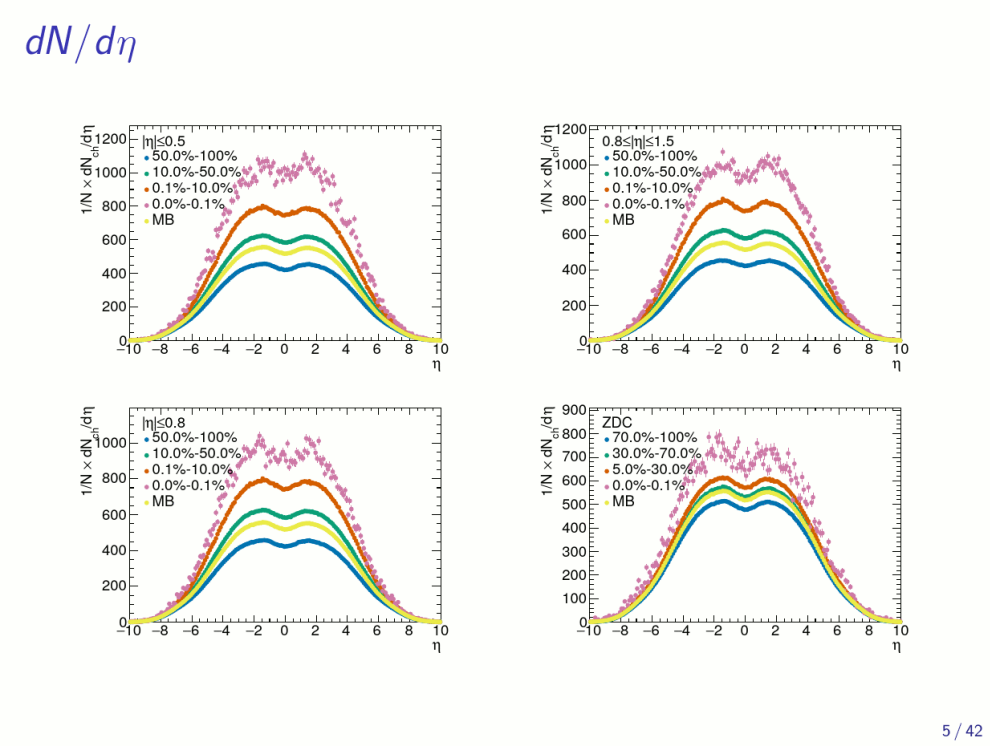

Figure 2: Example page of Aligenqa's output report.

If user tried inputting into Aligenqa file with unsupported data or which lacked some of root trees it would throw an error (without comment what has caused it). Moreover if user tried to download data that already was downloaded Aligenqa would refuse to work properly either. This is why I added basic error handling for it - each critical point of it's procedure will inform user what causes problem. Furthermore if error is not critical it would skip that particular step. In case of redownloading file would just use existing file, in case off missing data it would be omitted. Not only errors are more transparent to user, but also user may use Aligenqa with wider variety of input data (at least to a degree).

3.4 Adding new particle

While preparing output Aligenqa only searches for particular data, that is programmed in its code. In order for it to export data it needs to be added inside script file plotting.py. However while adding support for any particle is easy it requires this data to be included in supplied input file.

3.5 Modification of AliEn software

In order to supply Aligenqa with proper data I needed to modify existing AliEn software. Adding new particle into simulation requires editing its config file (and inclusion of that particle's parameters). However it would affect only raw output, AliEn software later performs operations on that data. In order to add support for new particles in those operations AliEn software code needs to be modified.

Main macro that needs to be modified in order to include partice support is easy to locate (as even Aligenqa documentation briefly mentions it). However that macro passes it's arguments into other macros. I've spent over a week trying to pinpoint function that is responsible for those data operations, unfortunatelly without success. Each change in code I made was followed by me running simulation in order to check whether my modifications were sufficient. Because of time constraints it was not possible to finish this part.

3.6 Summary

I was not able to fully test my modification of Aligenqa, because of lack of proper input data. However I have documented my findings and left simulation data for future usage in projects TWiki page [6]. This way if anyone would continue that project, this person has some grounds for furher work.

4 Conclusion

During my internship I have worked on software for benefit of other scientists. My script for working environment of ROOT and Therminator2 fulfills its objective, other people can set their working environments with minimal workload. Moreover a group of students were instructed how to use that software and did not need to guess which parts of its long manual are important for their work.

Aligenqa software was improved, now it works more reliably. Moreover if error happens within its execution - user will receive more verbose information on what caused it.

Part of AliEn software responsible for data generation for AliGenQA was (within time constraints) examined. Addition of new particles was partially solved. Furthermore effects of my work were documented and left for future usage at project's TWiki page [6].

5 References

1. Mikolaj Chojnacki, Adam Kisiel, Wojciech Florkowski, Wojciech Broniowski, *THERMINATOR 2: THERMal heavy IoN generATOR 2*, `https://arxiv.org/abs/1102.0273`

2. *Bash Reference Manual*, `https://www.gnu.org/software/bash/manual/html_node/index.html`

3. *ROOT v6.14 Reference Manual*, `https://root.cern.ch/doc/v614/`

4. *AliGenQA's README.MD file*

5. CERN *Building ALICE software: ALICE Analysis Tutorial*

6. Aleksander Szpakiewicz-Szatan *Software development for MC data analyzing*, `http://nica.fizyka.pw.edu.pl/do/view/Main/TeFeNica45`

6 Acknowledgements

I would like to thank my supervisor Mgr inż. Krystian Rosłon for giving me oppurtunity to work with his team and for his help. I would like to thank mgr inż. Marek Peryt for organisation of work of NICA Group that I have cooperated with and for all help I have received from him.

I would like to thank Joint Institute for Nuclear Research for allowing students such as me to take part in this internship and work within it's facilities. My internship here was great experience during which I have met awesome people and have broadened my horizons. Moreover I would like to thank Intitute for it's financial support without which participation in Summer Student internship would be severly impeded.

I would like to thank the Organising Committee: Elena Karpova, Elizabeth Tsukanova and Julia Rybachuk for their hard work and patience. Their help made it possible for me (and other students) to participate in this program.

I would also like to thank PhD Alexey Aparin for his cooperation, dr Łukasz Kamil Graczykowski for his help with creation of CERN account and Ms Susanna Safarova for her involvment during Russian language lessons.

7 Appendix - script's code with comments

```
#!/bin/bash

 ##############################################################################
#               Install ROOT and Therminator2 on Debian 9 or Ubuntu 14,16,17,18         #
#               Author: Aleksander Szpakiewicz-Szatan                                   #
#               04.07.2018                                                              #
#               contact: aleksander.szsz@gmail.com                                      #
 ##############################################################################

#########################################################################
# Constants and Variables preparation

# Constants
DOC="--manual"                          # command line switch to request manual compiltaion
SUP_OS="--list-supported-systems"       # command line switch to request printing list of
    operating systems
SHOWHELP="-h"                           # command line switch to request showing help
DEBIAN="debian"                         # codename for Debian
UBUNTU="ubuntu"                         # codename for Ubuntu
TEMP_DIR="/tmp"                         # temporary directory for ROOT install
ROOT_TARGET="/usr/local"                # install directory for ROOT
THERMINATOR2_TARGET="/opt"              # install directory for Therminator2 (it is
    installed in its subfolder)
START_DIR=`pwd`                         # remember start working directory (to get back to
    it later)

#######################################
# TABLE OF EXIT CODES
NO_ERROR=0
# SETUP ERRORS
        WRONG_PARAMETERS=1
        UNSUPPORTED_OS=2
        UNDETECTED_OS=3
        NON_ROOT_USER=4
# PREPARATION ERRORS
        APT_GET_UPDATE_FAIL=10
        APT_GET_INSTALL_FAIL=11
        APT_GET_DOXYGEN_FAIL=12
        CANNOT_CREATE_INSTALL_DIR=13
# ROOT ERRORS
        CANNOT_DOWNLOAD_ROOT=20
        ROOT_INST_FAIL=21
        ROOT_NOT_INSTALLED_PROPERLY=22
# THERMINATOR2 ERRORS
        THERMINATOR2_INST_FAIL=30
        THERMINATOR2_DOXY_FAIL=31
#######################################

# TABLE OF ROOT VERSIONS FOR VARIOUS OS VERSIONS
SOURCE_INSTALL="root_v6.14.00.source.tar.gz"
UBUNTU_18="root_v6.14.00.Linux-ubuntu18-x86_64-gcc7.3.tar.gz"
UBUNTU_17="root_v6.14.00.Linux-ubuntu17-x86_64-gcc7.2.tar.gz"
UBUNTU_16="root_v6.14.00.Linux-ubuntu16-x86_64-gcc5.4.tar.gz"
UBUNTU_14="root_v6.14.00.Linux-ubuntu14-x86_64-gcc4.8.tar.gz"

DISTRO=`awk "/^ID/␣{print␣$2}" /etc/os-release`                 # get OS ID from /etc/os-
    release
DISTRO=${DISTRO#*=}                                             # remove prefix ending with
    '='
DISTRO=`echo $DISTRO |awk '{print $1;}'`                        # get first word from string
OS_VERSION=`awk "/^VERSION_ID/␣{print␣$2}" /etc/os-release`     # get OS verion from /etc/os
    -release
```

```
OS_VERSION=${OS_VERSION#*\"}                                    # remove prefix ending with
    '"'
OS_VERSION=${OS_VERSION%.*}                                     # remove suffix starting
    with '.'

#########################################################################
# Checking CLI parameters

# Show help if user wants it (or doesn't know how to use this script)
if [[ $1 == $SHOWHELP || $# > 1 ]]; then
        echo "Usage␣$0␣\[$SHOWHELP\|$DOC\]"
        echo "This␣script␣will␣install␣ROOT␣in␣/usr/local␣and␣Therminator2␣in␣/opt"
        echo "The␣script␣need␣to␣be␣run␣with␣root␣privilages."
        echo "$DOC␣-␣additionally␣install␣doxygen␣and␣compile␣Therminator2␣documentation"
        echo "$SUP_OS␣-␣show␣list␣of␣supported␣OSes."
        echo "$SHOWHELP␣-␣shows␣text␣above."
        if [[ $1 == $SHOWHELP ]]; then          # if user asked for help it was desired
                exit $NO_ERROR                  # so return "NO_ERROR"
        else
                exit $WRONG_PARAMETERS          # else return "ERROR" as it was not desired
                    action
        fi
fi

# Show list of supported OSes
if [[ $1 == $SUP_OS ]]; then
        echo "Supported␣OSes:"
        echo "Debian␣9␣(Stretch)"
        echo "Ubuntu␣18.x"
        echo "Ubuntu␣17.x*"
        echo "Ubuntu␣16.x*"
        echo "Ubuntu␣14.x*"
        echo "*␣-␣not␣tested,␣but␣supposed␣to␣work."
        exit $NO_ERROR                          # return NO_ERROR
fi

# Is it supported Distro?
if [[ "$DISTRO" == "$DEBIAN" ]]; then   #If Debian - treat as Debian 9
        ROOT_FILE="$SOURCE_INSTALL"
        if [[ "$OS_VERSION" != "9" ]]; then #If not Debian, but not 9 - warn user
                echo "Warning,␣script␣tested␣only␣on␣Debian␣9␣(Stretch)␣version.">&2
        fi
elif [[ "$DISTRO" == "$UBUNTU" ]]; then         #If Ubuntu - check for versions
        if [[ "$OS_VERSION" == "18" ]]; then
                ROOT_FILE="$UBUNTU_18"
        elif [[ "$OS_VERSION" == "17" ]]; then
                ROOT_FILE="$UBUNTU_17"
        elif [[ "$OS_VERSION" == "16" ]]; then
                ROOT_FILE="$UBUNTU_16"
        elif [[ "$OS_VERSION" == "14" ]]; then
                ROOT_FILE="$UBUNTU_14"
        else
                echo "Unsupported␣Ubuntu␣version."
                echo "Supported␣versions␣are␣14.x,16.x,␣17.x␣and␣18.x."
                echo "Your␣is:␣$OS_VERSION."
                echo "This␣script␣was␣designed␣to␣on␣Debian␣9␣and␣those␣Ubuntu␣versions␣only
                    .␣"
                echo "You␣may␣modify␣it␣to␣run␣on␣other␣distros,␣but␣I␣can't␣guarantee␣it␣
                    will␣work␣(or␣be␣optimal)."
                exit $UNSUPPORTED_OS                    # return ERROR - it's
                    unsupported version of Ubuntu
        fi
else
        echo "Your␣OS␣is␣neither␣Debian␣or␣Ubuntu,␣it␣is:␣$DISTRO."
        echo "This␣script␣was␣designed␣to␣run␣on␣Debian␣or␣Ubuntu␣only.␣"
```

```
        echo "You may modify it to run on other distros, but I can't guarantee it will work
            (or be optimal)."
        exit $UNDETECTED_OS                                     # return ERROR - it's
            undetected OS
fi

#format ROOT_VERSION variable
ROOT_VERSION=${ROOT_FILE#*v}                                    # remove prefix ending with 'v'
if [[ "$ROOT_FILE" == "$SOURCE_INSTALL" ]]; then                # if using source install (like in
    Debian 9)
        ROOT_VERSION=${ROOT_VERSION%.s*}                        # remove suffix starting with '.s'
else                                                            # if using pre-compiled Ubuntu
    binary
        ROOT_VERSION=${ROOT_VERSION%.L*}                        # remove suffix starting with '.L'
fi

# Does user have root privilages?
if ! [ $(id -u) = 0 ]; then
        echo "The script need to be run with root privilages." >&2
        echo "Run $0 $SHOWHELP to get help." >&2
        exit $NON_ROOT_USER                                     # return ERROR - user does not have
            root privilages
fi

# Get username of user
if [ $SUDO_USER ]; then          # if user is sudoer
        REAL_USER=$SUDO_USER     # check for his real login from sudo
else                             # else
        REAL_USER=$(whoami)      # use whoami
fi

source /root/.bashrc             # use root's .bashrc (just to be sure that all enviromental
    variables are OK

# Get location of user's home directory using passwd
HOMEDIR=$( getent passwd "$REAL_USER" | cut -d: -f6 )

# Get number of CPU logical cores
CORES=`getconf _NPROCESSORS_ONLN`

#Does user want help files?
if [[ $1 == $DOC ]]; then
        MANUAL="1"
else
        MANUAL="0"
fi

################################################################################
# Prerequisities:
# Update apt database
echo "========================================="
echo "ROOT and Therminator2 have some dependencies. First let's update your database."
apt-get update

if [ $? -ne 0 ]; then   # did error occur in apt-get update?
        echo "Cannot update apt database. Check your internet connection. Is other process
            using application database?"
        exit $APT_GET_UPDATE_FAIL                               # return ERROR - apt-get update
            problem
fi

# Get dependencies
echo "You need C++ compiler, cmake, git and some minor utilities and need some dev libraries
    (with headers etc):"
apt-get install -y git dpkg-dev cmake g++ gcc binutils libx11-dev libxpm-dev libxft-dev
```

```
    libxext-dev libpng-dev libjpeg-dev python python-dev libtbb-dev

if [ $? -ne 0 ]; then   # did error occur in apt-get install?
        echo "Cannot␣install␣required␣libs.␣Check␣your␣internet␣connection.␣Is␣other␣process
            ␣using␣application␣database?"
        exit $APT_GET_INSTALL_FAIL                      # return ERROR - apt-get install
            problem
fi

# Get doxygen (if user want manual)
if [[ "$MANUAL" == "1" ]]; then
        echo "For␣documentation␣you␣need␣doxygen."
        apt-get -y install doxygen
fi

if [ $? -ne 0 ]; then   # did error occur in installing doxygen?
        echo "Cannot␣install␣doxygen.␣Check␣your␣internet␣connection.␣Is␣other␣process␣using
            ␣application␣database?"
        exit $APT_GET_DOXYGEN_FAIL                              # return ERROR - doxygen
            installation
fi

if [ ! -d "$THERMINATOR2_TARGET" ]; then
        echo "Directory␣$THERMINATOR2_TARGET␣does␣not␣exist."
        echo "Creating␣$THERMINATOR2_TARGET."
        mkdir $THERMINATOR2_TARGET
        if [ $? -ne 0 ]; then   # did error occur in creating install directory?
                echo "Could␣not␣create␣direcotry␣$THERMINATOR2_TARGET"
                exit $CANNOT_CREATE_INSTALL_DIR         # return ERROR - could not create
                    install directory (nor it exists)
        fi
fi
cd $TEMP_DIR

#############################################################################
# ROOT:
IS_ROOT=`which root`            # check for existing ROOT installationion
if [ -z "$IS_ROOT" ]; then      # if ROOT is not installed
        # Get ROOT
        echo "========================================="
        echo "Downloading␣ROOT␣version␣$ROOT_VERSION"
        wget -N https://root.cern.ch/download/$ROOT_FILE                # download ROOT
            archive, do not overwrite existing files if source is already downloaded (save
            bandwidth)
        if [ $? -ne 0 ]; then   # did error occur in downloading ROOT?
                echo "Unable␣to␣download␣ROOT␣:("
                exit $CANNOT_DOWNLOAD_ROOT                      # return ERROR - could not
                    download ROOT
        fi

        echo "Unpacking␣ROOT␣archive."
        tar xzf $ROOT_FILE                                      # unpack ROOT file

        #Choose the directory according to distro
        if [[ "$ROOT_FILE" == "$SOURCE_INSTALL" ]]; then
                ROOT_DIR="root-$ROOT_VERSION"                   # in case of source install
        else
                ROOT_DIR="root"                                 # in case of pre-compiled
                    install
        fi
        cd "$ROOT_DIR"

        # Compile on Debian
        if [[ "$ROOT_FILE" == "$SOURCE_INSTALL" ]]; then
                echo "Creating␣directory␣for␣ROOT␣build."       # prepare directory for ROOT
```

```
                    install
                sudo -E -u $REAL_USER mkdir obj
                cd obj
                echo "Configuring␣ROOT␣installation."
                sudo -E -u $REAL_USER cmake ..                  # compile with REAL_USER
                    privilages
                echo "Compiling␣ROOT␣with␣use␣of␣all␣available␣CPU␣cores,␣it␣may␣take␣a␣
                    while␣(~1h␣on␣i7␣4720HQ)."
                echo "Take␣a␣break,␣go␣for␣a␣walk␣or␣something..."
                cmake --build . --target install -- -j$CORES    # compile and install ROOT

                if [ $? -ne 0 ]; then           # did error occur in compilation or
                    installation of ROOT?
                        echo "ROOT␣installation␣failed."
                        exit $ROOT_INST_FAIL                    # return ERROR - could not
                            compile/install
                fi

                cd ..
        else
        #In case of Ubuntu move ROOT to correct dir
                echo "Installing␣ROOT␣in␣$ROOT_TARGET."
                chmod -R o+r *                                  # allow execution of ROOT
                    libraries
                chown -R root:root *
                cp -R -f * "$ROOT_TARGET"                       # copy files to ROOT_TARGET
                    directory
                cd "$ROOT_TARGET"

        fi

        IS_ROOT=`which root`            # check if ROOT is properly detected
        if [ -z "$IS_ROOT" ]; then      # if it isn't detected
                echo "ROOT␣instalation␣failed␣(ROOT␣is␣not␣detected)."
                exit $ROOT_NOT_INSTALLED_PROPERLY                       # return ERROR -
                    ROOT not properly installed
        else
                echo "ROOT␣instalation␣finished."
                # Clean up leftowers from temporary directory
                echo "Cleaning␣up␣$TEMP_DIR."
                rm -r $TEMP_DIR/root*
        fi
else    # if ROOT already installed - inform the user and skip it's installation
        echo "ROOT␣already␣installed."
fi



##############################################################################
# Therminator2:

# Download Therminator2:
cd "$THERMINATOR2_TARGET"
echo "========================================="
echo "Preparing␣directory␣for␣Therminator2"
mkdir therminator2
cd therminator2/
echo "Downloading␣latest␣version␣of␣Therminator2"
wget -N http://therminator2.ifj.edu.pl/therminator2-latest.tar.gz       # download
    Therminator2 archive, do not overwrite existing files if source is already downloaded (
    save bandwidth)
tar xzf therminator2-latest.tar.gz                                      # unpack
    Therminator2 files

# Fix a bug in source code (by inserting missing line of code)
```

```
echo "Adding␣missing␣line␣\"using namespace std;\"␣in␣therm2_events.cxx"
FILE='./build/src/therm2_events.cxx'
sed -i '/main/ i using namespace std;' $FILE    # look for line containing keyword main (int
     main()) and insert "using namespace std;" just above it

# Fix bugs in Makefile (by replacing three lines that contain wrong order of parameters and
    by ommiting lines providing Latex doxygen support)
FILE1='./Makefile'
FILE2='./Makefile.tmp'
FROM='$(LD) $(LFLAGS) $^ -o $@'
TO='$(LD) $^ -o $@ $(LFLAGS)'
echo "Correcting␣errors␣in␣$FILE␣from:"
echo "$FROM"
echo "to:"
echo "$TO"
mv $FILE1 $FILE2
sed -n '1,118p' $FILE2 >> $FILE1
printf '\t%s\n' "$TO">>$FILE1
sed -n '120,122p' $FILE2 >> $FILE1
printf '\t%s\n' "$TO">>$FILE1
sed -n '124,126p' $FILE2 >> $FILE1
printf '\t%s\n' "$TO">>$FILE1
echo "Disabling␣buggy␣Latex␣support␣in␣$FILE."
sed -n '128,141p' $FILE2 >> $FILE1
sed -n '144,146p' $FILE2 >> $FILE1
sed -n '148,169p' $FILE2 >> $FILE1
rm $FILE2

# Compile Therminator2
echo "Compiling␣Therminator2,␣it␣shouldn't␣take␣more␣than␣2-3␣minutes."
make -j $CORES

if [ $? -ne 0 ]; then   # did error occur in Therminator2 compilation?
        echo "Therminator2␣compilation␣failed."
        exit $THERMINATOR2_INST_FAIL            # return ERROR - Therminator2 compilation
            failed
fi

# Compile Therminator2 documentation
if [[ "$MANUAL" == "1" ]]; then
        echo "Compiling␣documentation␣for␣Therminator2"
        make doc -j $CORES
fi

if [ $? -ne 0 ]; then   # did error occur in Therminator2's documentation compilation?
        echo "Therminator2's␣documentation␣compilation␣failed."
        exit $THERMINATOR2_DOXY_FAIL            # return ERROR - Therminator2's
            documentation compilation failed
fi

THERMINATOR2_PATH=`pwd` # Get Therminator2 path

cd "$THERMINATOR2_TARGET"                       # Return to directory where user started
    script

chown -R $REAL_USER:$REAL_USER "therminator2"   # change ownership of Therminator2 directory
     (and files) to non-root user

##########################################################################
# Finish:
echo "========================================="
echo "To␣run␣ROOT␣just␣type␣\"root\"␣in␣terminal."
echo "To␣get␣info␣how␣to␣use␣Therminator2␣run␣\"$THERMINATOR2_PATH/runall.sh -h\"."
echo "Thank␣you␣for␣using␣this␣script."
```

```
exit $NO_ERROR  # return NO_ERROR
```