

# Genealogical Tree

0.0.1\_5702150

Generated by Doxygen 1.8.9.1

Wed Jul 1 2015 01:01:09



# Contents

<b>1</b>	<b>Genealogical Tree</b>	<b>1</b>
<b>2</b>	<b>Sources</b>	<b>5</b>
<b>3</b>	<b>Tests</b>	<b>7</b>
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>File Documentation</b>	<b>11</b>
5.1	src/main.cpp File Reference . . . . .	11
5.1.1	Function Documentation . . . . .	11
5.1.1.1	main . . . . .	11
5.2	src/version.h File Reference . . . . .	11
5.2.1	Detailed Description . . . . .	12
5.2.2	Macro Definition Documentation . . . . .	12
5.2.2.1	DEFINE_VERSION . . . . .	12
	<b>Index</b>	<b>13</b>



# Chapter 1

## Genealogical Tree

### Summary

Program should be able to **find all the descendant with name Bob for all the ascendants with name Will on any level of ancestry**. In order to present the capabilities of your app:

- implement the application to optimize the initialization time
- application should have built in data about genealogical tree of people living in particular country
- please generate a representative data that has sample people and relationships between them. Use all varieties of names (can be also generated) but also put two test names (Bob and Will) and connect them in different relationships.
- the application should possess tests that are checking possible edge cases and ensure the stability of the application.
- the designed data structure should ensure optimized search time on following fields: name, last name, date of birth and location.

### Expected installed software

A Modern C++ GNU compiler, *g++ 4.9.2 or above, and a recent cmake*, 3.1 or above, are the minimum for binaries. As well a valid *boost* library is supposed to be installed.

Regarding to documentation, *doxygen*, *latex*, *graphviz* and *plantuml.jar* are needed. For example, if you work with Xubuntu 15.04 or its **Docker** equivalent, the following commands might do the trick for you:

```
sudo apt-get -y install git build-essential libboost-all-dev doxygen doxygen-latex openjdk-8-jdk graphviz
sudo add-apt-repository -y ppa:george-edison55/cmake-3.x
sudo apt-get -y update
sudo apt-get -y install cmake
sudo apt-get -y upgrade
sudo mkdir /opt/plantuml && sudo chmod a+wr /opt/plantuml
wget http://sourceforge.net/projects/plantuml/files/plantuml.jar/download -O /opt/plantuml/plantuml.jar
```

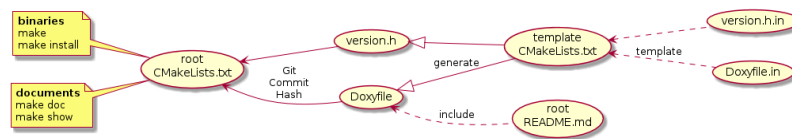
For other O.S., have a look to [Homebrew](#) or [Git/MinGW](#)

### Generate binaries & documentation

Usual commands:

```
mkdir build
cd build
cmake ..
```

```
make
make install
make doc
```



**Note:** If you happen to work with *OSX* and *Homebrew*, don't forget to invoke *cmake* pointing to the **GNU** compiler:

```
cmake -DCMAKE_CXX_COMPILER=g++-5 ..
```

**Note:** If you happen to work with *Windows* and *Git/MinGW*, don't forget to invoke *cmake* pointing to the **GNU** generator:

```
cmake -G "MSYS Makefiles" ..
```

As well a script, called **show** or something similar, will be created in your *home* directory as a shortcut for generating & viewing documentation. Don't hesitate to use it as a *template* for your specific environment.

## Generate only documentation

Similar commands to the previous ones:

```
mkdir build
cd build
cmake -DONLY_DOC=TRUE ..
make doc
```

**Note:** If you happen to work with *Windows* and *Git/MinGW*, don't forget to invoke *cmake* pointing to the **GNU** generator:

```
cmake -G "MSYS Makefiles" -DONLY_DOC=TRUE ..
```

## Development details

In order to generate binaries & documentation, the following versions were used:

For code

Pay attention to *cmake* and *gcc* versions. A minimum is required to work on several O.S. using modern C++. Feel free to locally hack **CMakeLists.txt** to meet your needs.

\*Linux\* ( Xubuntu 15.04 )

- **cmake 3.1.3**
- **gcc 4.9.2**
- **boost 1.55**

\*OSX\* ( Yosemite 10.10.3 )

- **cmake 3.2.2**
- **gcc 5.1**
- **boost 1.58**

**Note:** If you happen to work with *OSX* and *Homebrew*, don't forget to compile **boost** with the previous **gcc** compiler, not with the default *clang* one:

```
brew install gcc
brew install boost --cc=gcc-5
```

**\*Windows\* ( Win7 x64 )**

- **cmake 3.3.0**
- **gcc 5.1**
- **boost 1.58**

**For documentation**

Environment variables to locate PlantUML *jar* and default *PDF* viewer can be defined to overwrite default values. See **CMakeLists.txt** for further information on your platform.

**\*Linux\***

- **doxygen 1.8.9.1**
- **latex/pdfTeX 2.6-1.40.15**
- **graphviz/dot 2.38.0**
- **java/plantuml 1.8.0\_45/8026**

**\*OSX\***

- **doxygen 1.8.9.1**
- **latex/pdfTeX 2.6-1.40.15**
- **graphviz/dot 2.38.0**
- **java/plantuml 1.8.0\_40/8026**

**\*Windows\***

- **doxygen 1.8.9.1**
- **latex/pdfTeX 2.9.5496-1.40.15**
- **graphviz/dot 2.38.0**
- **java/plantuml 1.8.0\_45/8026**

**Note:** Don't forget configure *Doxyfile* and *CMakeLists.txt* to use **README.md** as *Main Page* for **latex** documentation.

**For IDE**

To use **NetBeans** don't forget to configure a *cmake* project with *custom build* folder. Add at that moment any extra customization in the command line used by *cmake* instruction. For example:

- **-DCMAKE\_CXX\_COMPILER=g++-5 for OSX**
- **-DONLY\_DOC=TRUE** for only documentation on **Linux/OSX**
- **-G "MSYS Makefiles" for Windows**
- **-G "MSYS Makefiles" -DONLY\_DOC=TRUE** for only documentation on **Windows**

**Note:** If you happen to use *jVi* plugin on *OSX*, don't forget to use **\*\*lc\*\*** instead of just **\*\*c\*\*** for its *\*/bin/bash\** flag.

## GIT Commit Hash

In order to add the specific **git commit hash** into code & documentation, *templates* are defined in the *template* folder for **Doxyfile** & **version.h** files.





## Chapter 2

# Sources

Source folder for headers & code files.

### Generate Files

[version.h](#) is generated with *GIT* information



## Chapter 3

# Tests

Future folder with *boost* test cases



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

src/ <a href="#">main.cpp</a>	11
src/ <a href="#">version.h</a>	11



## Chapter 5

# File Documentation

### 5.1 src/main.cpp File Reference

```
#include <iostream>
#include <utility>
#include <algorithm>
#include <boost/graph/graph_traits.hpp>
#include <boost/graph/adjacency_list.hpp>
#include <boost/graph/dijkstra_shortest_paths.hpp>
#include "version.h"
```

#### Functions

- int `main` (int argc, char \*\*argv)  
*Main function.*

#### 5.1.1 Function Documentation

##### 5.1.1.1 int main ( int argc, char \*\* argv )

Main function.

#### Parameters

<i>argc</i>	An integer argument count of the command line arguments
<i>argv</i>	An argument vector of the command line arguments

#### Returns

an integer 0 upon exit success

### 5.2 src/version.h File Reference

#### Macros

- #define `DEFINE_VERSION_FIRST` "0"
- #define `DEFINE_VERSION_MIDDLE` "0"
- #define `DEFINE_VERSION_LAST` "1"

- `#define DEFINE_GIT_DETAILS "5702150 (HEAD, origin/master, origin/develop, origin/HEAD, develop) Fix some conflits with some img/pdf files"`
- `#define DEFINE_GIT_COMMIT_HASH "5702150"`
- `#define DEFINE_VERSION`

## Variables

- static const char \* **VERSION** = "VERSION = " DEFINE\_VERSION
- static const char \* **GIT\_DETAILS** = "GIT\_DETAILS = " DEFINE\_GIT\_DETAILS

### 5.2.1 Detailed Description

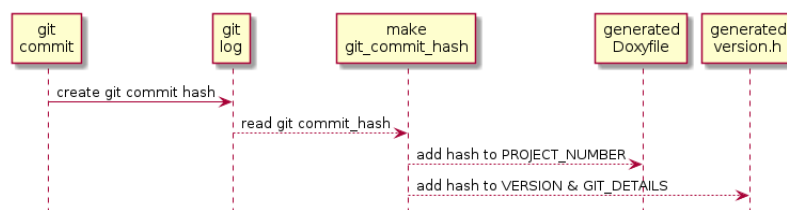
This metadata information might be located through **strings** command

- Linux/Solaris/Mac:

```
strings <binary> | grep VERSION
strings <binary> | grep GIT_DETAILS
```

- Windows (MinGW):

```
strings <binary> | findstr VERSION
strings <binary> | findstr GIT_DETAILS
```



### 5.2.2 Macro Definition Documentation

#### 5.2.2.1 #define DEFINE\_VERSION

**Value:**

```
DEFINE_VERSION_FIRST "." \
    DEFINE_VERSION_MIDDLE "." DEFINE_VERSION_LAST \
    "_" DEFINE_GIT_COMMIT_HASH
```



# Index

DEFINE\_VERSION  
version.h, [12](#)

main  
main.cpp, [11](#)  
main.cpp  
main, [11](#)

src/main.cpp, [11](#)  
src/version.h, [11](#)

version.h  
DEFINE\_VERSION, [12](#)