Draft_0.02

Browser & Refactor

Abidos C++

User Manual

O(n)

Fructu

Abidos C++ User Manual

COLLABORATORS			
	TITLE : Abidos C++ User Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	fructu	November 2012	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.02	November 2012	Draft version starting to write this documentation:	Fructu

Contents

List of Figures

Chapter 1

Introduction

1.1 Why this project ?

When you start a new project is good to start with a set of UML diagrams, but that is not always possible or is a hard task to do those diagrams with a tool.

Abidos extract the information of C++ headers and source code; and generates those diagrams for you fast and easily.

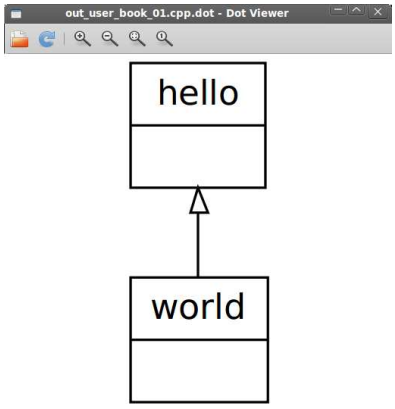
Example 1.1 Cpp Showing mode

With a piece of code like this:

```
class hello
{
};

class world: hello
{
};
```

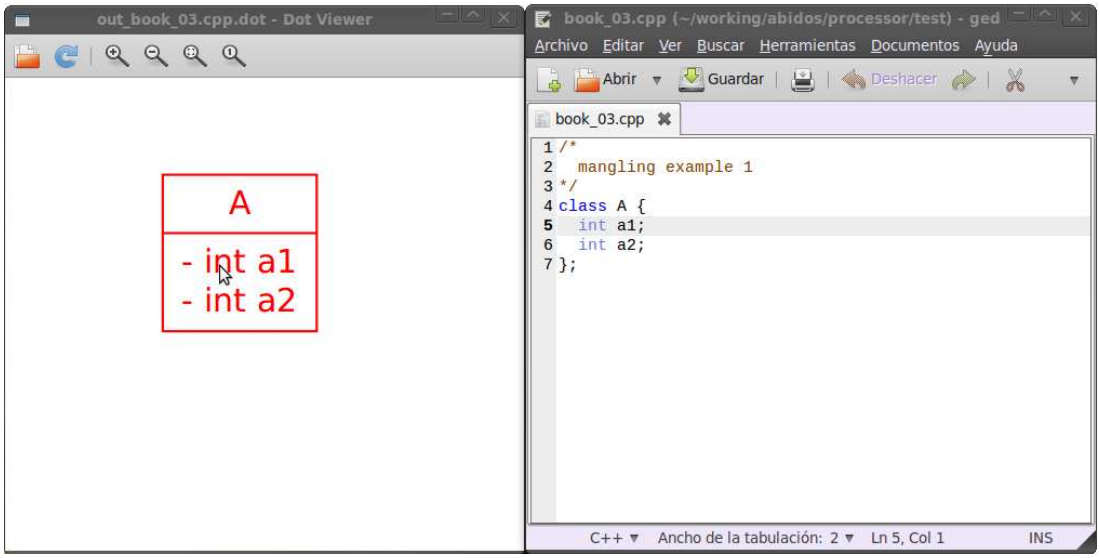
Abidos process it and will show you:



Browse code is sometimes a boring task, with abidos you can do it easily; Abidos generates UML diagrams, you can click into an attribute or method that you want to see or edit, and Abidos will open an editor with that piece of code.

Example 1.2 Cpp Browsing code

You can click in the a1 method of A class node and Abidos will open an editor and put the cursor in that line:



1.2 But there is another projects like doxygen . . .

Yes doxygen is a superb project, i started Abidos because i wanted to have other goals:

- Doxygen is designed to generate documentation, Abidos is about interacting with code.
- In future versions of Abidos Code refactoring code will be available.
- I was interested in the development of a C++ parser; is good do something like that.
- I would like to write other versions of abidos for other languages, i have a prototype of Abidos python project, for now Abidos_cpp is more active.
- I want have a project that i can add a new back-ends generator to do meta-programing for example generate sql interface (a class with methods to insert and modified information) from a struct declaration.
- Abidos use a descent parser and use backtracking , for now i am happy with that architecture but i have to investigate it more deeply.

Chapter 2

Installation

2.1 Get the project

You can obtain the project from github by 2 ways.

- clone the code with git:

```
git clone git@github.com:fructu/abidos_cpp.git
cd abidos_cpp/processor/
```

or you can

- download a zip package:

```
wget "https://github.com/fructu/abidos_cpp/archive/master.zip"
unzip master.zip
cd abidos_cpp-master/processor/
```

2.2 Make project and install

we will make the processor part of Abidos is the real core of Abidos C++:

```
cmake .
make VERBOSE=1 &> make_out.txt ❶
sudo make install
```

❶ Abidos will need this file **make_out.txt** to know what files have to parse (Abidos can parse Abidos because is a C++ Project).

you can test it with the abidos itself:

```
abidos_make_process.pl ❶
```

❶ this script do all the toolchain that abidos need for you.

This command encapsulates the toolchain of Abidos (it will be explained in next chapters).

abidos_cpp is installed in /opt/abidos_cpp directory. == Abidos parsing

2.3 The easy way passing one C++ file

If your entire C++ project is only one file you can show his UML diagram in this way.

```
abidos_cpp --out_dir . <file.cpp>
xdot_run.py files_output.dot
```

Some projects need pass the directories where includes are :

passing includes directory

```
abidos_cpp --includes includes/ --out_dir . t034.cpp
```

Note
The includes issue is like when you pass it to gcc or g+ compiler, see how you compile your project and you will know how invoke Abidos as well.

2.4 The easy way parsing a set of C++ files

Normally you will want parse all the files of a whole C++ project.

In order to parse a whole project you will need a make process with his Makefile.

Within Abidos package you have installed **abidos_make_process.pl**, this script analyzes the output of a **make process**, is the most straight method to obtain a UML diagram browsable.

It is possible to do this operations by hand, but first we should learn the easy way.

The easy way to execute Abidos is enter in a C++ project directory, and do:

```
cd <cpp_project>
make clean
make VERBOSE=1 &> make_out.txt ❶
abidos_make_process.pl
```

❶ Abidos needs to know what files needs to parse, where are the includes files , all those stuff of a project are in the output of a make process, in the next section you will see how **abidos_make_process.pl** do it.

2.5 The hacker way Abidos toolchain explanation

When you are going to parse a C++ project with Abidos, you need a special directory called **.abidos_cpp/** in the same directory you have the **Makefile**.

In order to work properly and parse successfully a C++ project abidos need be loaded with the file **.abidos_cpp/files_input**, in this file each line is like:

```
/home/.../src/main.cpp:/home/.../includes:/home/.../src
/home/.../src/file1.cpp:/home/.../includes:/home/.../src
```

Note
"..." is to do not make more bloat the example.

Those lines begin with a cpp file and are followed by some directories separated by **:** symbol in this directories is where abidos search for includes files.

Once abidos have parsed the whole project:

- All the files in **.abidos_cpp/files_input**.
- The files includes in those cpp files.

It generates **.abidos_cpp/files_output.dot** which is a file in graphviz format. [1]

Now with **xdot_run.py** based in xdot.py project [2] you can look **.abidos_cpp/files_output.dot** graphviz format.

you will see a navigable window with a UML diagram of the C++ project.

abidos_make_process.pl would do all this things for you, but for now there is a lot of remaining work to do in this script:

- Analyzes **make_out.txt** and generates **.abidos_cpp/files_input**, it works when the output of make process is like:

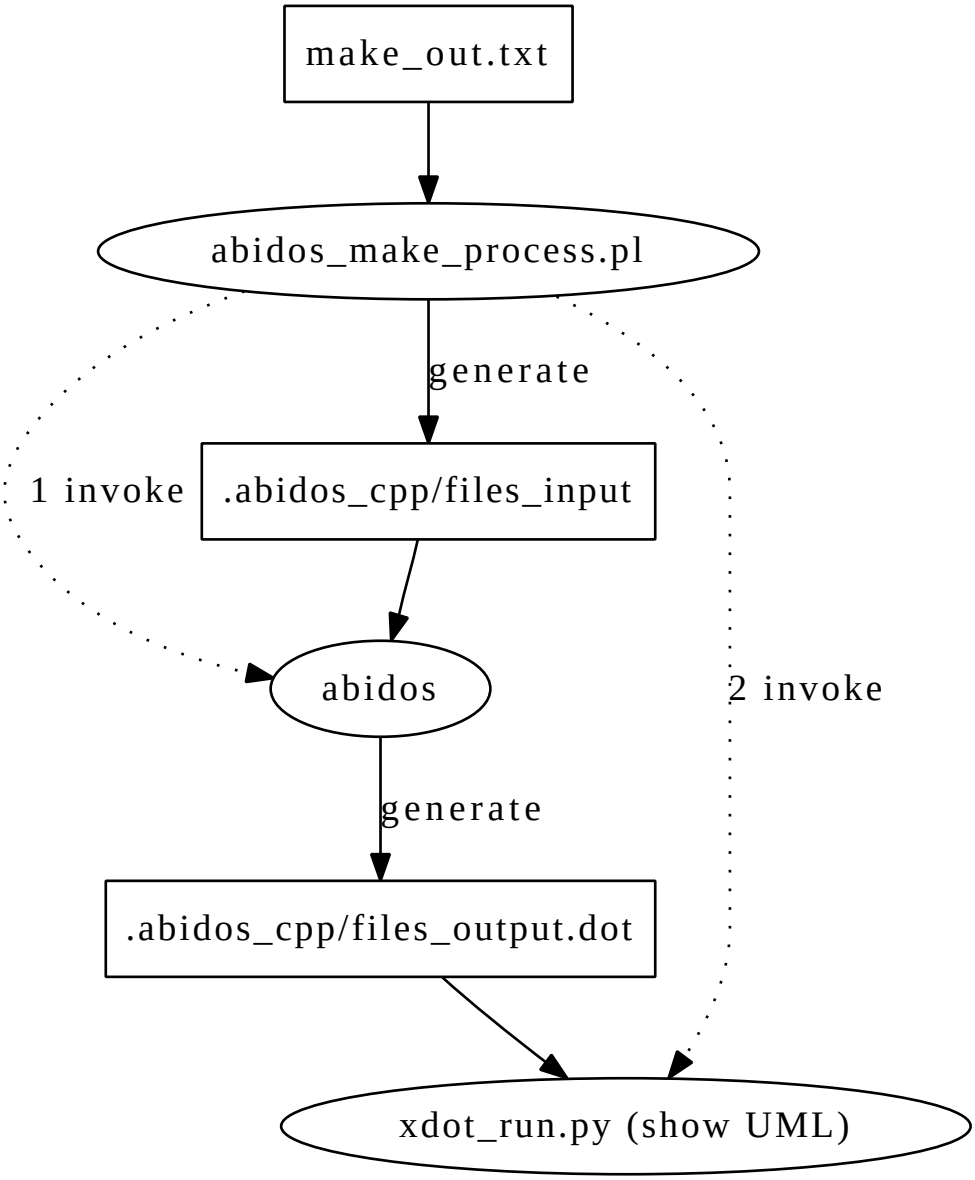
```
cd /home/.../src && /usr/bin/c++        \
-Wall -c -g -I/home/.../includes/       \
  -o CMakeFiles/abidos_cpp.dir/main.cpp.o   \
  -c /home/.../src/main.cpp
```

```
this line have been extracted from the make process of abidos itself, the
Makefile have obtained from "cmake ." process.
```

- It searches -c options to extract the cpp files.
- It searches -I options to extract the includes directories.
- It adds the directory of the cpp file too.

This figure shows the whole process:

[1] graphviz http://www.graphviz.org/ a superb project to generate and visualize graphs
[2] xdot.py http://code.google.com/p/jrfonseca/wiki/XDot with a little hacking of my own http://code.google.com/p/xdot-multi-line



to summarize the different parts of abidos have worked in the previous figure are:

- make_out.txt, is the output of make process.
- abidos_make_process.pl parses the make_out.txt and generates .abidos_cpp/files_input and executes abidos and xdot_run.py
- abidos load the cpp files from **.abidos_cpp/files_input** then generates
- .abidos_cpp/file_output.dot*, this file has dot format.
- xdot_run.py load **.abidos_cpp/file_output.dot** and show you to browse through the UML diagram.

When you have .abidos_cpp/file_output.dot generated you can launch the UML diagram whenever you want doing:

```
xdot_run.py/file_output.dot
```

Note
If you modified a C++ source file you would need to execute all the toolchain to see the new changes.

Now you have a deep knowledge about the abidos toolchain you can generate some parts of the toolchain by hand o by a script. You can write .abidos_cpp/files_input as you wish or you can comment some lines with # to drop some files from abidos parsing process.

You can invoke abidos to process .abidos_cpp/files_input

```
abidos_cpp --out_dir .abidos --loader .abidos_cpp/files_input (1)
```

(1) you can check if abidos worked looking if files_output.dot has been generated

```
ls .abidos_cpp/
files_input  files_output.dot
```

You can visualize the output of abidos

```
xdot_run.py .abidos_cpp/files_output.dot
```

The script **abidos_make_process.pl** for now is designed to parse the output from Makefiles that has been generated by cmake there a lot of others types of Makefiles that would not been parsed correctly by this script. If you need change **abidos_make_process.pl** to generate the **.abidos_cpp/files_input** is a good idea that you copy it in another place and change it, in the other hand you can write by hand the **.abidos_cpp/files_input** with the format explained before.

2.6 Whitelist

When you want Abidos generates a part of the whole UML diagram you can use this option.

For Example if you have this file:

main.cpp

```
class A
{
};

class B: A
{
};

class C
{
};

int main(int argc, char * argv[])
{
}
```

His whole UML generated by abidos_cpp is:

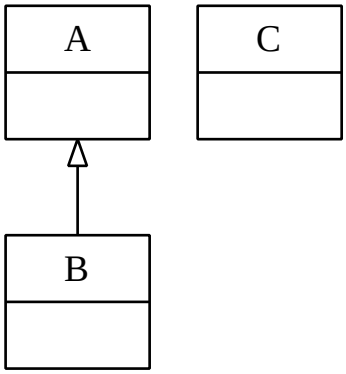


Figure 2.1: UML output.

And if you want for example to have a UML diagram with classes **A** and **B** but not **C** you can write the next file:

white_list_example.txt

```
#this is a comment
A
B
```

Now you can invoke **abidos_cpp** in this way:

```
abidos_cpp --out_dir . --white_list white_list_example.txt main.cpp
```

You can see the UML diagram with the next command:

```
xdot_run.py files_output.dot
```

It show you:

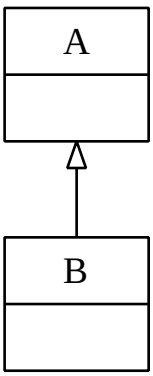


Figure 2.2: UML output.

Where only appears **A** and **B**.

Chapter 3

Uninstall

In order to uninstall abidos project execute:

```
cd abidos_cpp-master/processor/
sudo make uninstall
```

Chapter 4

Index