C++ EDITORS GNU/LINUX

Robin van Grinsven

NHTV/IADE/Europia

Disclaimer:

This paper is not on a PHD level of research. Thus, needs to be considered a personal research paper. A good start for your research. End of this document. You can do an informed decision on what editor you want to use.

Abstract

In this paper we make factual comparisons between different code environments setups. Since the possibilities are very wide we scope down the software with few software packages that are Popular. The packages will be featureful. And need to acknowledge that they are capable of coding in C++. This paper is developed to help companies decide what code editor to implement in their workflow. To prevent bias as much as possible the focus is on features, the implementation of the features and the added benefit of those features. In the conclusion we talk about the red line of the products the main benefits and main downsides.

Keywords: Coding, Editors, Comparison, IDE, Software

Table of Contents

C++ EDITORS GNU/LINUX

code editors come in many different shapes. You have IDE's and Text editors. A IDE is more a extendented text editor on the features of project management. We discuss both software packages. Since both can be used. We pick some big features and compare them between the picked packages.

IDE and Text editors

IDE(Integrated Development environments) and text editors are very similar. However, the difference is in the design/purpose of the programs. IDE are especially developed for programming. And make a complete package for the programming tasks. Everything is build and developed to work together. For the single task of programming in that langues. In general text editors you find that you need to construct multiple existing programs for a good working environment. Thus, more need to configure.

IDE's

We are going at some IDE's and some Text editors. The IDE's are: Code::Blocks, CodeLite, Eclipse and Mono develop. These are feature-full and open source widely used programs.
Code blocks and Code Lite main focus are C++. Code blocks also has native support for Fortran.
Eclipse is build for Java. However with plugins your can make it work with C++. This editor is developed for websites
MonoDevelop is developed for working with C#. but again can work with C++. This is used with the unity game engine.

Text editors

The text editors we go and look at are Emacs, Sublime, Vim and Visual studio code.
We look at emacs because it has the longest lifespan of all the editors. Is highly customizable. And has many big database of plugins. In essence you can make out of it what you want.

Vim is very close to Emacs however very light weight. And also has a big community. Vim is more popular then Emacs.

Sublime has a heavy focus on making typing easy. Goes for slick design. It is loved by many developers.

Lastly visual studio code. This is a striped down visual studio. Just the text editor. So developers on a Linux machine can have

Unix helping programs.

For Missing features in text editors GNU developed some programs to fill in the gaps. You wil notice in the future of this document. Some of these tools are build in IDE's like Code::Blocks

listing these programs are:

GCC/G++: these commands compile the code to machine languages. Gcc to compile c code and G++ for c++ code.

Make/AutoMake/AutoConf/LibTool : these are Linux tools compile multiple files of one projects. This is comparable to the sln file in Visual studio and cbp from Code::Blocks. So Project files.

Chmod + : give the promission in linux to execute the compiled file.

./the_compiled_file_name : executes the file.

GDB/undoDB : debuggers. To analyse code and know what happens in the code.

Objdump/Boomerang/hex-ray/Hopper: decompiler. Makes binear code back to assembly or c.

Git/Svn/p4: version controllers.

Bison : a Praser generator, warns about parsers ambiguities.

Macro proccesor: m4

Feature comparison

For comparing the tools you can look statistically at the features. Other factors to consider a program by are documentation, work flow and usability.

File management

When looking at file management we dissect the part of the programs that handles the project and files. Considering the time spend on managing the project versus working on the code it self. A good project management leads to more productivity.

programs	Auto-save(AS)	Overview	Header Files	ProjectM File	Source control
Code::Blocks	Plugin	Yes	Basic	.cbp	Plugin: svn,git
CodeLite	Integrated: plugin	Yes	Full	no	Plugin:svn,git
Eclipse	Plugin	Yes	No	.eclipseproduct	Plugin: git
Emacs	Change config/buffer	No	No	No	VC:git,svn, plugin: p4v
Mono-develop	Yes	Yes	Basic	mdtool	svn,git, plugin p4v
sublime	Yes	Yes	no	no	git,svn
vim	plugin	no	no	no	Plugin: git,svn,p4v
Visual studio code	Yes	Yes	Basic	Yes	Plugin: git,P4v Extern: svn

A basic feature of securing progress is the *auto save.* As humans tent to forget to save the file regularly as well as not want to spend time on this. Because we expect stability of the system. Nor should the user be concerned about losing progress. As a simple auto save would solve the issue.

A note should be taken that most of these programs support this in a plugin state. And is not provided in an as-is. However concerning the Emacs product, it is handled different. In default when opening a file Emacs copy's the file. This is the buffer autosave file. Whenever you work in the Emacs buffer(copy of the file in RAM). Every 10 seconds it saves current state to the buffer autosave file on closing the file Emacs deletes the buffer file. If the file is open on Emacs

and the pc crashes and after reboot you open the normal file. Emacs recognizes you still have the buffer file. And asks if you want to open that. In other programs this procces is also appearend when you do a file recovery.

Overview of your project files helps navigating trough the file and shows all the files coupled to your project. Most of the IDE's also have a class browser. Act as the same way. Where it tree and shows the classes in all.

The generating of *header files* is really an IDE feature. Since this is a thing only happening when coding in C or C++. It generates the basis of a header file when creating a new file for a new class for instance. Eclipse does not have this feature since it is focus on Java. An other point of interest is the CodeLite feature. CodeLite does not only generate the header file on creating of a new file. It also keeps it up to date to the current state of the file.

Another feature that is more appear end in IDE's is the feature of the *Project management file*. This it the file that saves the connection of all files together including the compile structure. This is normally the file that used to start up the IDE. Thus sometimes also contains some editor settings. For visual studio users this is called the sln(solution) file.

For *source control* most of the time the editor asks for plug-ins. That is because source control is used over a whole project. Not only the coding side. Thus there are third party programs that are in need of use. Every company has their own approach so simple plugins are used by the editors.

Code evaluation

Once the code is written the program needs to be evaluated. On speed, resources, bugs and efficiency. There are multiple tools for these assignments. All give u other insight on your code. That this is an important part of development does not need to be addressed. We address the most used code evaluation features. We see many third party programs here. Or integration of other proven programs. What you see is that IDE's include the programs during install while text editors mostly ask extra setups we can notice mono develop has not many of the features.

programs	Compiler	Debugger	Profler	Disassambler	Break point
Code::Blocks	MinGW + custom	GNU GDB	Yes	yes	yes
CodeLite	GCC, Clang, VC + custom	Yes(multi-threading)	Valgrind	LLVM	Yes
Eclipse	External	yes	Yes	yes	yes
Emacs	GCC,make, terminal interface	GDB GUD	Yes(text)	External	Yes GUD
Mono-develop	GCC + custom	yes	no	no	no
sublime	External	plugin	no	External	Yes Xdebug
vim	External	plugin	no	External	Hard to use
Visual studio code	own	own	?	?	?

Compilers GCC is still the most popular. However most of the tools leave you to decide what you use for a compiler. For Sublime and Vim an external compiler is needed. This means most likely you need to open a terminal externally and compile it there. For Emacs there is an exception. It is able to do terminal commands in the editor. Eclipse require plugin that handel it for the user.

CodeLite outshines at *debugging* since it is able to debug multi-threading programs. But this topic has no further outshines

Profilers are also more dedicaded to IDE's, notiable is Emacs having it intergrated. And Mono develop does not. For Visual studio code it get's harder to find the functions because it has a similar name to it parent Visual studio.

Disassembler are programs that try to make binary files back to a other langues. Most used is assembly. This makes u able to find what the compiler made out of your code. And read them back in to instructions. This tool is used for backwards engineering. But also for finding what your final instructions are on the machine. So you can optimize your code. This is also a feature some have and some don't.

For *breaking point* text editors use mostly external programs. And for the IDE's it is intergrated. Looking at Vim it is seen as a hard implementation. Since you need to intergrate the breaking point by adding text line In the code file. This leads to clutter and extra clean up steps.

Data interaction

This is mainly the features how you help you develop your code you want. This include text editing as well as creating a good overview what you code currently is. You see that the text editors are just as good as the IDE's in these tasks some even better.

programs	Auto indent	Find and replace	Multi cursor	Class browser	Multi window	Spellcheck	Auto completon	Code folding
Code::Blocks	yes	yes	no	Yes	yes	No	yes	yes
CodeLite	plugin	yes	Build in	yes	yes	plugin	Poor	no
Eclipse	Yes	Yes	yes	yes	yes	Yes	Yes	Yes
Emacs	Yes(correc tive)	Yes	Plugin	Yes	Yes(tle design)	Yes	Yes(iron)	yes
Mono-develop	Yes	Yes	No	Yes	yes	No	yes	?
sublime	Yes	yes	yes	Yes(over project)	Yes: no overlaps	Yes	yes	yes
vim	Yes	yes	plugin	plugin	Yes	Yes	yes	yes
Visual studio code	Yes	yes	Yes	yes	Yes :no overlaps	plugin	Yes(intelliJ)	yes

For readability in text we *indent* code to show that is in above function, like a tree. Code editors reconise when you type a code and know when to expect to put a tab in the code. Emacs has a special way to handel it. when you press tab on a already typed sentence. It evaluates the code and fixs it to the code structure. So wont change when it is already right. This helps you spot syntax errors. However by default it tabs with 2 spaces instead of 4. you can change that in the config if you want it different.

Find and replace is the function that helps programmers rename their variable names. This is the most standard feature in editors.

Multi Cursor is a feature that is not used alot. It allow you to type in mutible places in your code. Good for renaming varibiable or do the same thing in variouse places.

Class browser gives you the ability to look at the code of a class you are calling in a other place of code. Sublime is able to arch it over to other classes over your whole project.

Sometimes we want the code to have no english spelling error's. Here we use spellcheckers

Multible windows are handy for looking at more then 1 code at once. When you need to know what is going on in the other code or if you quickly want to switch between code's. Some of the editors pick that you can not overlap windows over eachother. Because that can lead to clutter on the screen. For instance emacs use a tile based system when you split the view in 2.

checking spelling errors is handy if you want basic english and readibilty. Or you are bad at spelling. This can get in the way sometimes. Since we use repesentivies in code like”Pvalue” for declaring a pointer to a other value. However as come to no suprise the text editors provide a better support for it then IDE's.

Auto complete is another widely supported feature. However some are more advanced then others. Some have a narrow field to look for guessing others look at the whole project.

Code folding this is a feature that can be good or bad. What it means that the code that has a longer line then fit on the screen it start writing on the next line. This can be confusing if you not notice that the line continues on the next line. And can give a hard read.

Highlights

We describe the most notable points of every program. This does not mean there are non more.

Pro's

Code::Blocks: easy fixable editor.

CodeLite: many options for settings.

Eclipse: support for many languages.

Emacs: highly customizable.

Mono-develop: modern look.

Sublime: slick design. Developed for usability and speed.

Vim: most light weight editor.

Visual studio code: very similar to visual studio.

Con's

Code::Blocks: unstable display of data structure.

CodeLite: no big community compare to other on this list.

Eclipse: no native support for C++

Emacs: take time to set up a good environment.

Mono-develop: low on extra features.

Sublime: is a text editor and only one that cost money. And is closed source.

Vim: shortcut heavy.

Visual studio code: because name is very similar to visual studio. It is hard to find this product specific info.

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.

LiveEdu staff (May 23,2016). “10+ Best text editors for Programming 2016/2017”. http://blog.liveedu.tv/10-best-text-editors-programming-2016/

https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

https://en.wikipedia.org/wiki/List_of_text_editors

https://en.wikipedia.org/wiki/GNU_toolchain

http://codeblocks.org/

https://www.codelite.org/

http://www.eclipse.org/

https://www.gnu.org/software/emacs/

http://www.monodevelop.com/

https://www.sublimetext.com/

http://www.vim.org/

https://code.visualstudio.com/

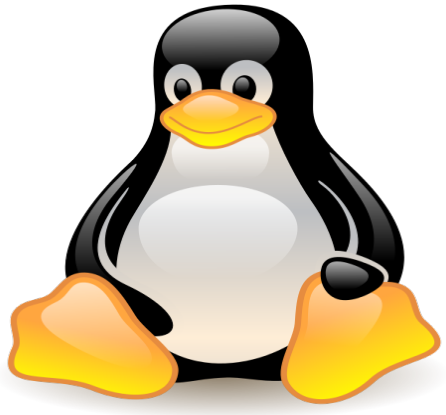
icons :

GNU(gnu not unix) development icon

Aurelio A. Heckert <aurium@gmail.com> - gnu.org

File:Heckert GNU white.svg
Created: 12 December 2005

linux tux pinguin:



Permission details

The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted. Attribution: lewing@isc.tamu.edu Larry Ewing and The GIMP

Jeremy Kratz - https://github.com/isocpp/logos

File:ISO C++ Logo.svg
Created: 30 January 2017