COP 3014: Spring 2018 A Guide to Using CLion

January 16, 2018

1 CLion

CLion is a comprehensive C++ Integrated Development Environment developed and maintained by JetBrains. It is very intuitive and easy to use. For this class, we will use CLion for all of our work.

1.1 Applying for a Student License

In order to use CLion, you need to apply for a license. Go to https://www.jetbrains.com/student/. Click on the "Apply" button. You should see the following screen:

Apply with:	UNIVERSITY EMAIL ADDRESS	ISIC/ITIC MEMBERSHIP	OFFICIAL DOCUMENT
Status:	 I'm a student 		
	🔵 l'm a teacher		
Name:	First name	Last name	
	Our software will be registered to your	real name.	
Email address:			
	Your valid university email address, e.g. I confirm that the email address provid		
Country:	United States		•
	I have read and l accept the	JetBrains Privacy Policy	
	I have read and I accept the APPLY FOR FREE PRODUCTS	JetBrains Privacy Policy	

JetBrains Products for Learning

© 2013-2018 JetBrains | Help | Support | Privacy Policy | Terms Of Use | Build #1134

Figure 1: Applying for a Student License

Please provide an FSU email. You will get a reply from JetBrains in a few minutes, asking you to confirm the request for a student license.

Once you confirm the request, they will send you another email, asking you to create a JetBrains Student Account. This takes about 5 minutes. Once you create the account, you should see a page with your Key.

2 Installing CLion

You can now download and install CLion. Download CLion from https://www.jetbrains.com/ clion/download. The page will automatically show you the right installer for your Operating System. Download the installer.

2.1 For Macs/Linux Machines

If you have a Mac or a Linux Machine, you already have a lot of the prerequisite programs needed for CLion. All you need to do is a couple of simple steps.

2.1.1 Mac OSX

Once you have downloaded the dmg file, use Finder to open the Applications folder. Move the file you just downloaded from the Downloads folder into the Applications folder. You're all set.

2.1.2 Linux Machines

Download the tar from the website. Unpack it and then navigate to the **bin** directory and run CLion.sh

2.2 For Windows Machines

Download the installer from the website and run it. It will start up the installation wizard.

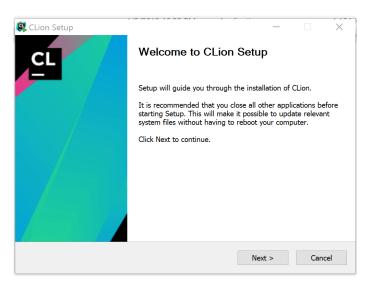


Figure 2: Installation Wizard

Click on Next. It will ask you to choose a location to install CLion. Go with the default option.

CI		tall Location				
	Choose the	folder in which	to install CLi	ion.		
Setup will install CLic			a different	folder, click	Browse a	ind
select another folder.	. Click Next to contin	ue.				
Destination Folder						
	\JetBrains\CLion 201	7.3.1		Bro	wse	
		7.3.1		Bro	wse	
C:\Program Files	7MB	7.3.1		Bro	wse	
C:\Program Files	7MB	7.3.1		Bro	wse	

Figure 3: Installation Location

Click on Next. It will ask you for a temporary folder to put the temporary files. Go with the Default Option. Click on Next. It will ask you to choose file associations. Check all the boxes.

CLion Setup	Installation Configure y	Options your CLion ir	nstallation	_		×
Create Desktop Shortcut						
Create Associations	.C	.cpp	.cxx	.hpp	.hxx	
			< Back	Next >	Cance	4

Figure 4: Choosing File Associations

Click on Next. This will complete the setup. It will ask if you want to run CLion. Say Yes.

3 Running CLion for the first time

Once you have installed CLion, you can run it. There is a bit of setup to do when we run CLion for the very first time. This is mostly the same on all platforms. For Windows, there is an extra setup to get a Unix based C++ compiler installed, but the CLion Wizard will guide you through it.

When we first run CLion, it will ask if we would like to import settings. You will see the following dialog box.

🛱 Complete Installation	\times
Import CLion settings from:	
Custom location. Config folder or installation home of the previous vers	sion:
C:\Program Files\JetBrains\CLion 2017.3.1	
● Do not import settings	
OK	
pload Clion from	

Figure 5: Importing Setting

Choose "Do Not Import Settings". Click OK. It will noiw ask you to activate your CLion License. Put in the Username and Password for the JetBrains account you created. Click on Activate

,	
CLion License Activation -	- 🗆 X
● Activate ○ Evaluate for free	Buy CLion
Activate license with:	
JetBrains Account OActivation code	○License server
Username or email:	
Password:	Forgot?
	Activate Exit

Figure 6: Activate your license

It will ask you to choose a theme. Pick one that you like.

Intell/ Project) @ main.cpp) Modelistati × @ main.cpp × stancups < lostreaco using namespace std; Ont stail() { cost << "Bello, World!" << end; return 0; fl restpoints + - (D)	Darcula project @ main.cpp v
	Section 2 (Construction of the section of the

Figure 7: Choose a Theme

Click on Next: Toolchains. This is where the paths diverge for Mac and Windows machines.

3.1 For Macs / Linux Machines

Macs and Linux Machines most likely already have working installations of the C++ compiler, the make utility and a version of the GDB debugger. CLion should be able to automatically detect these. You should see a screen that looks like this:

	Build, Execution, Deployment > Toolchains
Appearance & Behavior Menus and Toolbars	CMake executable:
System Settings	G Bundled CMake 3.8.2
Scopes	Custom:
Notifications Quick Lists	Debugger.
Path Variables Keymap	Bundled GDB 7.11.1
▶ Editor	O Custom GDB:
Plugins	O Bundled LLDB 3.9.0
 Version Control Build, Execution, Deployment 	 Bundled LLDB 3.9.0
Tookhains	V CMaker 3.8.2
CMake	Solution of the state of th
Debugger	V C Compiler: /usr/bin/cc
Python Debugger	C++ Compiler: /usr/bin/c++
Python Interpreter	√ CDB: 7.11.1
Console	
Languages & Frameworks	
> Tools	
> Tools	OK Cancel Apply Help

Figure 8: Toolchains

Click on OK. Your setup is now complete.

3.2 For Windows Machines

Windows machines do not come with an installation of the required programs, so you should see a screen that looks like this:

Customize CLion		
JI Themes Toolchains Default pl	ugins Featured plugins	
Configure Toolchains		
+ - 🗊 🕈 🕈		
	No tookheins configured	
No tradebairs	You can add MinGW or Crowin toolcheir	
No toolcheins		
Toolchains can be changed later in Set	ttings Build, Execution, Deployment Toolchains	
Skip Remaining and Set Defaults	Back to UI Themes	Next: Default plugins

Figure 9: Toolchains

Click on the link that says Cygwin. You should now see this:

	→ Default plugins →	reatured progres		
onfigure Toolchain	s			
+ - 🗅 + +	Name:	Cygwin		
/gwin	Environment:		Downlo	oad
	Cygwin 🗸		~	
		(3 Not specified		
	CMake:	Bundled	~	
		Does not work in this environment		
	Make:	Detecting	~	-
	C Compiler:	Detecting	~	
	C++ Compiler:	Detecting	~	
		a		
	Debugger:	Use Cyqwin GDB		
		Not found, please install this package		

Figure 10: Toolchains

Click on Download. It will open the following screen on your web browser.



Figure 11: Installing Cygwin

Download the installer and run it. This might require Admin privileges.

- It will ask you for installation options. Choose "Install from the Internet".
- It will ask you to choose a directory for installation. Go with the default option. It will also ask you to choose users. Choose "All Uses".
- It will ask you where to store the Downloaded files. Go with the default option.
- On the next screen, choose "Proxy Settings".
- The next screen will show you a list of "mirrors". You can pick any of them. I used the one from RIT, towards the bottom of the list. Click on Next. You should see the following screen



Figure 12: Installing Cygwin

Click Next. It will ask you to choose which packages to install. To keep things simple, we're going to install everything.

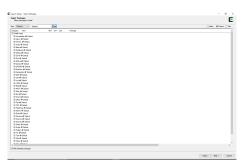


Figure 13: Installing Cygwin

Make sure it looks exactly like it does in the image. Now, click Next. Cygwin will be installed on your computer. This will take a while. Grab some coffee and cookies in the meantime. Once the installation is done, you will see the following screen.

cygwin (default) MinGW 32 mingw-w64 32-bit mingw-w64 64-bit ↓	Name:	cygwin		
	Environment:		Download	
	+	Cygwin 🗸	E:\Toolchains\cygwin2.8	~
			Version: 2.8.2	
		CMake:	Bundled	~
			Version: 3.8.2	
		Make:	Auto: E:\Toolchains\cygwin2.8\bin\make.exe	
		C Compiler:	Auto: E:\Toolchains\cygwin2.8\bin\gcc.exe	
		C++ Compiler:	Auto: E:\Toolchains\cygwin2.8\bin\c++.exe	
		Debugger:	Use Cygwin GDB	
		Version: GNU gdb (GDB) (Cygwin 7.11.1-2) 7.11.1		

Figure 14: Toolchains

CLion should automatically be able to detect the required programs now. You installation is complete.

4 Creating a Project

In order to write and run a C++ program on CLion, you should first create a project. Open CLion and then choose File ->New Project.

New Project		E Project
Import Project New Social Pile Christikeshiftensert Open URL Open URL Open Social • Gase Project		
P Settings CtrivAlt+S Default Settings		
import Settings Export Settings Settings Repository	Search Everywhere Double Shift Go to File Carl+Shift+N	
🖉 Relaad CMake Project		
a gave All Corles	Recent Files Ctrl+E	
Synchronize ColleAlter Invalidate Caches / Restart	Navigation Bar Alt+Home	
T Bist.	Drop files here to open	
Power Save Hode		
Ext.		

Figure 15: Opening a New Project

You will now see a screen where it asks you to choose where to save the project. Also, the project is called "untitled" by default. Change it to a name of your preference.

😑 💿 New Project	
C++ A C++ Executable	Location: /home/sharanya/CLionProjects/Project2
C++ Library	Language standard: C++11 💌
C Executable	
	\$
	Create

Figure 16: New Project Name and Location

Click on Create. This will open a new project with a default program stub. This program is called "main.cpp". We will usually require to to give specific names to your programs. This requires you to refactor the project. Right Click on "main.cpp" in the Project Explorer Window on the left and choose Refactor ->Rename.

	💑 main.cpp 🔪		4); 🔄 Project2 🕨 🕷 🔲 🔾
Project	⊕ 💠 🕸 🕹 CMakeLis	ts.txt × 🛃 main.cpp ×	
	-/CLionProjects/Project2 1 #inc	lude <iostream></iostream>	
A CMake	diete twt 3 int	main() {	
🗒 main.c	4	<pre>std::cout << "Hello, World!" << std::endl; return θ;</pre>	
III External L	New +	return o;	
	Ø Reload CMake Project		
	¥ Cut Ctrl+X		
	Ctrl+C		
	Copy Path Ctrl+Shift+C		
	Copy Relative Path Ctrl+Alt+Shift+C		
	1 Paste Ctrl+V		
	Find Usages Alt+F7		
	Inspect Code		
	Befactor +	Benamite Shift+F6	
	Clean Python Compiled Files	Change Signature Ctrl+F6	
	Add to Favorites +	Move F6	
	Delete Delete	Copy F5	
	Local History	Safe <u>D</u> elete Alt+Delete	
	5 Synchronize 'main.cpp'	Extract +	
	Show in Files	Inline Ctrl+Alt+N	
	File Path Ctrl+Alt+Shift+2	Invert Boolean	
	Compare With Ctrl+D	Puļl Members Up	
	@ Create Gist	Push Members Down	
	W create dist		

Figure 17: Renaming the file

In the following window, give the file its required name.

	🔋 🗊 Rename
1	Rename file 'main.cpp' and its usages to:
	sample.cpp
	Search in <u>c</u> omments and strings
	Refactor Preview Cancel Help

Figure 18: Renaming a file

Click on Refactor. You are now ready to start writing your program.

5 Running your program

Once you have completed writing your program, save your work by choosing File ->Save. Before running your program, you have to Build your project. This compiles your program and if your program is error free, it gets your program ready for execution. Choose Run ->Build. You should see something like the following screen:

Project2) 🖾 sam	ple.cpp)		🗄 🔂 Projectz - 🕨 🕷 🗏 🛛
Project	00.0.1	A CMakeLists bit < Lample.cpp >	
In Project - (CLO > In crusteesuff) A CMaketosta - sample.cpp In External Ubrack	debug et	Inclusion distribution Let main[] = the second sec	
essages build		817.2.3/bin/cmake/bin/cmakebuild /home/sharanya/CLionProjects/Project2/cmake	٥
Scanning dep [50%] Build [100%] Linki		Project2 Files/Project2.dir/sample.cpg.o	

Figure 19: Building the Project

The console window on the bottom of your screen will show you the Build results. If your program has errors, they will be listed there. If the program compiles fine, it will say "100% Built Target" followed by the Project name.

Once your project builds successfully, you can run it. Choose the project on the drop down list

in the top right corner and then click on the green arrow button next to it. Your output will be displayed on the bottom of the screen in the console window.



Figure 20: Running the Project

6 Opening an Existing Project

To open an existing project choose File >Open. In the dialog box that pops up, navigate to the location where you saved the project. Click on the project name and then click OK.

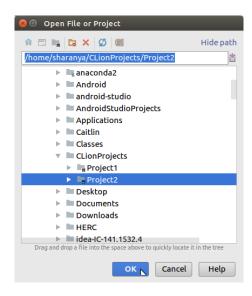


Figure 21: Running the Project

If you already have a project open, CLion will ask you if you would like to open this project in the same window or in a new window. You can choose either option.

7 Turning in your assignment

CLion would have created a folder where it places your projects by default. Alternatively, you might have specified a folder to put your project in when you created it. Navigate to the folder on Finder (Mac) or Folders (PC) and you will see a Folder for your Project. Opening that will show you all the source files under that project. The .cpp file is the file you need to turn in through Canvas

8 Troubleshooting

This section can be used for troubleshooting the most common errors you might encounter while installing CLion. If your problem persists after trying this, please contact your instructor/ TA for help.

8.1 On Windows

The most common issue on Windows is a glitch in the CygWin installation when a few critical dependency files are not downloaded. Please go through the following steps:

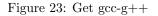
- 1. Open CLion. Go to File ->Settings. Choose Build, Execution, Deployment. and then choose Toolchains.
- 2. You will see "Download" link on the right. Click on the link.
- 3. Go through the steps for the CygWin installation again (the list below Figure 11), until you see the packages screen.
- 4. Go to the Drop-down box on the top left, that says "Pending" and choose "Full".
- 5. Search for gcc-core. You will see the following screen. Click on the circular arrow next to "Skip" on the line for gcc-core: GNU Compiler Collection (C, OpenMP). Keep clicking until you see a number like 6.4.

w Full	✓ Search gcc-core Clear					
urrent	New	Bin?	Src?	Categories	Size	Package
4.0-1	Keep	nja		Devel	16,464k	cygwin32-gcc-core: GCC for Cygwin 32bit toolchain (C, OpenMP)
4.0-1	Keep	nja		Devel	7,926k	djgpp-gcc-core: GCC for DJGPP toolchain (C)
.4.0-4	6.4.0-2	\times		Devel	17,829k	gcc-core: GNU Compiler Collection (C, OpenMP)
.4.0-1	Keep	nja		Devel	14,476k	mingw64i686-gcc-core: GCC for Win32 (i686-w64-mingw32) toolchain (C, OpenMP)
6.4.0-2	Keep	nja		Devel	14,921k	mingw64-x86_64-gcc-core: GCC for Win64 toolchain (C, OpenMP)

Figure 22: Get gcc-core

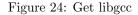
6. Clear the search bar. Search for gcc-g++. You will see the following screen. Click on the circular arrow next to "Skip" on the line for gcc-g++: GNU Compiler Collection (C++). Keep clicking until you see a number like 6.4.

Select Packages Select packages t	o install					
View Full	✓ Search gcc-g++ CI	ear				
Current	New	Bin?	Src2	Categories	Size	Package
Current	TNOW		0.01	Caregonou	0120	- denage
6.4.0-1	Keep			Devel		cygwin32-gcc-g++: GCC for Cygwin 32bit toolchain (C++)
				-	10,456k	2
6.4.0-1 5.4.0-1	Кеер	nja nja		Devel	10,456k 8,279k	cygwin32-gcc-g++: GCC for Cygwin 32bit toolchain (C++)
6.4.0-1	😯 Кеер 😯 Кеер	nja nja		Devel Devel	10,456k 8,279k 10,666k	cygwin32-gcc-g++: GCC for Cygwin 32bit toolchain (C++) djgpp-gcc-g++: GCC for DJGPP toolchain (C++)



7. Clear the search bar. Search for libgcc. You will see the following screen. Click on the circular arrow next to "Skip" on the line for libgcc1: GCC C runtime library. Keep clicking until you see a number like 6.4.

Select Packages Select packages to i	install								
View Full ~	' Search	libgcc Cl	ear						
Current		New	Bi	n? S	Grc?	Categories	Size		Package
6.4.0-4		0 6.4.0 2	X			Libs		27k	libgcc1: GCC C runtime library
		Skip	nja	n/	la	Libs		2k	libgccpp1: Boehm-Demers-Weiser garbage collector library



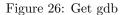
- 8. Clear the search bar. Search for make. You will see the following screen. Click on the circular arrow next to "Skip" on the line for cmake: Cross Platform makefile generation system. Keep clicking until you see a number like 3.3.
- 9. Scroll down a little bit until you see make: The GNU version of the make utility. Click on the circular arrow next to "Skip" on this line until you see a number like 4.1.

iew Full	 Search make 	Clear				
Current	New	Bin?	Src?	Categories	Size	Package
1.14.1-3	Keep	nja		Devel	773k	automake1.14: (1.14) a tool for generating GNU-compliant Makefiles
1.15.1-1	Keep	nja		Devel	792k	automake 1.15: (1.15) a tool for generating GNU-compliant Makefiles
1.4p6-11	Keep	nja		Devel	248k	automake 1.4: (1.4) a tool for generating GNU-compliant Makefiles
1.5-11	Keep	nja		Devel	332k	automake 1.5: (1.5) a tool for generating GNU-compliant Makefiles
1.6.3-12	Keep	nja		Devel	365k	automake 1.6: (1.6) a tool for generating GNU-compliant Makefiles
1.7.9-11	Keep	nja		Devel	426k	automake 1.7: (1.7) a tool for generating GNU-compliant Makefiles
1.8.5-11	Keep	nja		Devel	499k	automake 1.8: (1.8) a tool for generating GNU-compliant Makefiles
1.9.6-11	Keep	nja		Devel	557k	automake 1.9: (1.9) a tool for generating GNU-compliant Makefiles
3.6.2-1	0 3.3.2 1	\boxtimes		Devel	3,577k	cmake: Cross-platform makefile generation system
3.6.2-1	Keep	nja		Debug	123,232k	cmake-debuginfo: Debug info for cmake
3.6.2-1	Keep	nja		Devel	869k	cmake-doc: Cross-platform makefile generation system (documentation)
3.6.2-1	Keep	nja		Devel	1,233k	cmake-gui: Cross-platform makefile generation system (GUI)
3.6.2-1	Keep	nja		Editors	5k	emacs-cmake: Cross-platform makefile generation system (Emacs mode)
5.39.0-2	Keep	nja		Devel	209k	extra-cmake-modules: Extra CMake Modules for KDE
1.9.6-2	Keep	nja		Devel	419k	gcc-tools-epoch1-automake: (gcc-special) a tool for generating GNU-compliant Makefile
1.11.6-1	Keep	nja		Devel	589k	gcc-tools-epoch2-automake: (gcc-special) a tool for generating GNU-compliant Makefile
1.0.3-1	Keep	nja		Devel	6k	gccmakedep: X Makefile dependency tool for GCC
1.0.7-1	Keep	nja		Devel	34k	imake: X Imake legacy build system
1.0.7-1	Keep	nja		Debug	41k	imake-debuginfo: Debug info for imake
0.95.8-1	Keep	nja		Libs	3k	libWMaker-devel: Window Maker interface library (development)
0.95.8-1	Keep	nja		Libs	4k	libWMaker1: Window Maker interface library (runtime)
0.0.3-1	0.0.3-2	\boxtimes		Graphics	5k	libpagemakertools: MS Publisher file converters
0.0.3-1	0.0.3-2	\times		Debug	591k	libpagemaker0.0-debuginfo: Debug info for libpagemaker0.0
0.0.3-1	0.0.3-2	\boxtimes		Libs	2k	libpagemaker0.0-devel: Adobe PageMaker import filter library (development)
0.0.3-1	0.0.3-2	\boxtimes		Libs	109k	libpagemaker0.0-doc: Adobe PageMaker import filter library (API documentation)
0.0.3-1	0.0.3-2	\boxtimes		Libs	65k	libpagemaker0.0_0: Adobe PageMaker import filter library (runtime)
4.2.1-2	0 4.1-1	\boxtimes		Devel	414k	make: The GNU version of the 'make' utility
4.2.1-2	Keep	nja		Debug	368k	make-debuginfo: Debug info for make
1.0.5-1	Keep	nja		Devel	30k	makedepend: X Makefile dependency tool
105.1	A Keen	nío		Debug	634	makedenend-debuginfo: Debug info for makedenend

Figure 25: Get cmake and make

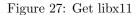
10. Clear the search bar. Search for gdb. You will see the following screen. Click on the circular arrow next to "Skip" on the line for gdb: The GNU debugger). Keep clicking until you see a number like 7.10.

Select Packages Select packages to install						
∕iew Full ∨ Sea	rch gdb Clear					
Current	New	Bin?	Src?	Categories	Size	Package
0.6.7+20170812+gitfa44c9a-1	Keep	nja		Devel	411k	cgdb: A curses-based interface to the GNU Debugger (GDB)
0.6.7-1	😯 Keep	nja		Debug	334k	cgdb-debuginfo: Debug info for cgdb
7.10.1-1	7.11.1-2	\times		Devel	2,609k	gdb: The GNU Debugger
7.10.1-1	🕃 Keep	nja		Debug	8,434k	gdb-debuginfo: Debug info for gdb
1.12-1	🚯 Keep	nja		Database	132k	gdbm: GNU dbm implementation
1.12-1	🕃 Keep	nja		Debug	203k	gdbm-debuginfo: Debug info for gdbm
1.12-1	🚯 Keep	nja		Libs	12k	libgdbm-devel: GNU dbm implementation
1.12-1	🕃 Keep	nja		Libs	19k	libgdbm4: GNU dbm implementation
1.12-1	🚯 Keep	nja		Devel	56k	mingw64-i686-gdbm: GNU dbm implementation for Win32 toolchain
1.12-1	🚯 Keep	nía		Devel	57k	mingw64-x86 64-gdbm: GNU dbm implementation for Win64 toolch



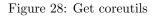
11. Clear the search bar. Search for libx11. You will see the following screen. Click on the circular arrow next to "Skip" on the line for libx11-devel: XOrg X11 core library (development). Keep clicking until you see a number like 1.6.

elect Packages Select packages	to install					
ew Full	✓ Search libx11 Cle	ar				
Current	New	Bin?	Src?	Categories	Size	Package
1.6.5-1	Keep	n/a		Debug	2,202k	libX11-debuginfo: Debug info for libX11
1.6.5-1	1.6.3-1	\times		Libs	68k	libX11-devel: X.Org X11 core library (development)
1.6.5-1	Keep	n/a		Doc	1,894k	libX11-doc: X.Org X11 core library (documentation)
1.6.5-1	Keep	n/a		Libs	2k	libX11-xcb-devel: X.Org X11 XCB interface (development)
1.6.5-1	Keep	n/a		Libs	2k	libX11-xcb1: X.Org X11 XCB interface (runtime)
1.6.5-1	Keep	n/a		Libs	734k	libX11 6: X.Org X11 core library (runtime)



12. Clear the search bar. Search for coreutils. You will see the following screen. Click on the circular arrow next to "Skip" on the line for coreutils: GNU core utilities (includes fileutils, sh-utils and textutils). Keep clicking until you see a number like 8.25.

slect Packages Select packages to instal									
ew Full	 Search coreutils C 	lear							
				-					
Current	New	Bin?	Src?	Categories	Size	Package			
Current 8.26-2	New	Bin?	Src?	Categories Base		Package coreutils: GNU core utilities (includes fileutils, sh-utils and textutils)			



13. Clear the search bar. Then click Next. You will see the following screen where CygWin will show you a list of dependencies. Just click Next. CygWin will now install the new packages we have specified. This should not take too long, only about 10 minutes. On the final screen of the CygWin installation wizard, click Finish.

E Cygwin Setup - Resolving Dependencies	-		×
Resolving Dependencies The following packages are required to satisfy dependencies.		l	E
libboost_atomic1.64 (1.64.0-1) Boost C++ libraries Required by: libboost-devel libboost_chrono1.64 (1.64.0-1) Boost C++ libraries Required by: libboost-devel, libboost_timer1.64		^	
libboost_container1.64 (1.64.0-1) Boost C++ libraries Required by: libboost-devel < ✓ Select required packages (RECOMMENDED)		>	
< Back Ne	ext >	Car	ncel

Figure 29: Dependencies

14. Back in CLion, the software will now try and detect the new libraries. After about a minutes, it should have automatically detected all the libraries. Click OK.

😫 Settings					×
Q-	Build, Execution, Deploym	ent → Toolchains			
V Appearance & Behavior Appearance	Cygwin +	Name:	Cygwin		
Menus and Toolbars	D	Environment			nload
> System Settings		Cygwin 🗸	C:\cygwin64		·
File Colors			2.9.0 Version: 2.9.0		
Scopes 0	1 +	CMake:	Bundled		·
Notifications			C Version: 3.9.6		
Quick Lists		Make:	Detecting		·
Path Variables		C Compiler:	Detecting		/
Keymap		C++ Compiler:	Detecting		/
/ Editor		Cre compiler.	/// · · · · · · · · · · · · · · · · · ·		
Plugins > Version Control					
> Version Control		Debugger:	Use Cygwin GDB		
 Build, Execution, Deployment Toolchains 			^(b) Version: GNU gdb (GDB) (Cygwin 7.10.1-1) 7.10.1		
CMake 1					
> Debugger					
Python Debugger					
Python Interpreter					
> Deployment					
> Console					
> Languages & Frameworks					
> Tools					
0				OK Cancel Ap	ply

Figure 30: CLion detecting CygWin Libraries

15. Restart CLion. The old project would have been opened under the old settings. Open a new project. You should now be able to run it without issues. The very first time you build a project with CLion, it will take a little time to index all the libraries. Please wait until the little arrow on the top right goes green before running.

8.2 On a Mac

CLion usually installs without trouble on a Mac. Some older versions of OSX might have issues due to the different versions of the C++ compiler. OSX does not allow you to directly update the Clang tools needed for this, but you can re-install the XCode developer tools, which will get all the updates you need indirectly. Please go through the following steps.

1. Open the Terminal.

- 2. Re-install XCode developer tools by typing xcode-select --install and press Enter.
- Accept the XCode user license. On the terminal, type sudo xcodebuild -license Enter the admin password for the computer when prompted. Please not that you will not see the password being typed in. This is ok. Type in the password and press Enter.
- 4. The license will then be presented on screen. Hit the space bar to scroll through the text. When you get to the end of the text, you'll be prompted to enter "agree". Type this in and hit Enter.
- 5. Restart CLion, click on File ->Reload CMake Project and everything should work.