

In a Ubuntu 20.04 LTS VM

```
sudo apt-key adv --fetch-keys http://repos.codelite.org/CodeLite.asc
```

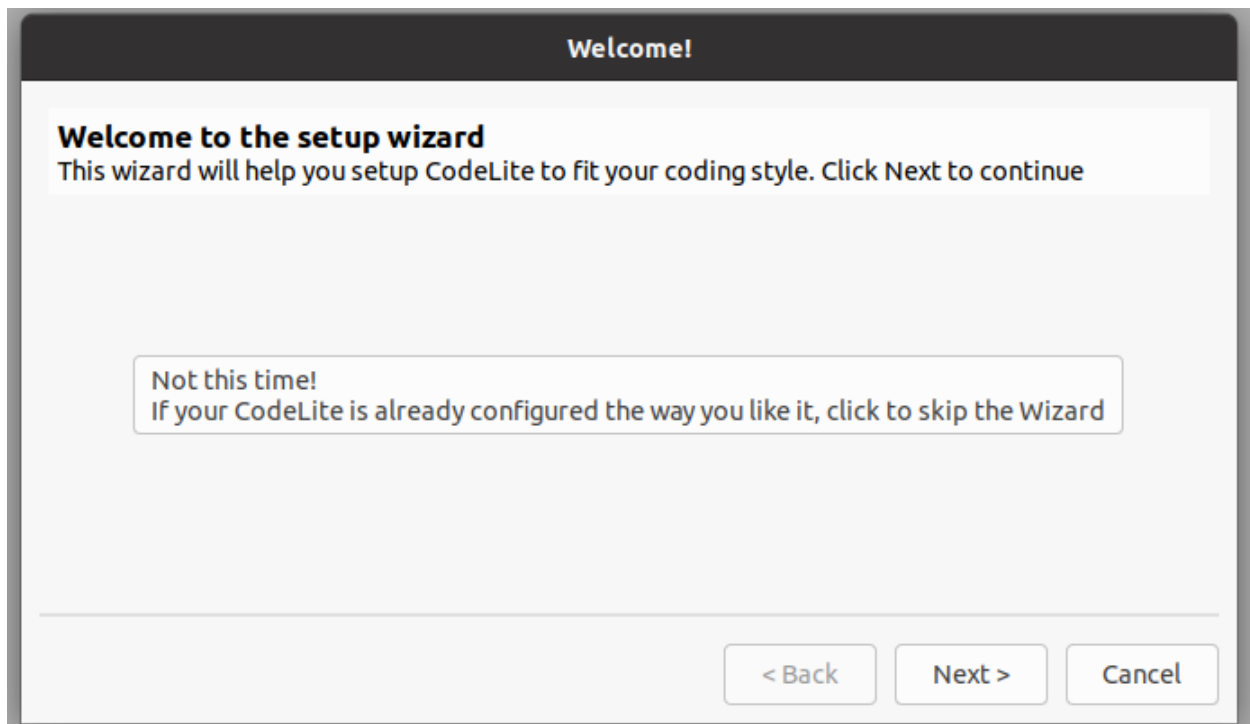
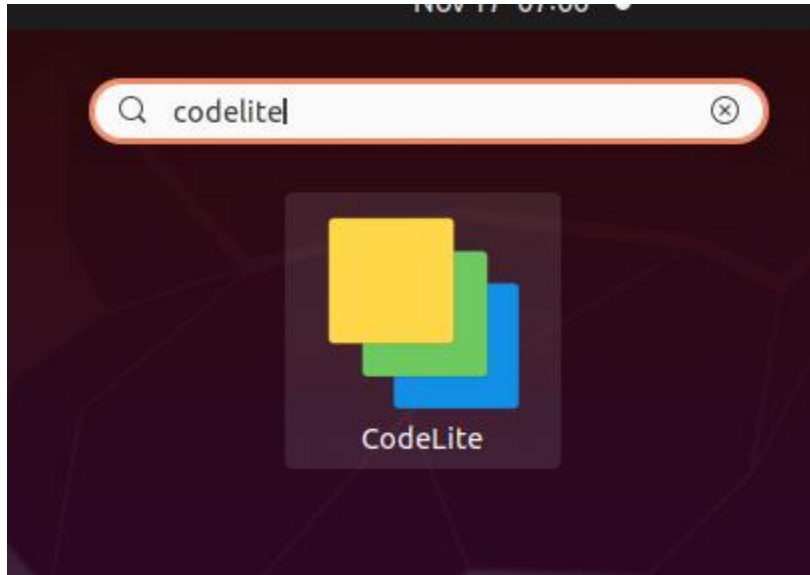
```
sudo apt-add-repository "deb http://repos.codelite.org/ubuntu/ $(lsb_release  
-sc) universe"
```

```
sudo apt-get update
```

```
sudo apt-get install codelite=15.0*
```

## CodeLite

One of the better unknown IDEs with support for CMake is CodeLite. Everybody has their favorite. CodeBlocks is also installed in the DEV VM but we will cover configuring CodeLite now. The first run goes like this.



**Welcome!**

**Development Profile**  
Select the profile that best describes you

☐ Default (Don't change the current settings)

☐ Both C/C++ and Web development

☒ C/C++ development

☐ C/C++ development (Blockchain using EOSIO)

☐ Web development (PHP, JS etc)

< Back

Next >

Cancel

Be sure to set for just C/C++.

**Welcome!**

**Setup compilers**  
Let CodeLite configure your installed compilers or help you install one

Scan  
Click to scan your computer for installed compilers

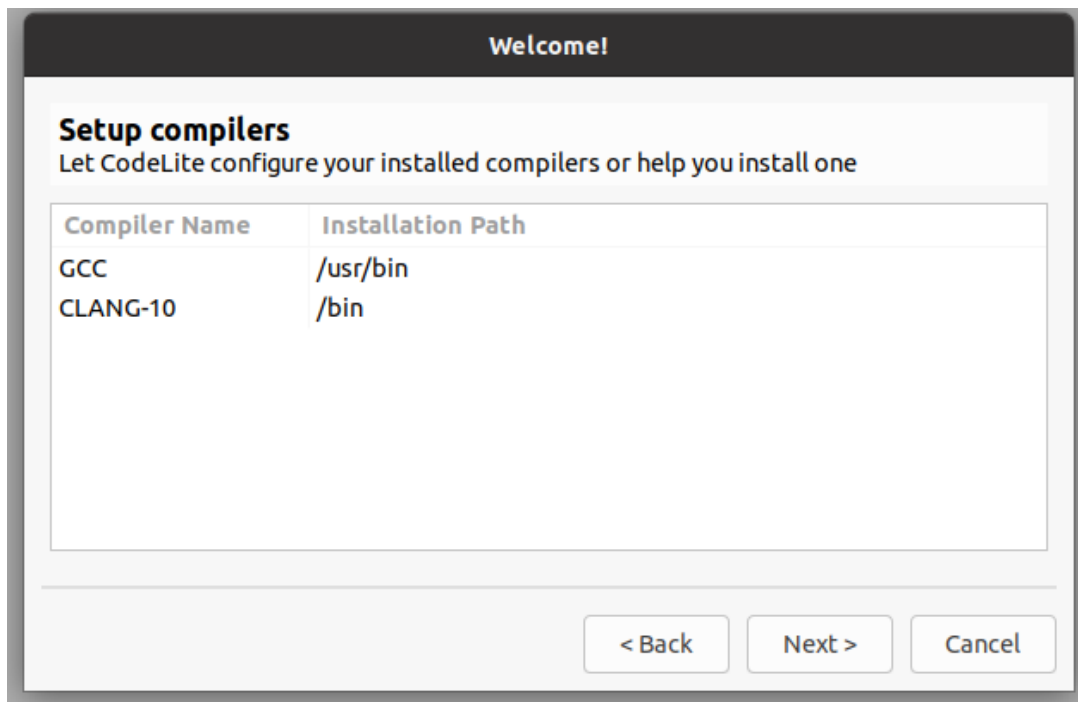
Install  
Click to download a MinGW compiler

< Back

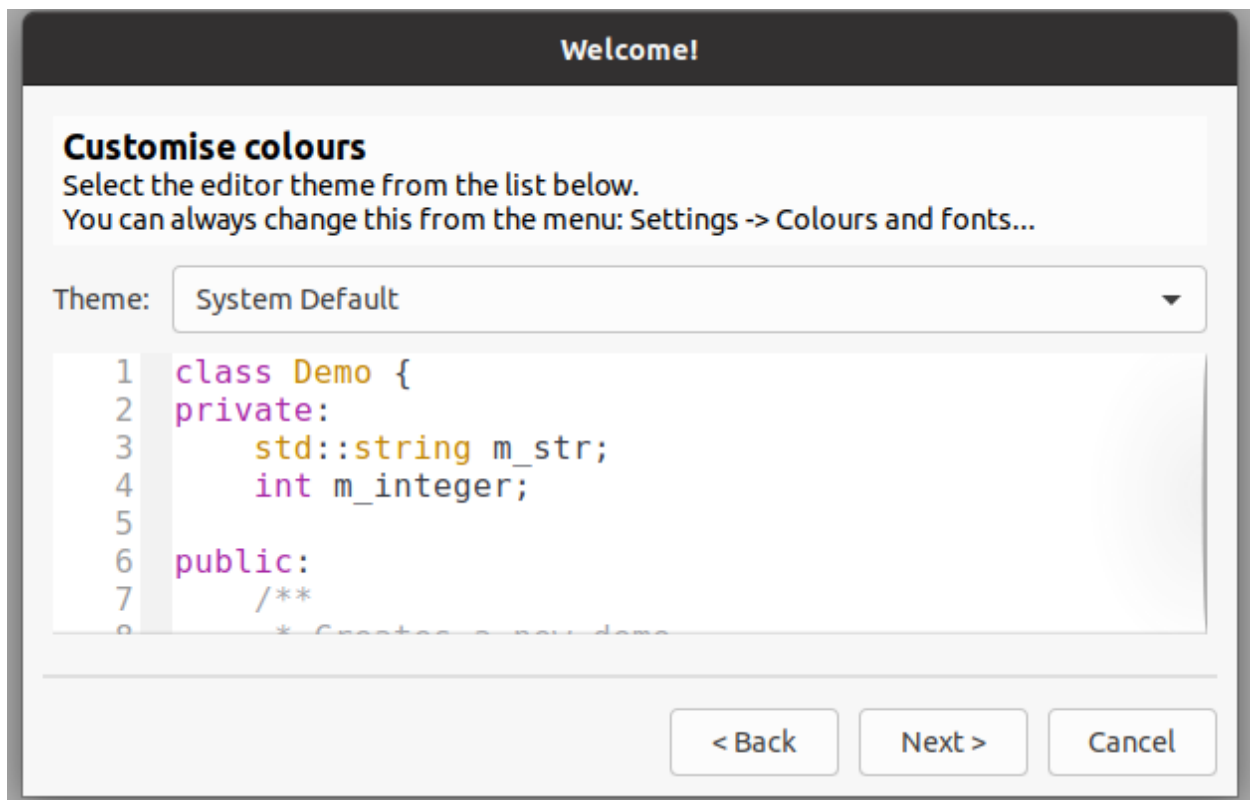
Next >

Cancel

Choose Scan.



Click Next.



Choose a theme.

Welcome!

### Whitespace & Indentation

Should CodeLite use TABS or SPACES for indentation?

Indentation

☒ Indent using SPACES

☐ Indent using TABS

Whitespace Visibility:

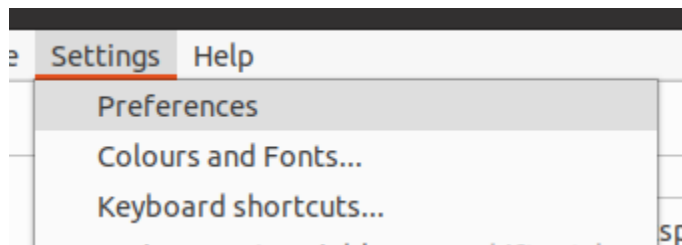
☐ Invisible

☒ Visible always

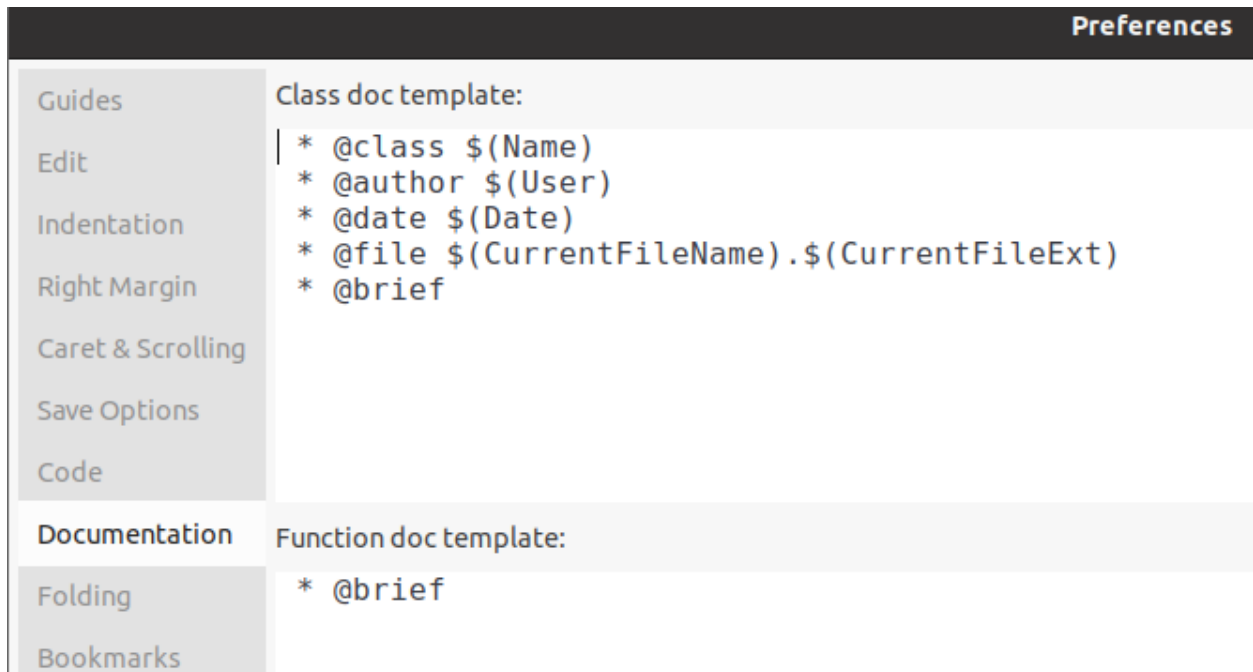
☐ Visible after indentation

< Back   Finish   Cancel

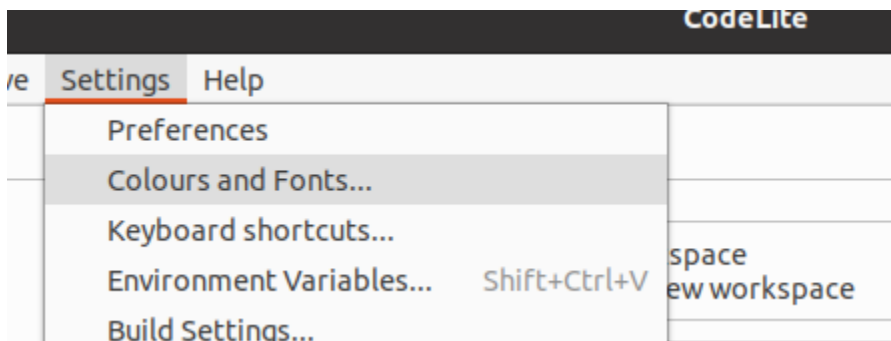
Be certain we use spaces and have visible whitespace.



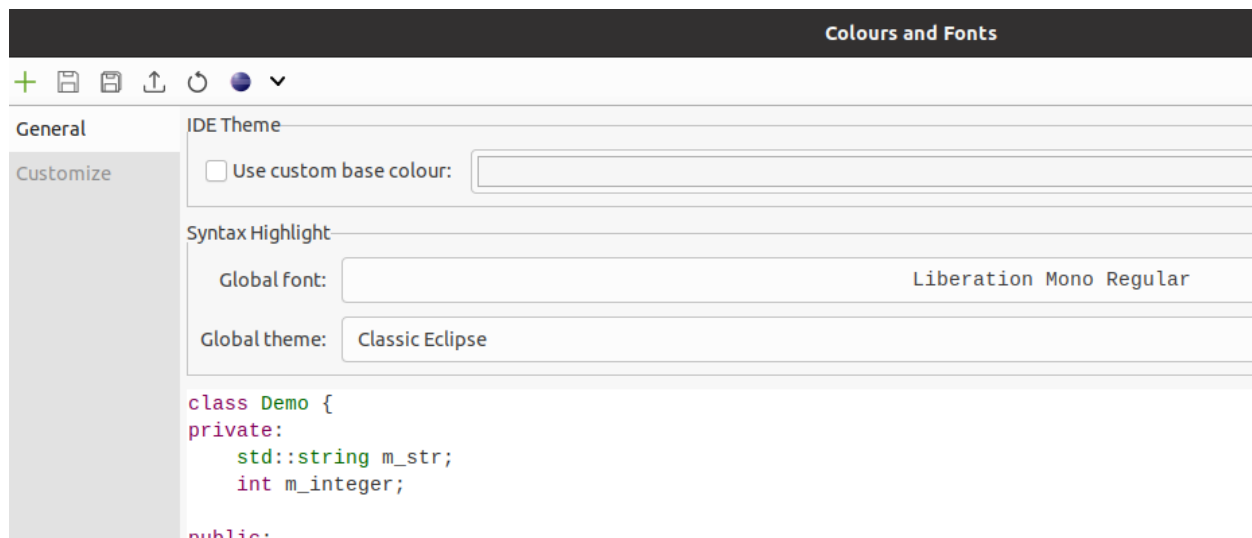
Settings->Preferences



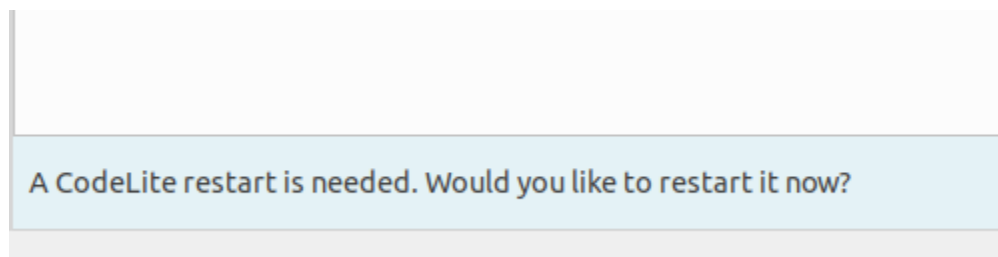
Your standard class header template should be updated to include copyright and whatever else one deems needs to be in each and every file. It pulls User and Date from the environment.



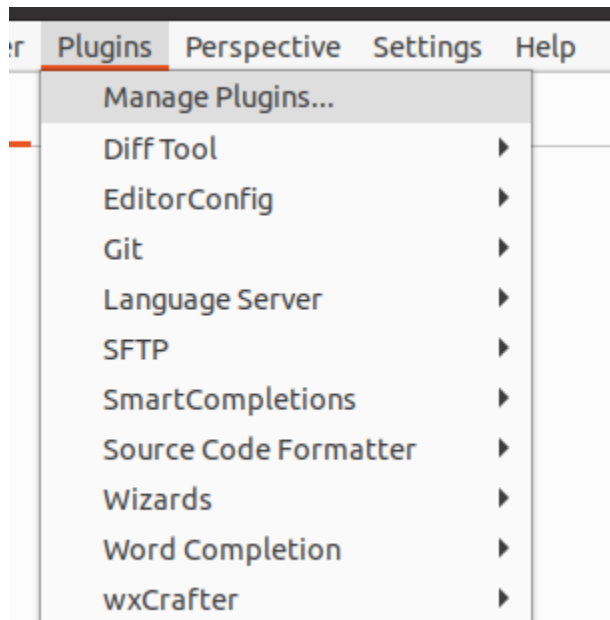
Settings->Colours and Fonts



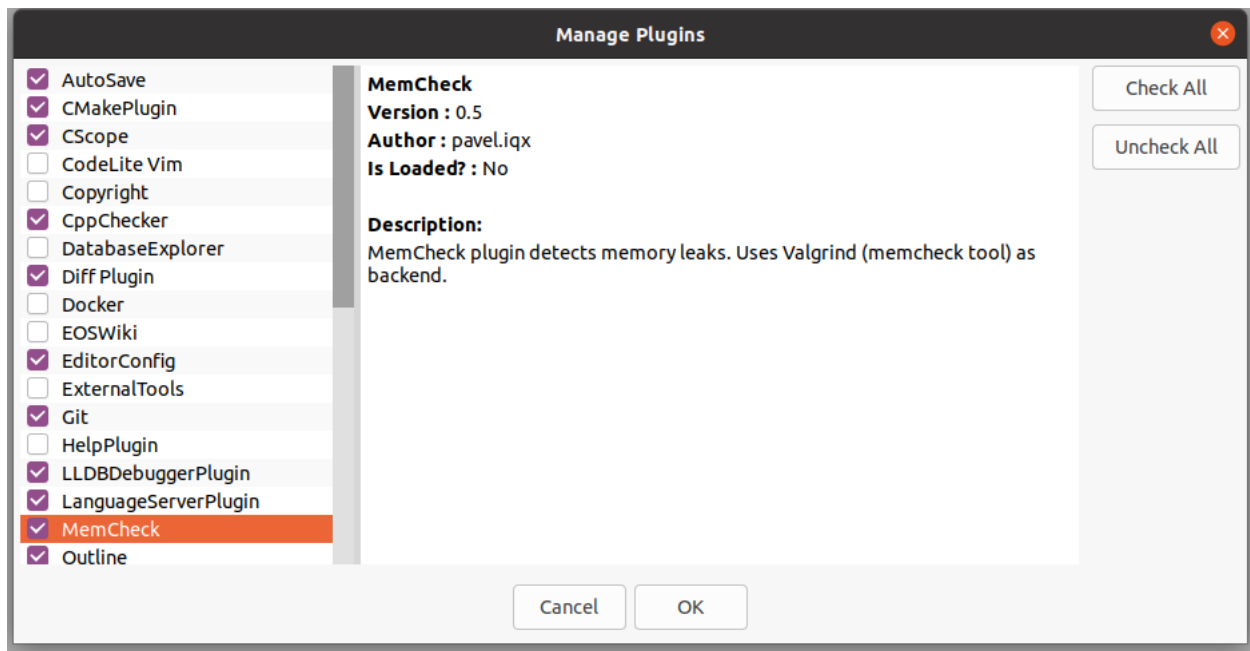
Choose a font and theme you like. **Do not import themes!**



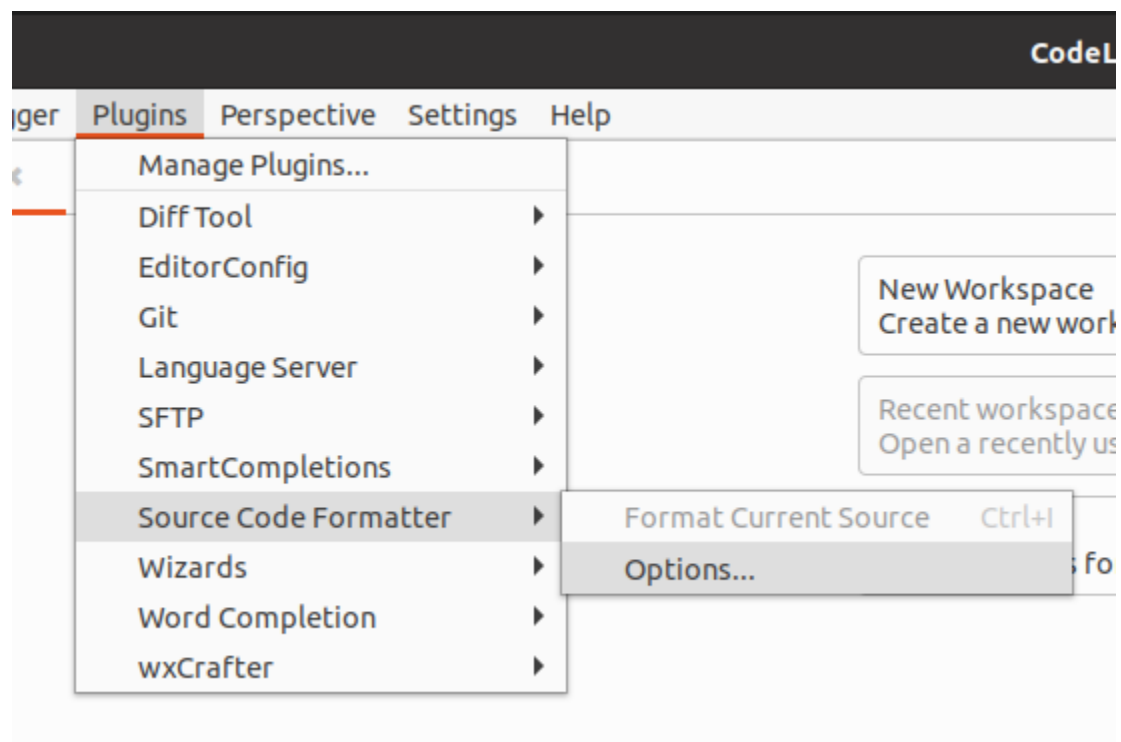
By now it will want to restart. Exit CodeLite and restart.



Plugins->Manage Plugins



Enable these. The ones that aren't shown in this image were left as defaults.



Plugins->Source Code Formatter->Options

GeneralC++PHP

☒ Format editor on file save

C++ formatter: AStyle

PHP formatter: Builtin

Change to AStyle and check “Format editor on file save”

Source Code Form

GeneralC++PHP

clang-formatAStyle

PreDefined Styles	ANSI
Brackets	Linux
<div> <div>Indentation</div> <div>Formatting</div> </div>	<div>Preprocessors</div> <div>Break Blocks, Break Blocks All, Pad Parenthesis</div>
Break Blocks	<input checked="" type="checkbox"/>
Pad Parenthesis	<input type="checkbox"/>
Break Blocks All	<input checked="" type="checkbox"/>
Pad Parenthesis Outside	<input type="checkbox"/>
Break else-if	<input type="checkbox"/>
Pad Parenthesis Inside	<input checked="" type="checkbox"/>
Pad Operators	<input type="checkbox"/>
UnPad Parenthesis	<input type="checkbox"/>
One Line Keep Statement	<input type="checkbox"/>
Fill Empty Lines	<input type="checkbox"/>
One Line Keep Blocks	<input checked="" type="checkbox"/>

Brackets

Bracket Style options define the bracket style to use

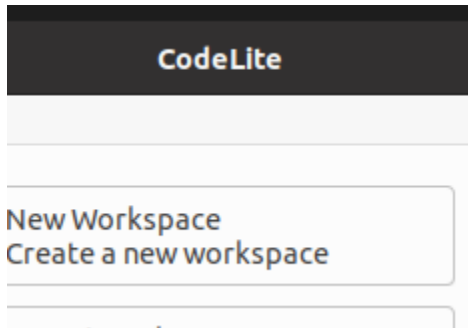
AStyle Only: Custom user settings

#  
# Do not let source lines go beyond this many characters  
#  
--max-code-length=130  
--break-blocks  
--convert-tabs  
--break-one-line-headers  
--pad-header  
--break-blocks

Check the boxes to match above. Most important is to paste in the contents of our .astylerc. This plug-in doesn't have all of the switches we need.

Again, the content of this plugin and .astylerc will change based on the formal coding style based on The Barr Group coding standard.

Apply and close.



New Workspace

Workspace Path:

/home/developer/Projects

Workspace Name:

codelite

Generated File:

/home/developer/Projects/codelite/codelite.workspace

☒ Create the workspace under a separate directory

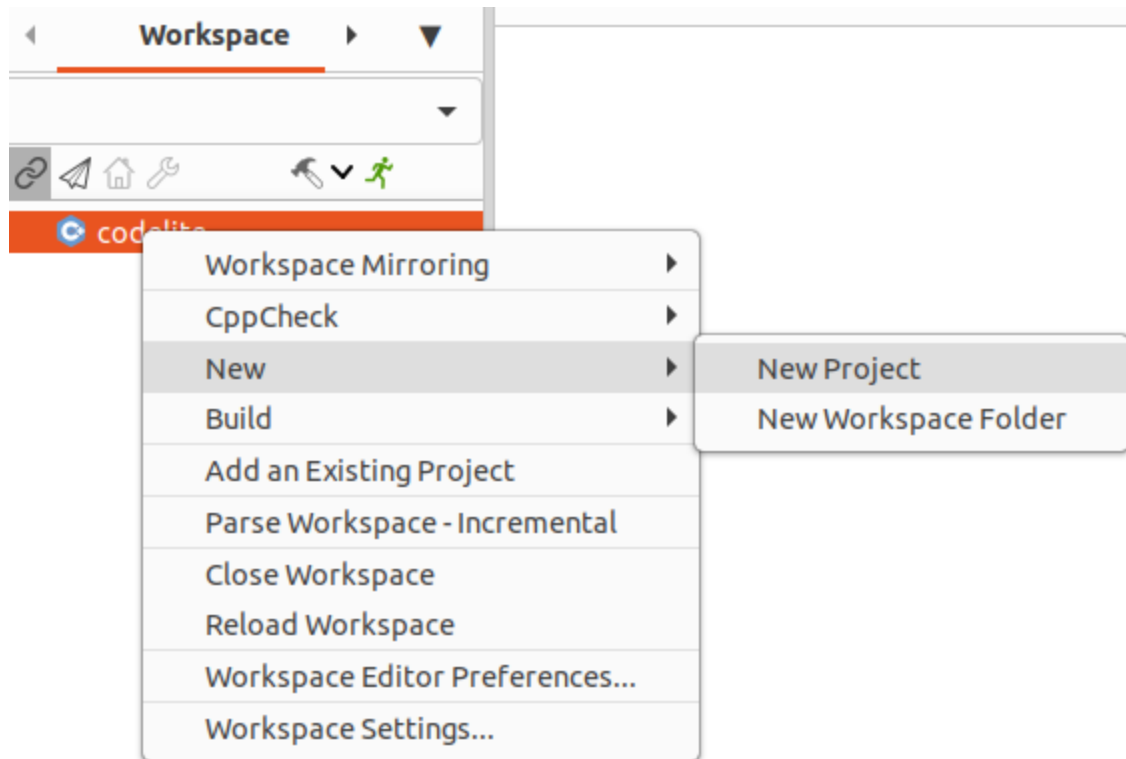
Cancel

OK

Put a codelite workspace under our Projects directory. Yes, CodeLite will want to restart right about now.

A CodeLite restart is needed. Would you like to restart it now?

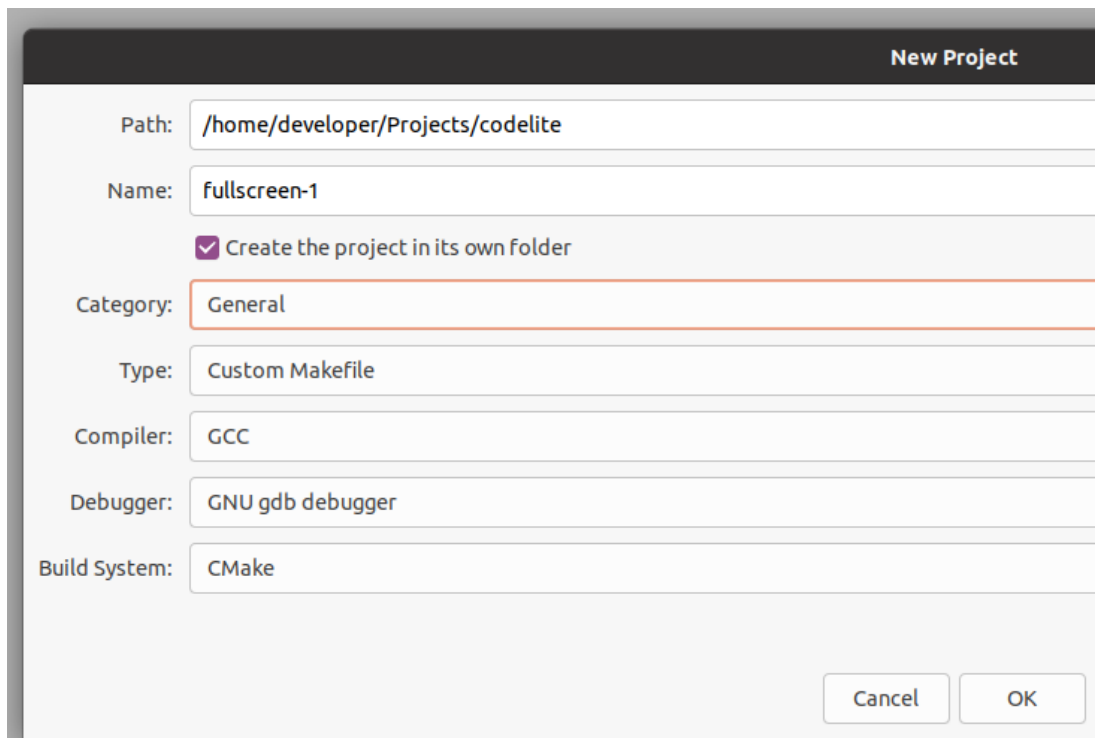
Please do.



Right click on codelite and navigate to “New Project”

It’s worth pointing out the “workspace” concept at this point in time. You see “Workspace Editor Preferences” and “Workspace Settings” on the submenu. That’s because a workspace isn’t a project, it’s a working style. Each workspace can have its own coding style, tab settings, C++ document template. If you are writing code for Spacely Sprockets in the morning you create a Spacely Sprockets workspace that has their copyright information in the template, coding style, and other settings. In the afternoon you might be coding for Really Cool Games Inc. so you have a different workspace for them.

One of the things not yet contained within these workspace specific things is your Git credentials. That will be coming at some point.



**New Project**

Path: /home/developer/Projects/codelite

Name: fullscreen-1

☒ Create the project in its own folder

Category: General

Type: Custom Makefile

Compiler: GCC

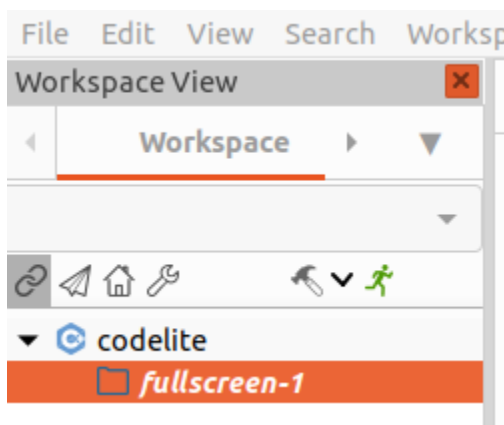
Debugger: GNU gdb debugger

Build System: CMake

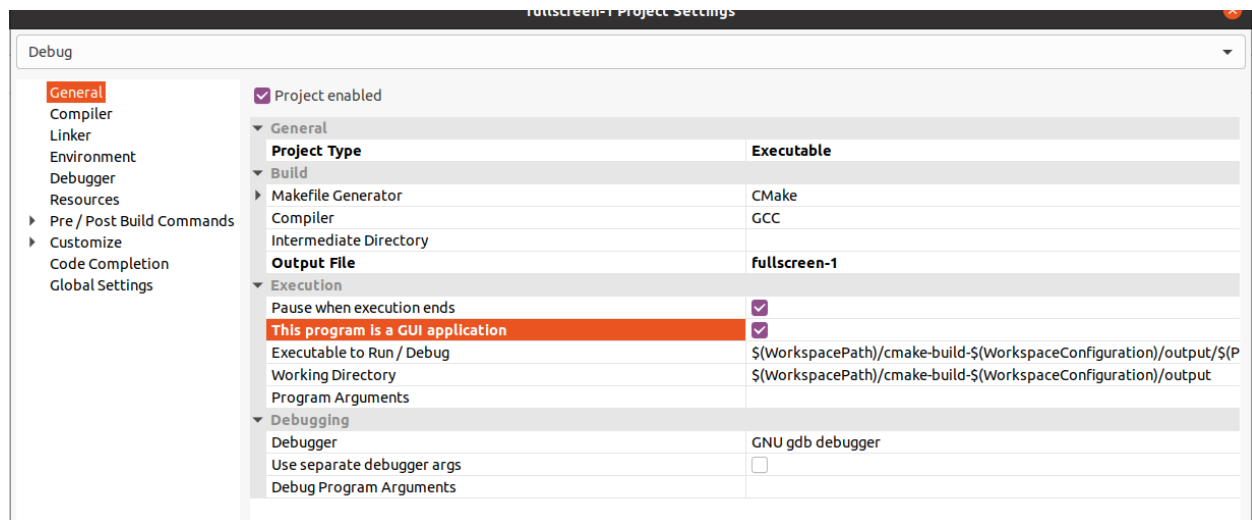
Cancel OK

We cannot use the GUI category here because NanoGUI does not yet have an entry in Type. Choose “General”, “Custom Makefile”, GCC, “GNU gdb”, and CMake. That last one is really important. Everything is moving towards CMake and that works the best in containers.

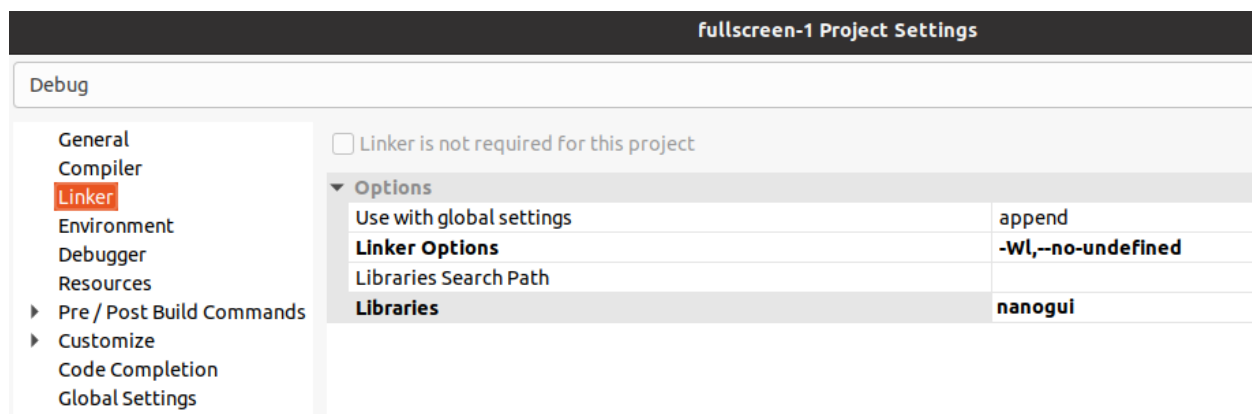
It used to be this IDE allowed you to create your CMakeLists.txt file by hand. Now that they have all of those “types” they want to control the content of that file, generating it as needed. Click on the little wrench tool button with fullscreen-1 highlighted.

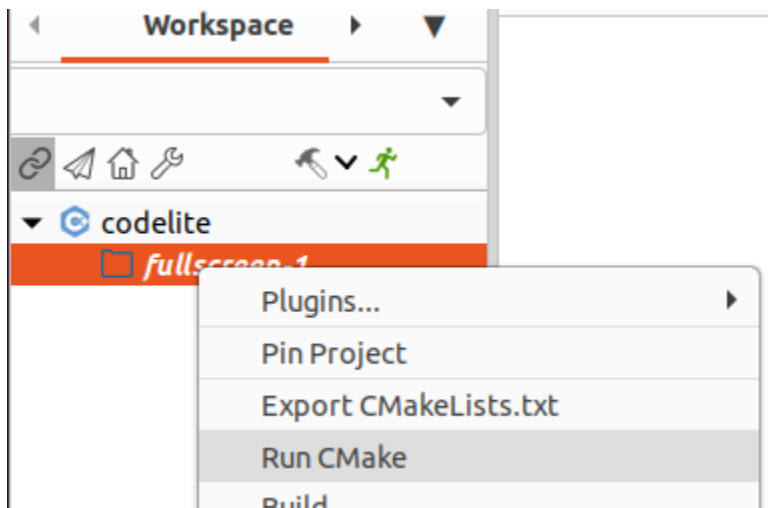


Fill in the General information as follows:



Checking the GUI application box tells CodeLite to launch application without a console wrapper. Other than that you should only have to fill in the “Output File”.





Right click on our project and choose “Run CMake”. You will see stuff like this scroll past in the build window.

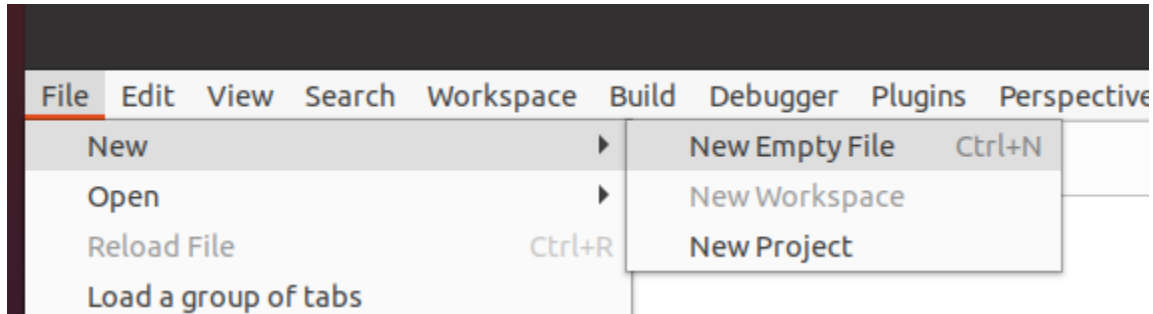
```
cmake /home/developer/Projects/codelite/fullscreen-1
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/developer/Projects/codelite/cmake-build-Debug/fullscreen-1
==== Done ====
```

Any subsequent time you “Run CMake” you will only see something like this.

```
cmake /home/developer/Projects/codelite/fullscreen-1
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/developer/Projects/codelite/cmake-build-Debug/fullscreen-1
==== Done ====
```

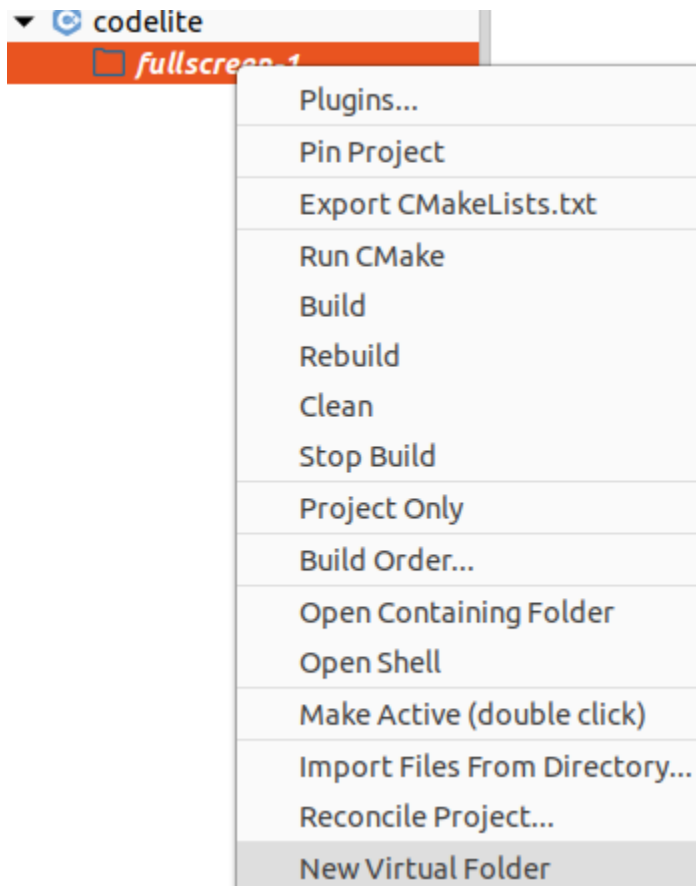
That's because it caches all of the previous information.

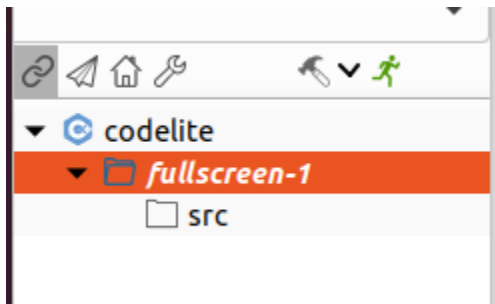
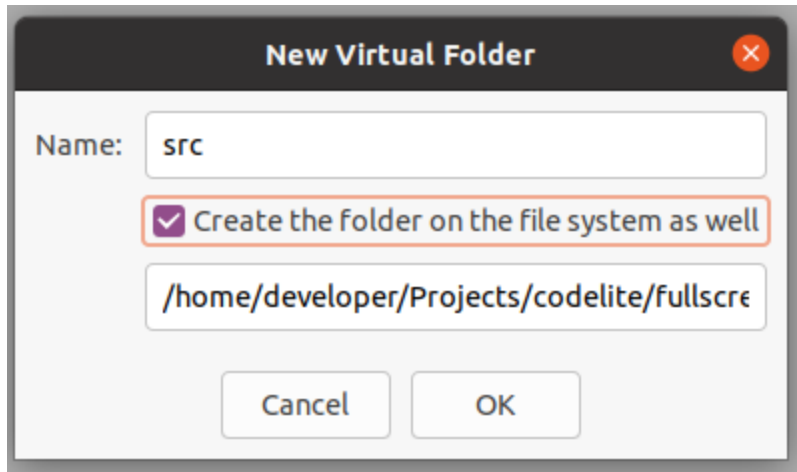
```
developer@KneeVoice-developer-VirtualBox:~/Projects/codelite$ ls cmake-build-Debug/
fullscreen-1
developer@KneeVoice-developer-VirtualBox:~/Projects/codelite$ ls cmake-build-Debug/fullscreen-1/
CMakeCache.txt CMakeFiles cmake_install.cmake Makefile
developer@KneeVoice-developer-VirtualBox:~/Projects/codelite$
```



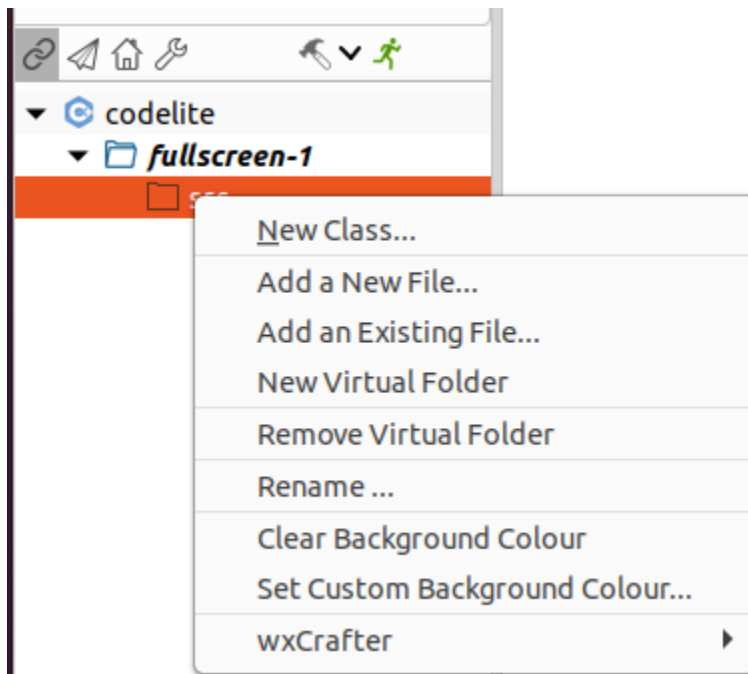
Previous iterations of CodeLite used to create a main.cpp for “Hello World” but the current version does not.

Right click and select “New Virtual Folder”

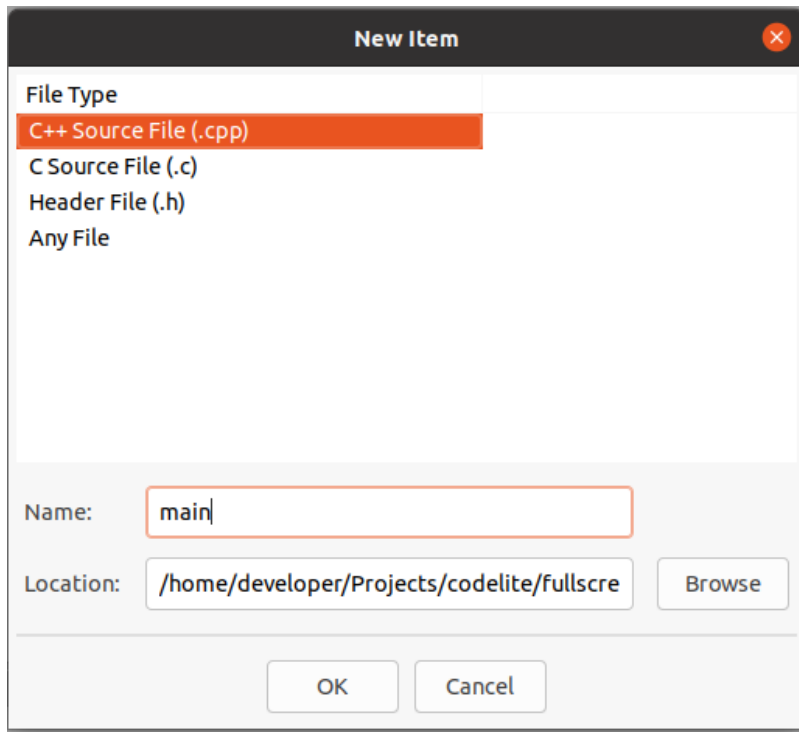




Right click on the new virtual folder and view the menu.



Previous versions of CodeLite used to automatically create this folder and put a main.cpp for “Hello World” in it. Our current one does not. Choose “Add a New File.”



After clicking “OK” you will see main.cpp opened in the edit window. Paste in the following text.

```
#include <iostream>

#include "mainwindow.h"

using namespace nanogui;

int main(int /* argc */, char ** /* argv */)
{
    nanogui::init();

    /* cheat by using scoped variables */
    {
        MainWindow *mw = new MainWindow();

        mw->set_visible(true);
        mw->perform_layout();

        nanogui::mainloop(-1);
    }

    nanogui::shutdown();
    return 0;
}
```

Save the file and you will see everything the editor believes is wrong.

```

src/main.cpp
1  #include <iostream>
2
3  #include "mainwindow.h"
4
5  using namespace nanogui;
6
7  int main(int /* argc */, char ** /* argv */)
8  {
9      nanogui::init();
10
11      /* cheat by using scoped variables */
12      {
13          MainWindow *mw = new MainWindow();
14
15          mw->set_visible(true);
16          mw->perform_layout();
17
18          nanogui::mainloop(-1);
19      }
20
21      nanogui::shutdown();
22      return 0;
23  }
24

```

Right click on the src folder and choose “New Class”.

New Class

Class Name:

Namespace:

Inherits:

File name:

Block Guard:

Virtual Directory:

Path:

File

☐ Create .hpp instead of .h
☒ Use lowercase file names

☒ Use #pragma once

Cancel

OK

We use lowercase file names and #pragma once. After clicking “OK” navigate to the header file tab and paste replace what is there with the following:

```
#pragma once

#include "nanogui.h"

using namespace nanogui;

class MainWindow
{
public:
    MainWindow();
    ~MainWindow();

    void set_visible( bool yesNo);
    void perform_layout();

private:
    void add_navigation( Window *win);
    void navigate_to( int idx);

    Screen      *m_screen;
    FormHelper  *m_gui;
    Window      *m_win1;
    Window      *m_win2;
    Window      *m_win3;
};
```

Ignore all of the red arrows you see on save. Now navigate to mainwindow.cpp tab and replace what is there with the following:

```
#include "mainwindow.h"

using namespace nanogui;

MainWindow::MainWindow()
{
    m_screen      = new Screen(Vector2i(1024, 768), "Full Screen Test",
                               /* resizable */ false, /* fullscreen */ true);

    m_gui         = new FormHelper( m_screen);
    m_win1        = m_gui->add_window( m_screen->size());
    Label *lbl1   = new Label(m_win1, "This is Window 1");

    lbl1->set_color( Color(0, 0, 255, 1));
    lbl1->set_position(Vector2i(100, 100));
    m_win1->add_child(lbl1);

    add_navigation(m_win1);

    m_win2        = m_gui->add_window( m_screen->size());
    Label *lbl2   = new Label(m_win2, "This is Window 2");
```

```

    lbl2->set_color( Color(0, 188, 0, 1));
    lbl2->set_position(Vector2i(200, 200));
    m_win2->add_child(lbl2);

    add_navigation(m_win2);

    m_win3      = m_gui->add_window( m_screen->size());
    Label *lbl3 = new Label(m_win3, "This is Window 3");

    lbl3->set_color( Color(0, 188, 0, 1));
    lbl3->set_position(Vector2i(300, 300));
    m_win3->add_child(lbl3);

    add_navigation(m_win3);
}

MainWindow::~MainWindow()
{
    /*
     * Children are supposed to be deleted as long as they are parented.
     * TODO:: run valgrind to be certain of this.
     */
}

void MainWindow::add_navigation(Window *win)
{
    ComboBox *cb = new ComboBox(win,
                                {"Example Window 1", "Example Window 2",
                                "Example Window 3"},
                                {"Window 1", "Window 2", "Window 3"});
    cb->set_callback( []() { navigate_to( selected_index());});
    cb->set_position( Vector2i(500,500));

    win->add_child(cb);
}

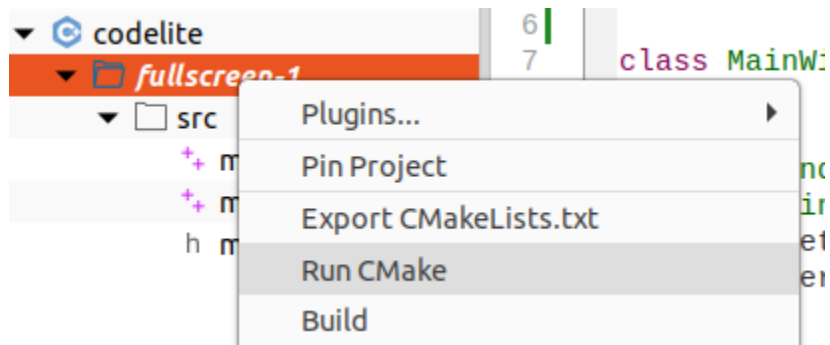
void MainWindow::navigate_to(int idx)
{
    switch( idx)
    {
    case 0:
        m_gui->set_window( m_win1);
        break;
    case 1:
        m_gui->set_window( m_win2);
        break;
    default:
        m_gui->set_window( m_win3);
        break;
    }
}

void MainWindow::set_visible(bool yesNo)
{
    m_screen->set_visible( yesNo);
}

```

```
}  
  
void MainWindow::perform_layout()  
{  
    m_screen->perform_layout();  
}
```

Again, ignore the red arrows when you save. Right click on the project and choose “Run CMake” again.



The IDE doesn't know about new things until they've been added to CMakeLists.txt.