N'GINE – CODE::BLOCKS Configuration

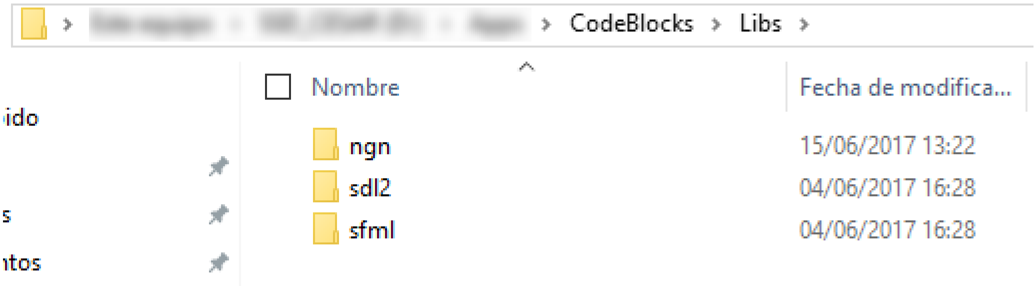
Installation of Libraries – Windows

We will begin by downloading the additional libraries from their official websites:

SDL2: https://www.libsdl.org/download-2.0.php

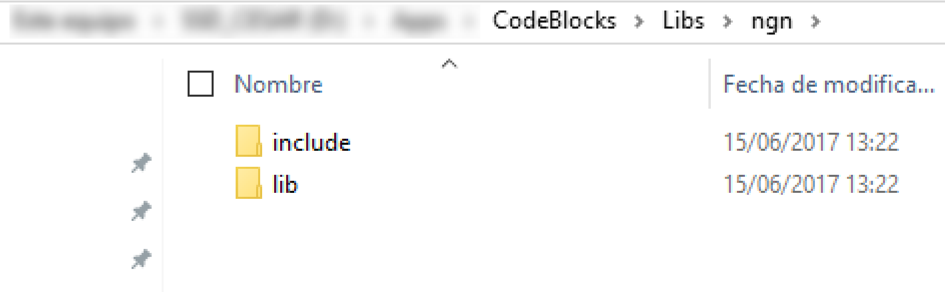
SFML: https://www.sfml-dev.org/

Next, we will create a folder named "Libs" within the directory where CODE::BLOCKS is installed. Inside this folder, we will create the "ngn," "sdl2," and "sfml" folders.



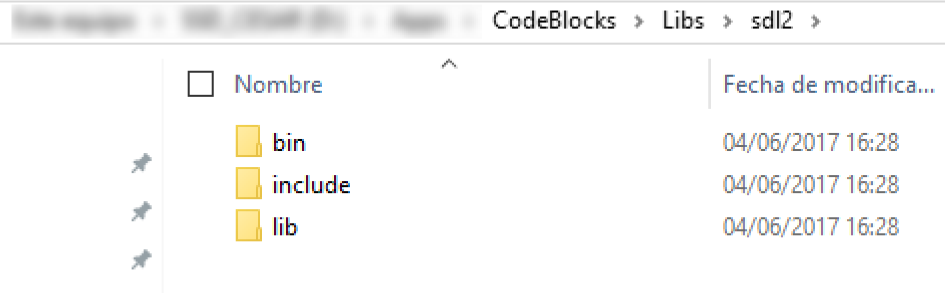
Nombre	Fecha de modifica...
ngn	15/06/2017 13:22
sdl2	04/06/2017 16:28
sfml	04/06/2017 16:28

We will copy the “include”, “lib”, and optionally “bin” directories corresponding to each library into each folder.



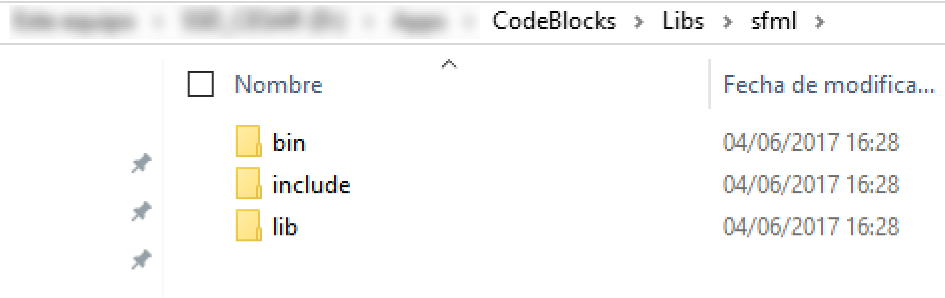
CodeBlocks › Libs › ngn ›

□ Nombre	Fecha de modifica...
include	15/06/2017 13:22
lib	15/06/2017 13:22



CodeBlocks › Libs › sdl2 ›

□ Nombre	Fecha de modifica...
bin	04/06/2017 16:28
include	04/06/2017 16:28
lib	04/06/2017 16:28

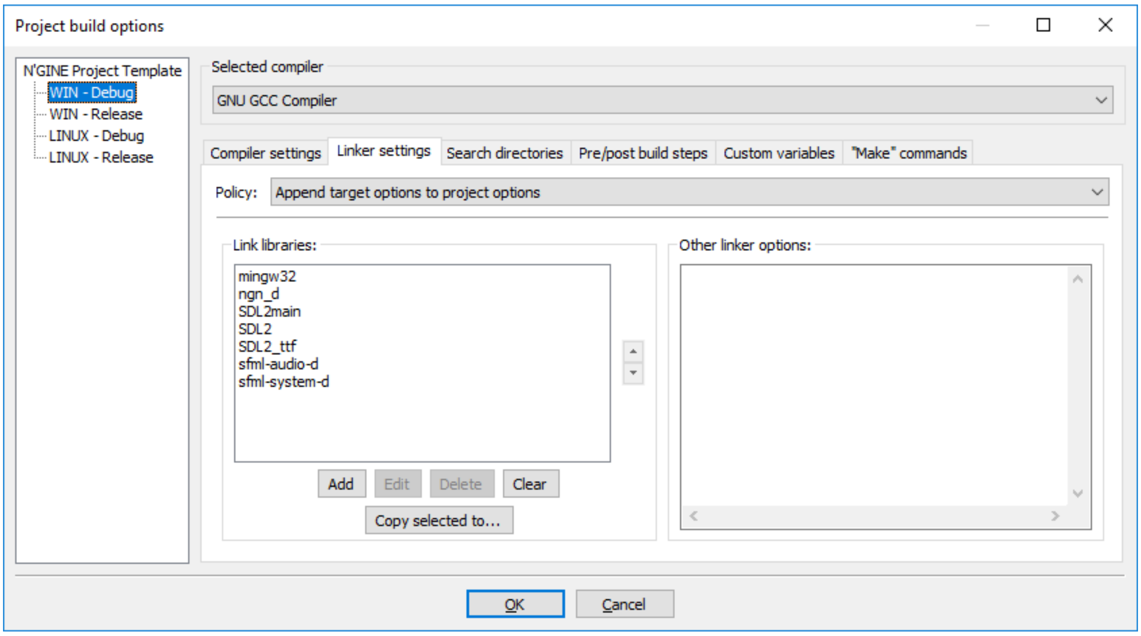


CodeBlocks › Libs › sfml ›

□ Nombre	Fecha de modifica...
bin	04/06/2017 16:28
include	04/06/2017 16:28
lib	04/06/2017 16:28

Project Configuration - CODE::BLOCKS

In “Project build options”, in the “WIN-DEBUG” section, we will configure the following parameters:"



Selected compiler: GNU GCC Compiler

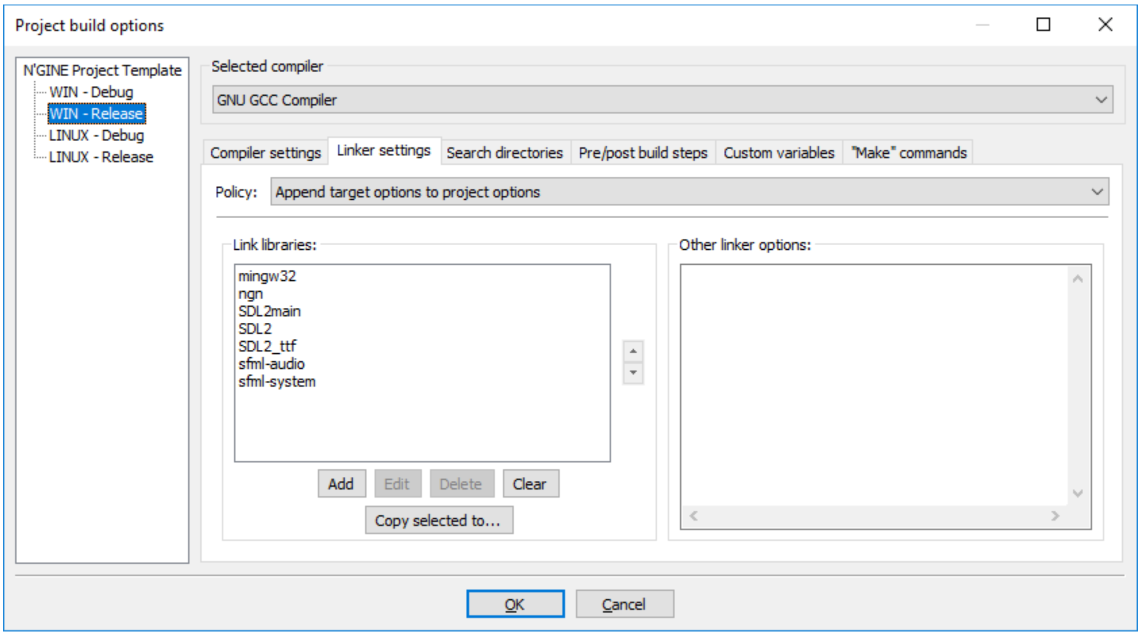
Policy: Append target options to project options

Link libraries:

```
mingw32
ngn_d
SDL2main
SDL2
SDL2_ttf
sfml-audio-d
sfml-system-d
```

Other linker options:

In “Project build options”, in the “WIN-RELEASE” section, we will configure the following parameters:



Selected compiler: GNU GCC Compiler

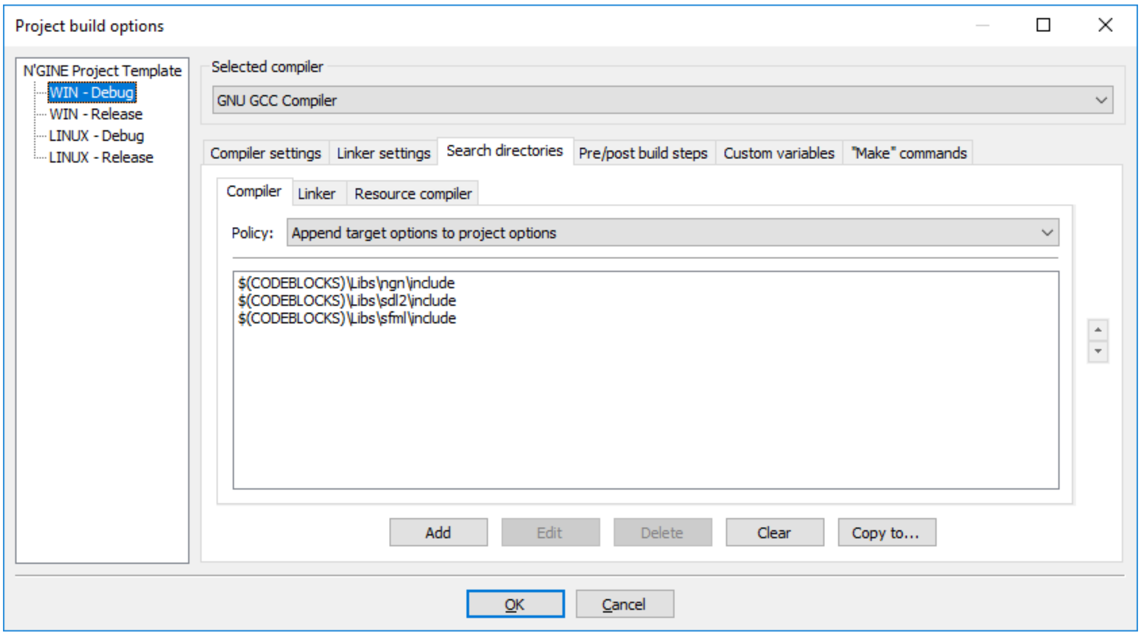
Policy: Append target options to project options

Link libraries:

```
mingw32
ngn
SDL2main
SDL2
SDL2_ttf
sfml-audio
sfml-system
```

Other linker options:

In “Project build options”, we will configure the following parameters in both the “WIN-DEBUG” and “WIN-RELEASE” sections:



Project build options

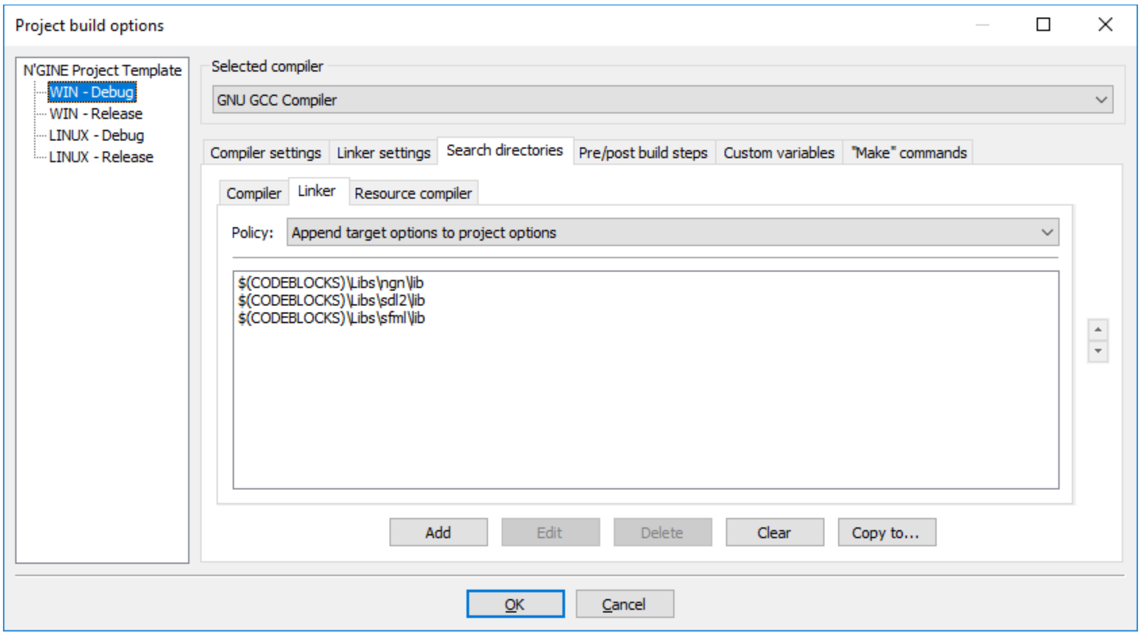
N'GINE Project Template
WIN - Debug
WIN - Release
LINUX - Debug
LINUX - Release

Selected compiler: GNU GCC Compiler

Search directories > Compiler

Policy: Append target options to project options

```
$(CODEBLOCKS)\Libs\ngn\include
$(CODEBLOCKS)\Libs\sdl2\include
$(CODEBLOCKS)\Libs\sfml\include
```



Project build options

N'GINE Project Template
WIN - Debug
WIN - Release
LINUX - Debug
LINUX - Release

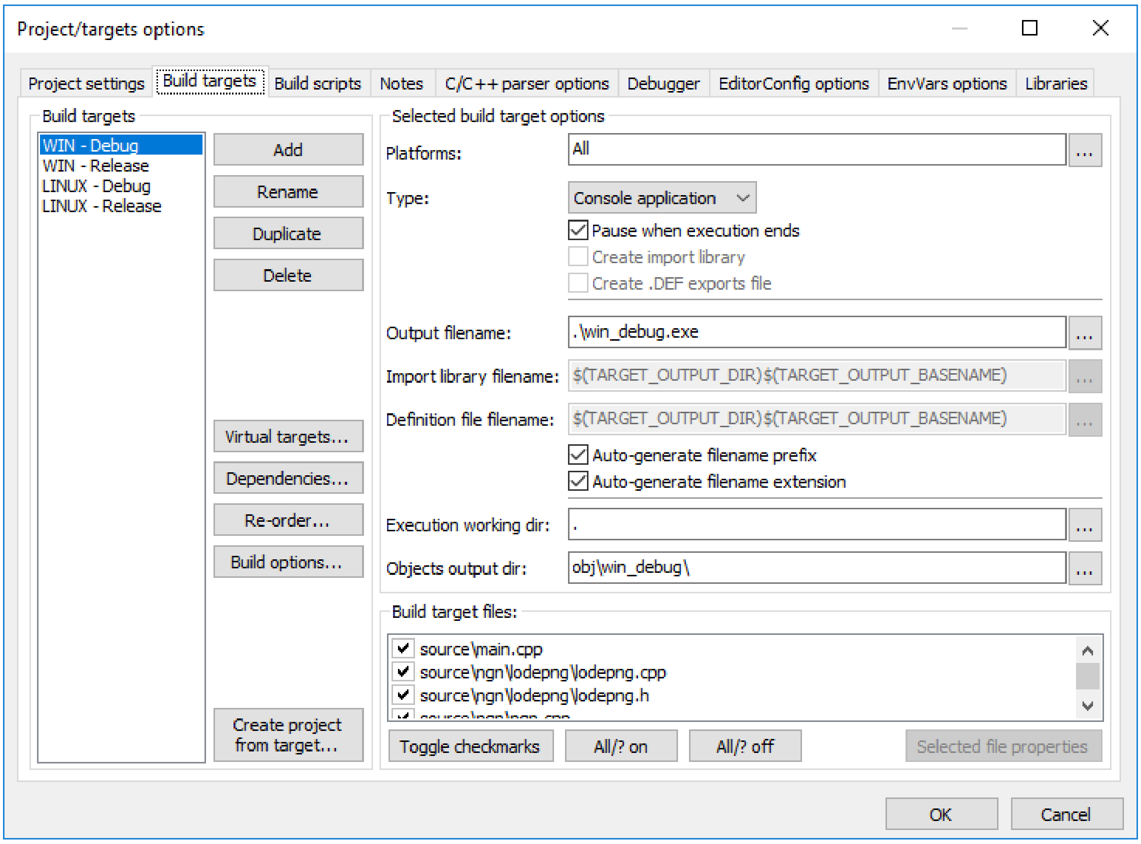
Selected compiler: GNU GCC Compiler

Search directories > Linker

Policy: Append target options to project options

```
$(CODEBLOCKS)\Libs\ngn\lib
$(CODEBLOCKS)\Libs\sdl2\lib
$(CODEBLOCKS)\Libs\sfml\lib
```

In “Project/target options”, we will configure the following parameters:



Build targets

- WIN - Debug
- WIN - Release
- LINUX - Debug
- LINUX - Release

Selected build target options

Platforms: All

Type: Console application

☑ Pause when execution ends
□ Create import library
□ Create .DEF exports file

Output filename: .\win_debug.exe

Import library filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

Definition file filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

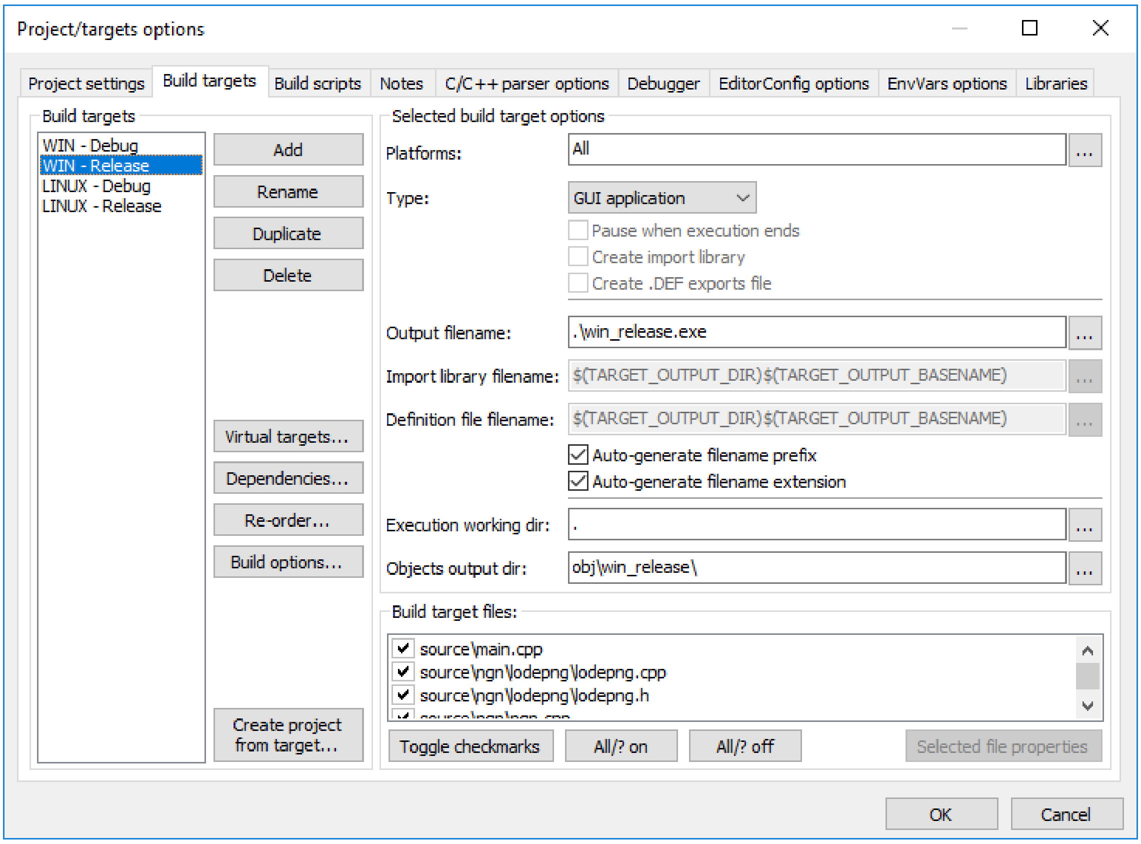
☑ Auto-generate filename prefix
☑ Auto-generate filename extension

Execution working dir: .

Objects output dir: obj\win_debug\

Build target files:

☑ source\main.cpp
☑ source\ngn\lodepng\lodepng.cpp
☑ source\ngn\lodepng\lodepng.h
☑ source\ngn\ngn.cpp



Build targets

- WIN - Debug
- WIN - Release
- LINUX - Debug
- LINUX - Release

Selected build target options

Platforms: All

Type: GUI application

□ Pause when execution ends
□ Create import library
□ Create .DEF exports file

Output filename: .\win_release.exe

Import library filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

Definition file filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

☑ Auto-generate filename prefix
☑ Auto-generate filename extension

Execution working dir: .

Objects output dir: obj\win_release\

Build target files:

☑ source\main.cpp
☑ source\ngn\lodepng\lodepng.cpp
☑ source\ngn\lodepng\lodepng.h
☑ source\ngn\ngn.cpp

Installation on CODE::BLOCKS 20 - WINDOWS

Version 20 of CODE::BLOCKS includes the latest version of the TDM-GCC MinGW compiler. If you have installed the version of CODE::BLOCKS that comes with this compiler or if you don't have it installed (clean CODE::BLOCKS installation), you will need to install or update the compiler version. This is necessary because the latest stable version of the SFML library is compiled with MinGW-windows-gcc-8.1.0, and it won't work on versions older than this. To perform this installation or update, follow these steps:

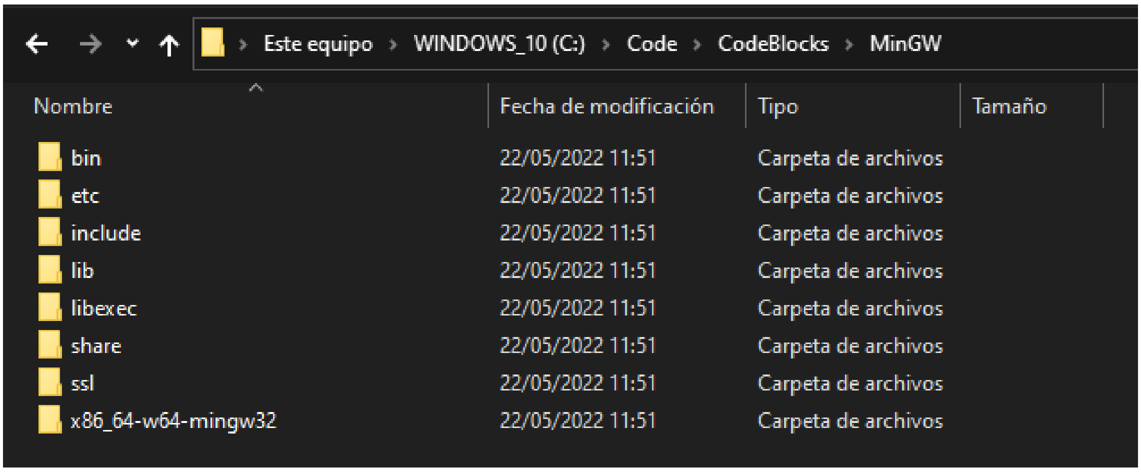
1 – Download the compiler installer from the following link:

https://www.mingw-w64.org/

(As of this review, I installed version 11.3.0 using MSYS2: https://www.mingw-w64.org/downloads/#msys2)

2 – In the CODE::BLOCKS installation folder, delete all contents of the MinGW folder. If this folder doesn't exist, create it.

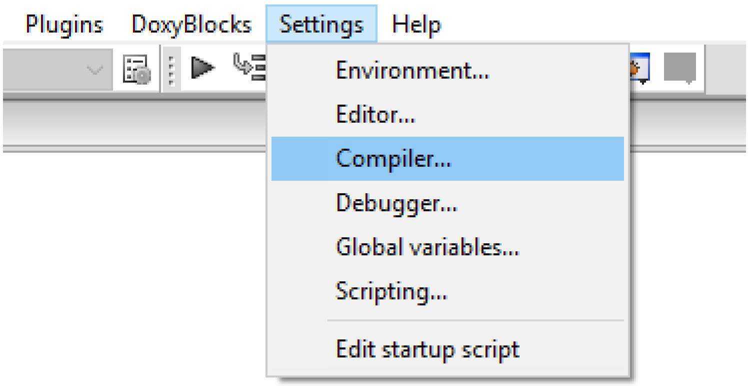
3 – Copy all contents from the downloaded compressed file into the MinGW folder.



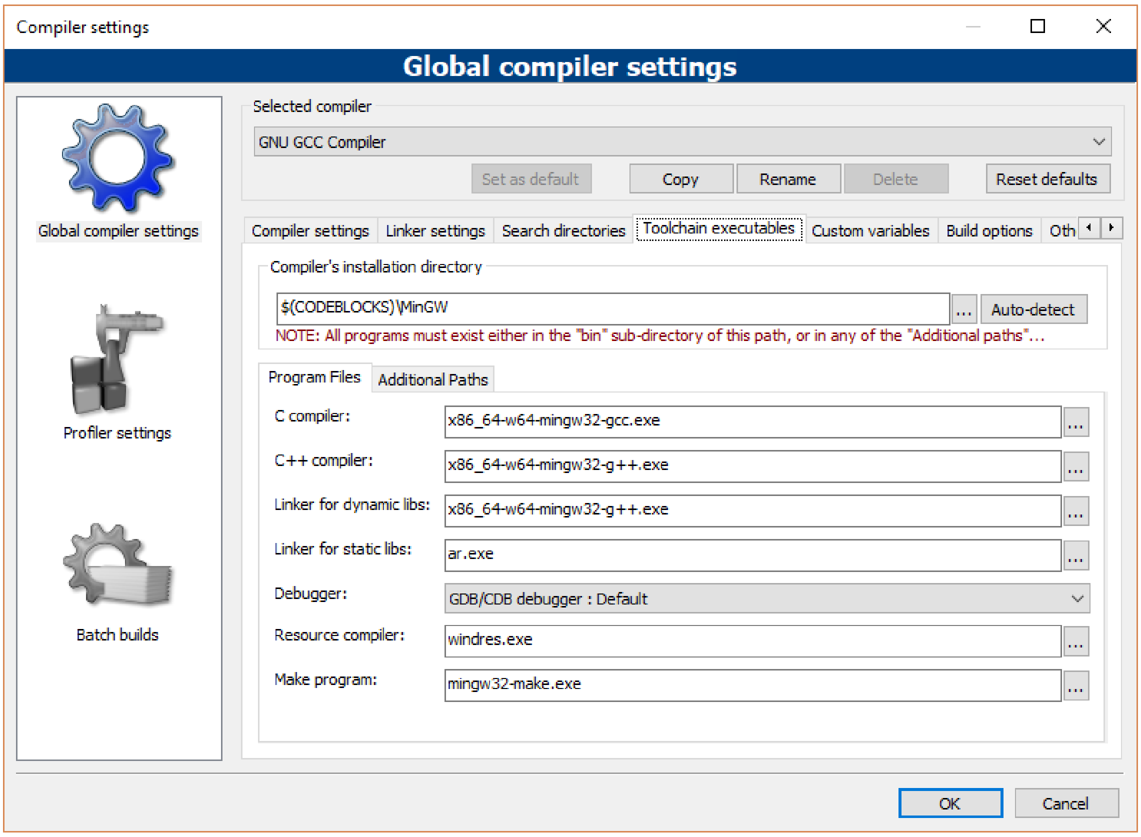
Este equipo › WINDOWS_10 (C:) › Code › CodeBlocks › MinGW

Nombre	Fecha de modificación	Tipo	Tamaño
bin	22/05/2022 11:51	Carpeta de archivos	
etc	22/05/2022 11:51	Carpeta de archivos	
include	22/05/2022 11:51	Carpeta de archivos	
lib	22/05/2022 11:51	Carpeta de archivos	
libexec	22/05/2022 11:51	Carpeta de archivos	
share	22/05/2022 11:51	Carpeta de archivos	
ssl	22/05/2022 11:51	Carpeta de archivos	
x86_64-w64-mingw32	22/05/2022 11:51	Carpeta de archivos	

4 – Run CODE::BLOCKS to let it detect the compiler we just installed. If it doesn't do this automatically, go to SETTINGS, COMPILER:



We will go to the TOOLCHAIN EXECUTABLES tab and enter the path where the compiler files are located. If we want it to be portable, we can use the system variable $(CODEBLOCKS) as the path. The AUTO-DETECT button should detect the necessary executables.



Selected compiler: GNU GCC Compiler

Compiler's installation directory: $(CODEBLOCKS)\MinGW

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional paths"...

C compiler: x86_64-w64-mingw32-gcc.exe

C++ compiler: x86_64-w64-mingw32-g++.exe

Linker for dynamic libs: x86_64-w64-mingw32-g++.exe

Linker for static libs: ar.exe

Debugger: GDB/CDB debugger : Default

Resource compiler: windres.exe

Make program: mingw32-make.exe

With this done, CODE::BLOCKS should compile without any issues.

Installation of Libraries – Linux and Raspbian

From the terminal, we will search for the SDL2 libraries with the following command:

```
sudo apt-cache search libsdl2
```

Next, we will install the libraries with the following command:

```
sudo apt-get update
sudo apt-get install libsdl2-dev
```

We will search for the SDL2-TTF libraries with the following command:

```
sudo apt-cache search libsdl2-ttf
```

Then, we will install the libraries with the following command:

```
sudo apt-get update
sudo apt-get install libsdl2-ttf-dev
```

We will search for the SFML libraries with the following command:

```
sudo apt-cache search sfml
```

Next, we will install the libraries with the following command:

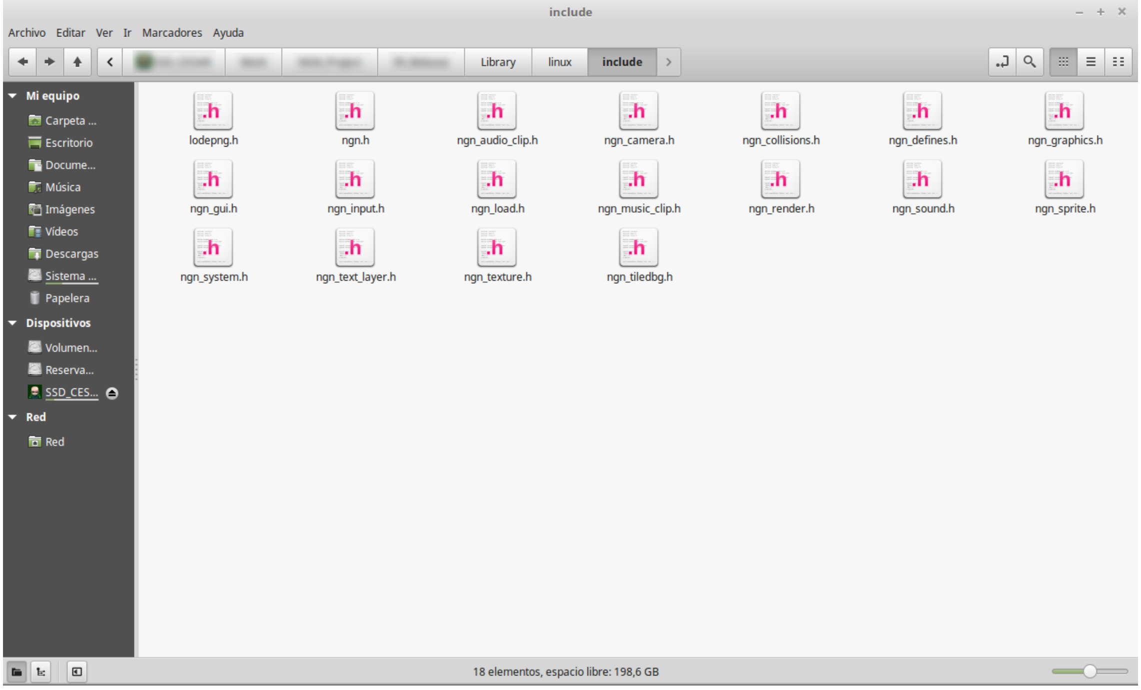
```
sudo apt-get update
sudo apt-get install libsfml-dev
```

It's possible that some C++ compiler dependencies are missing. In that case, we will install them with the following commands:

```
sudo apt-get update
sudo apt-get install build-essential
```

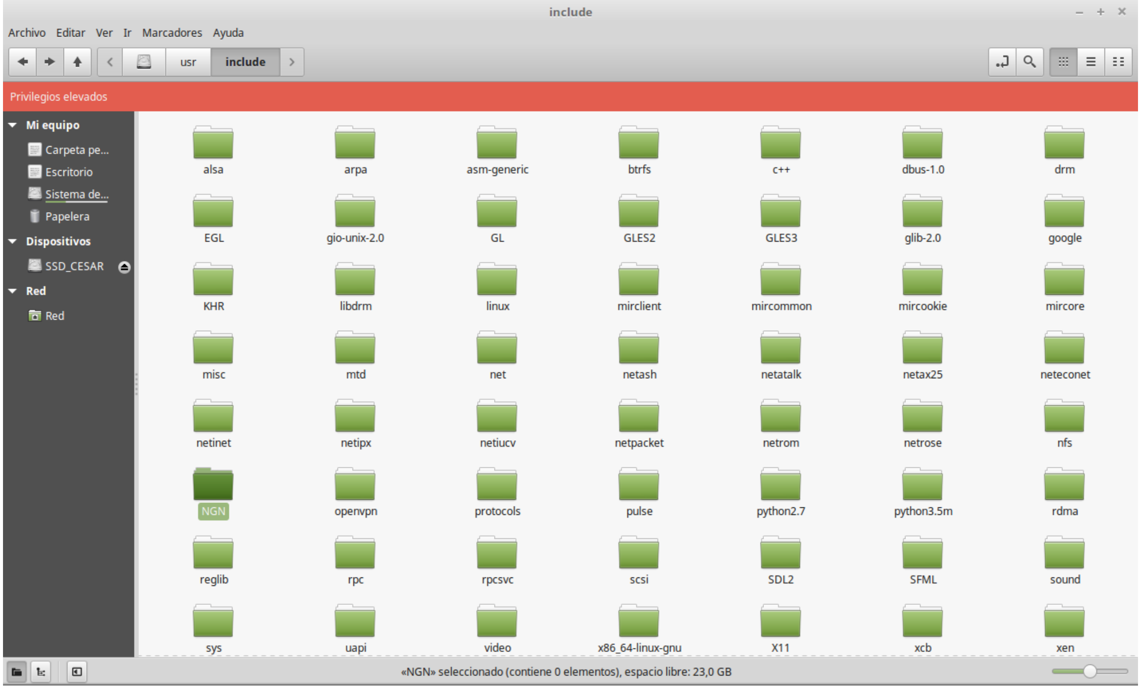
For the installation of the N'gine library, we will follow these steps:

Copy the INCLUDE (.h) files contained in the library:



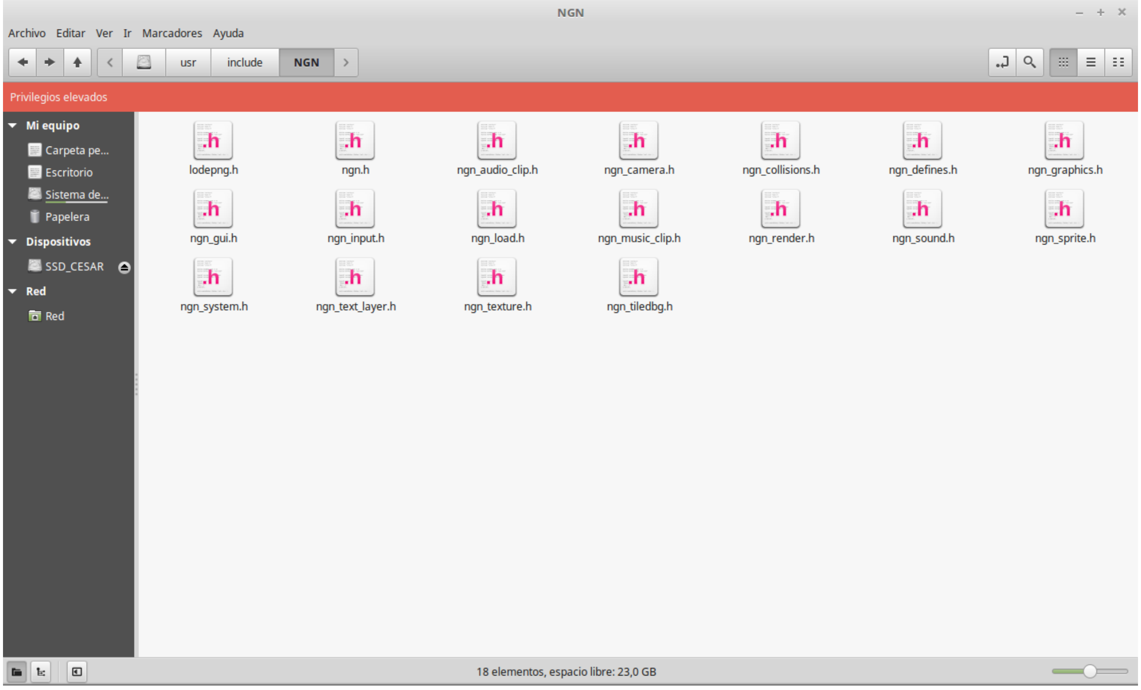
18 elementos, espacio libre: 198,6 GB

We will navigate to the system folder “usr/include” with elevated privileges and create a folder with the name 'NGN' (in uppercase):

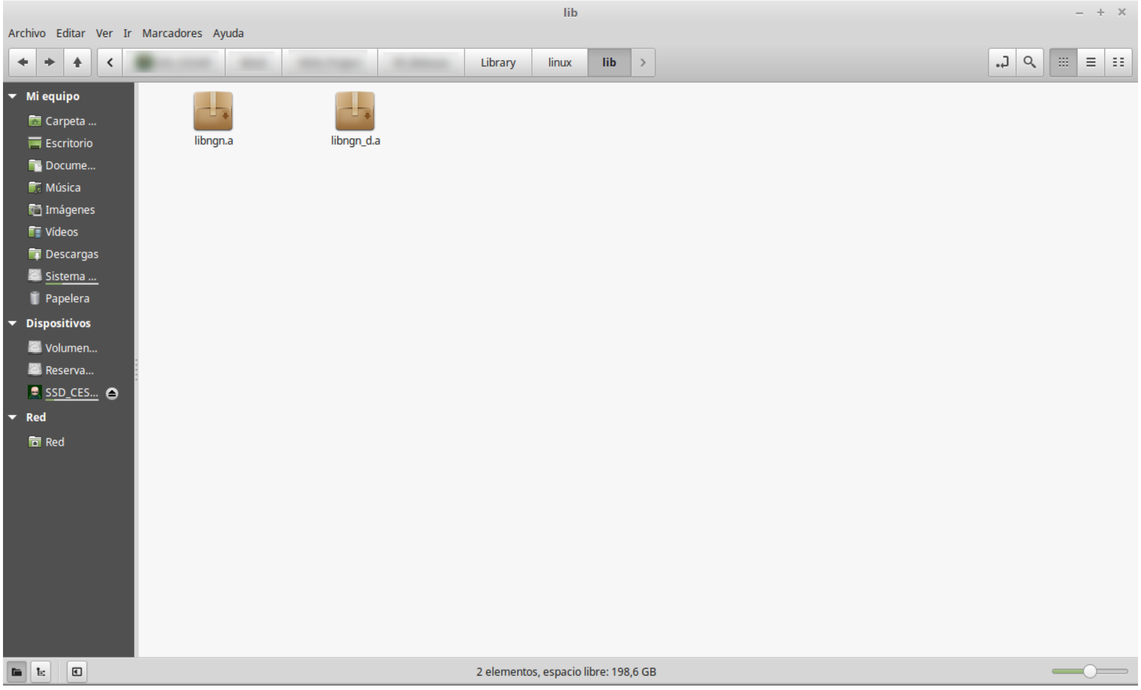


«NGN» seleccionado (contiene 0 elementos), espacio libre: 23,0 GB

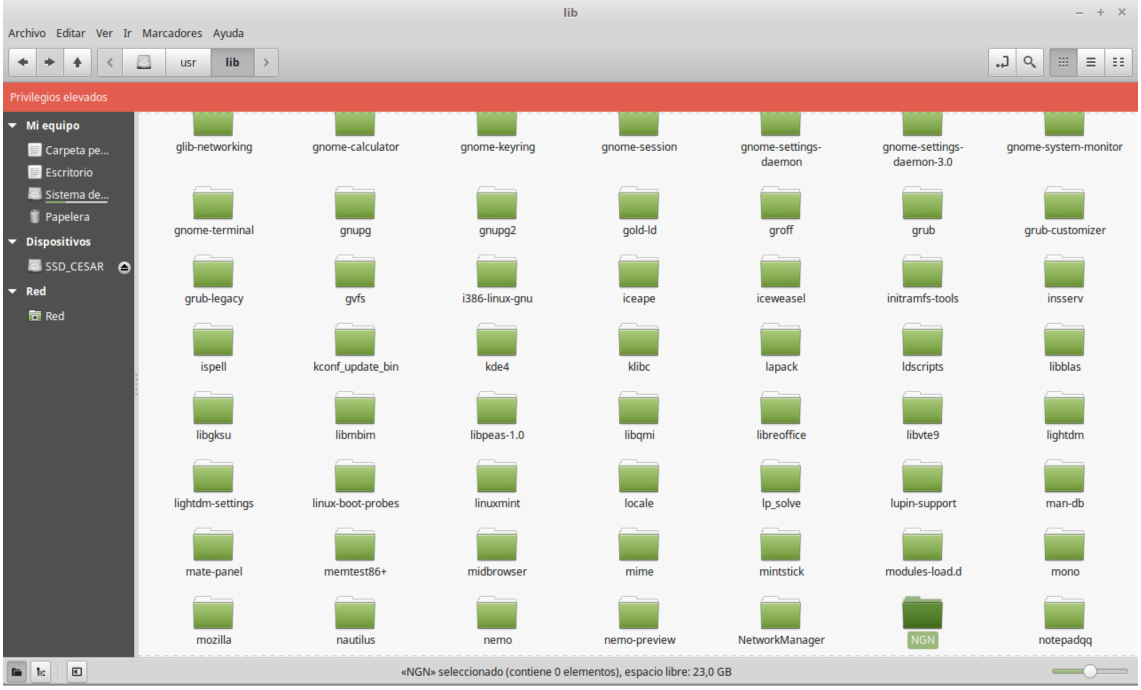
Next, we will paste the .h files inside it:



Now we will copy the binary files of the library (.a) from the lib folder:

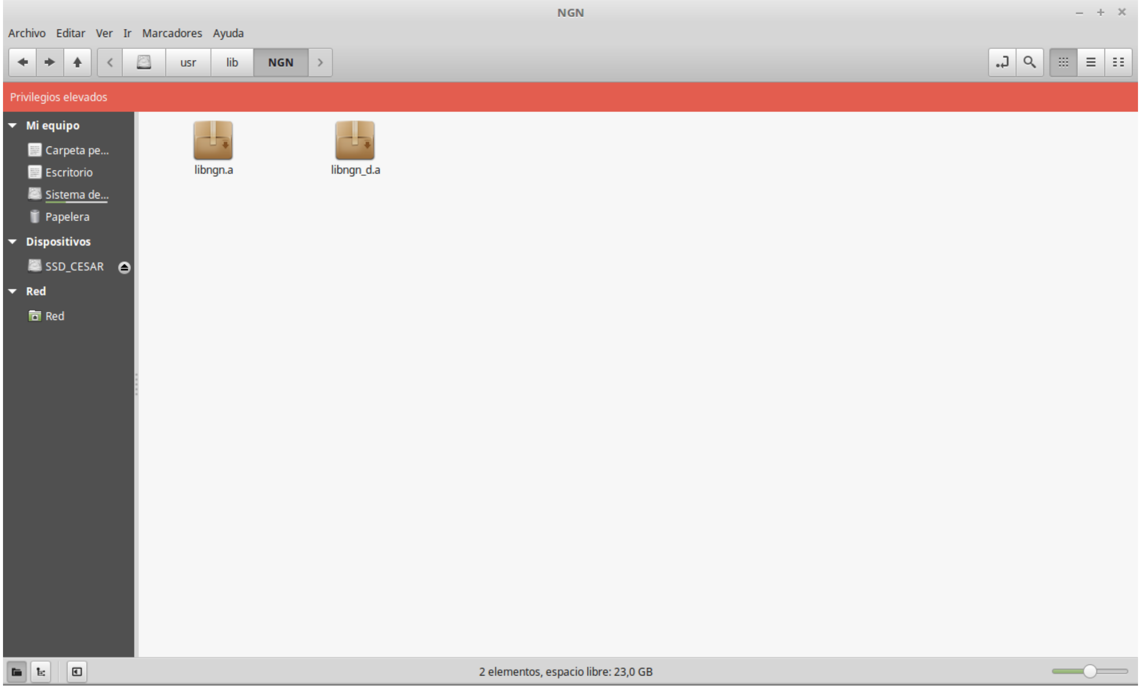


Now we will navigate to the system folder “usr/lib” with elevated privileges and create another folder named 'NGN' (in uppercase):



«NGN» seleccionado (contiene 0 elementos), espacio libre: 23,0 GB

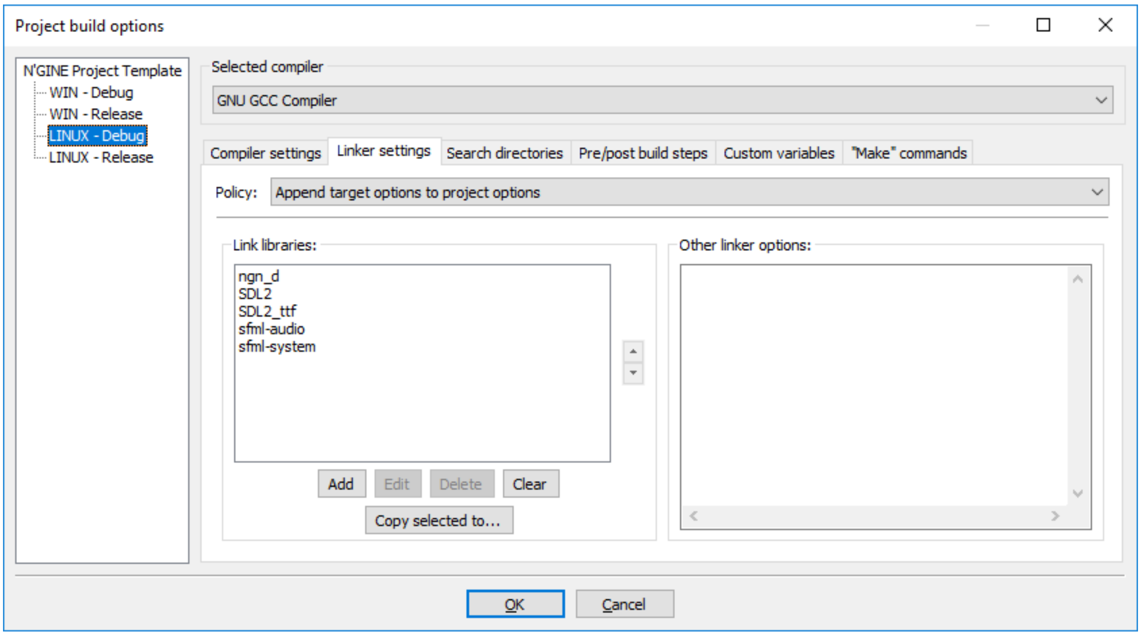
To finish, we will paste the .a files inside this folder:



2 elementos, espacio libre: 23,0 GB

Project Configuration – CODE::BLOCKS

In “Project build options”, in the “LINUX-DEBUG” section, we will configure the following parameters:



Project build options

- N'GINE Project Template
  - WIN - Debug
  - WIN - Release
  - LINUX - Debug
  - LINUX - Release

Selected compiler: GNU GCC Compiler

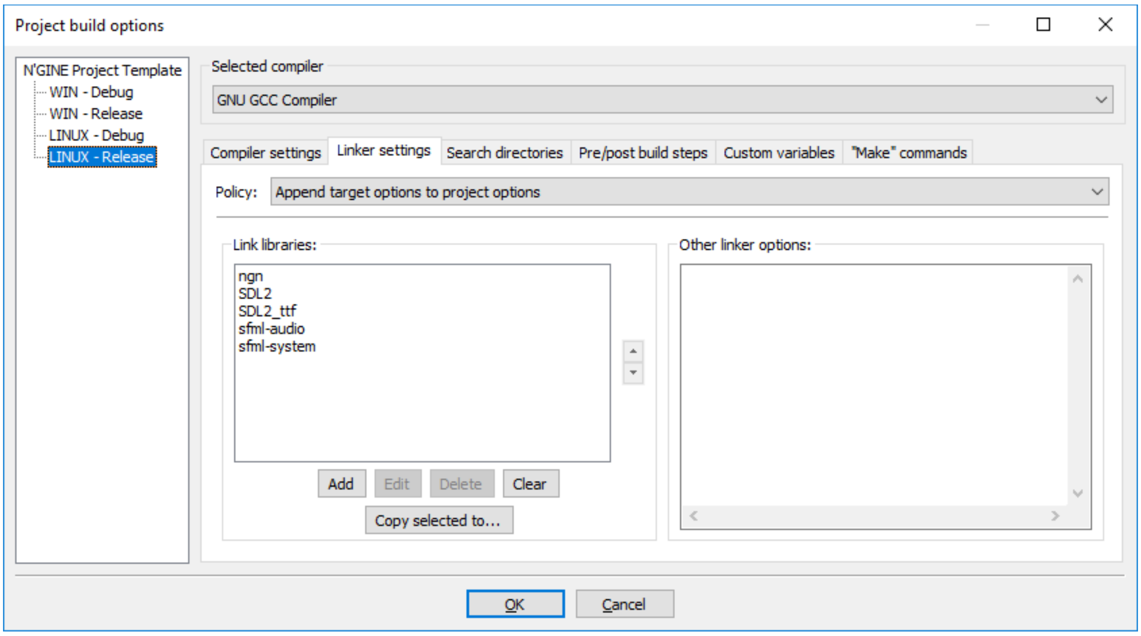
Policy: Append target options to project options

Link libraries:

```
ngn_d
SDL2
SDL2_ttf
sfml-audio
sfml-system
```

Other linker options:

In “Project build options”, in the “LINUX-DEBUG” section, we will configure the following parameters:



Project build options

- N'GINE Project Template
  - WIN - Debug
  - WIN - Release
  - LINUX - Debug
  - LINUX - Release

Selected compiler: GNU GCC Compiler

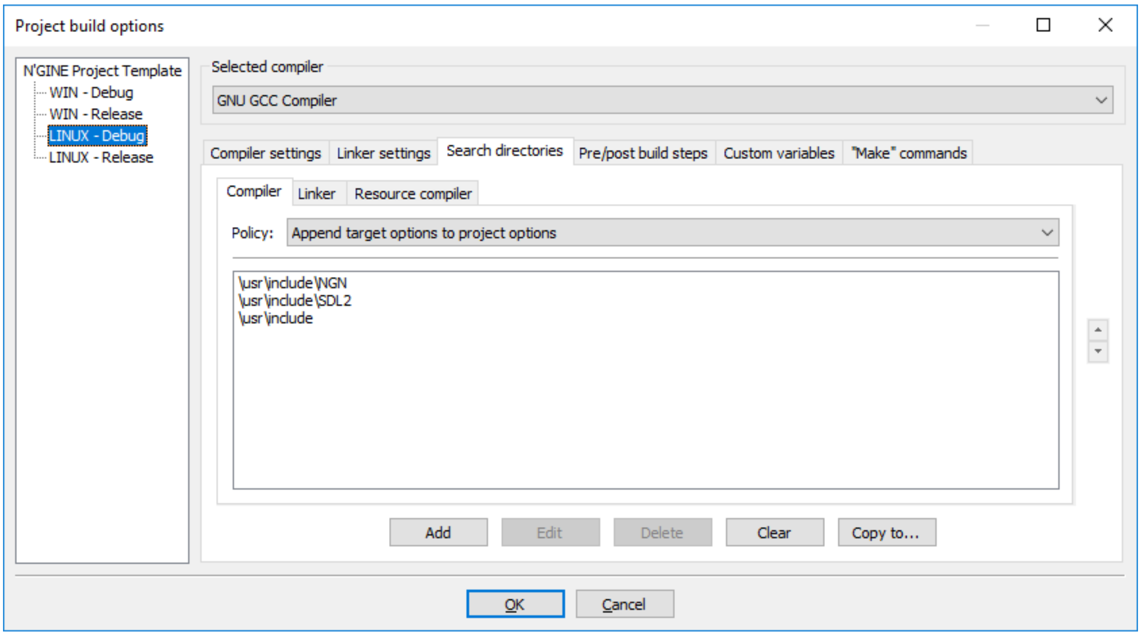
Policy: Append target options to project options

Link libraries:

```
ngn
SDL2
SDL2_ttf
sfml-audio
sfml-system
```

Other linker options:

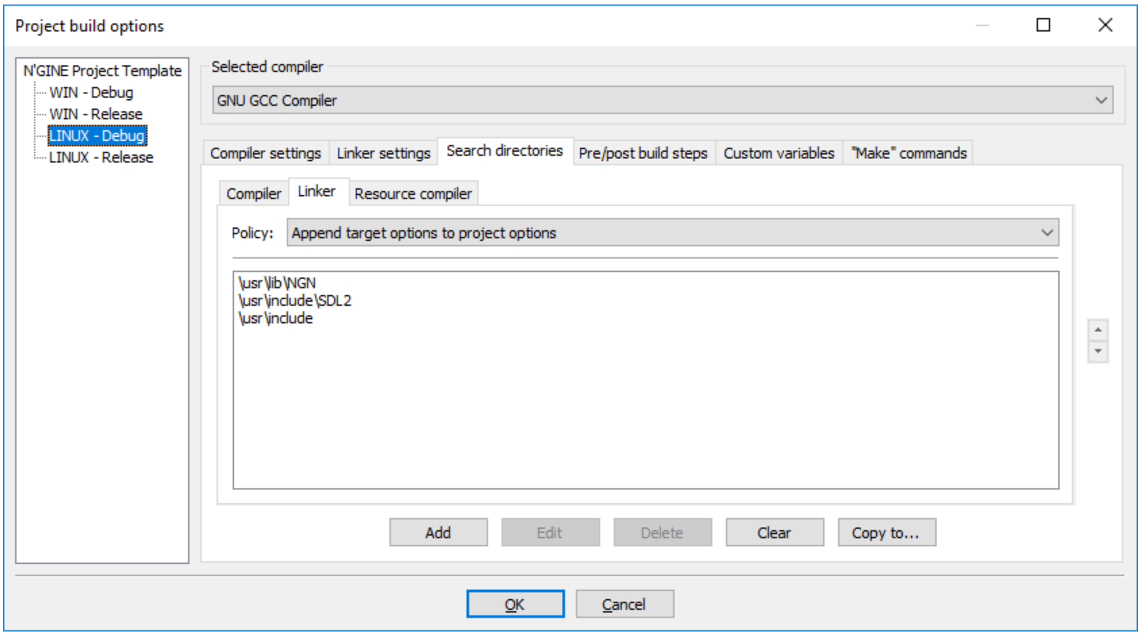
In “Project build options”, we will configure the following parameters in both the “LINUX-DEBUG” and “LINUX-RELEASE” sections:



Selected compiler: GNU GCC Compiler

Policy: Append target options to project options

```
\usr\include\NGN
\usr\include\SDL2
\usr\include
```

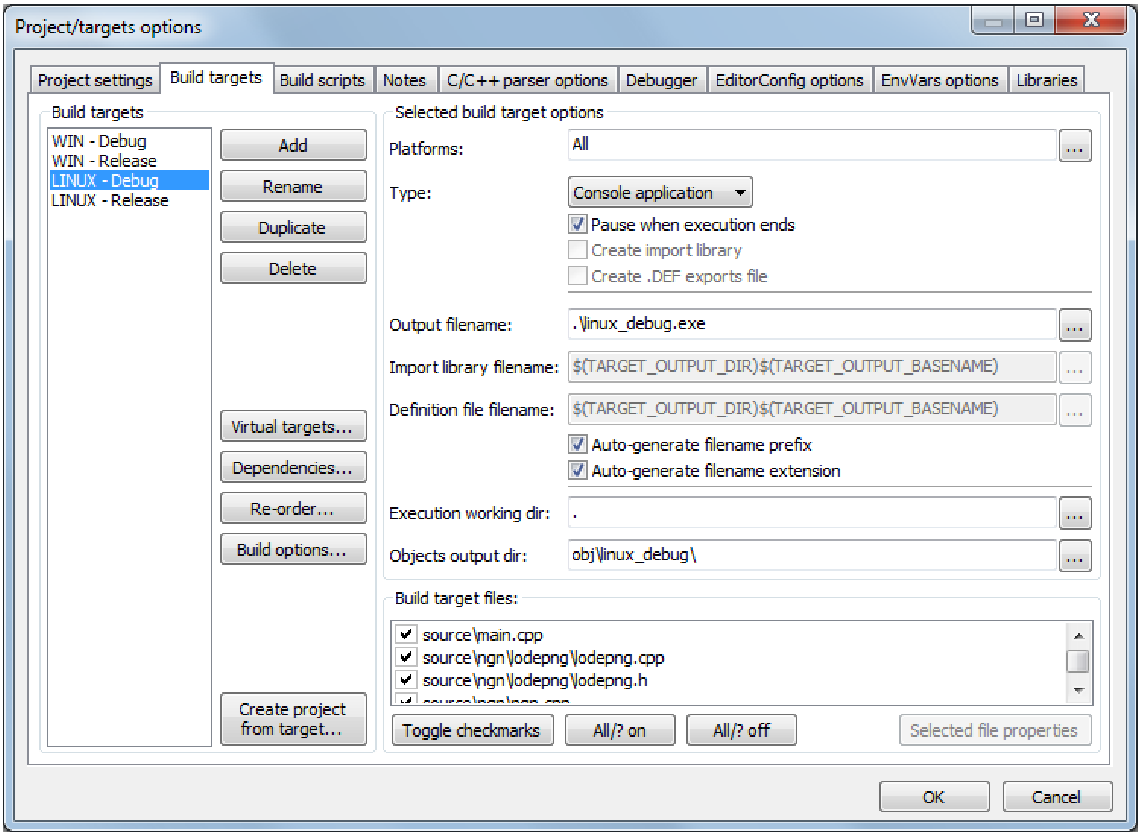


Selected compiler: GNU GCC Compiler

Policy: Append target options to project options

```
\usr\lib\NGN
\usr\include\SDL2
\usr\include
```

In 'Project/target options,' we will configure the following parameters:



Project/targets options

Build targets

WIN - Debug
WIN - Release
LINUX - Debug
LINUX - Release

Selected build target options

Platforms: All

Type: Console application

☑ Pause when execution ends
☐ Create import library
☐ Create .DEF exports file

Output filename: .\linux_debug.exe

Import library filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

Definition file filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

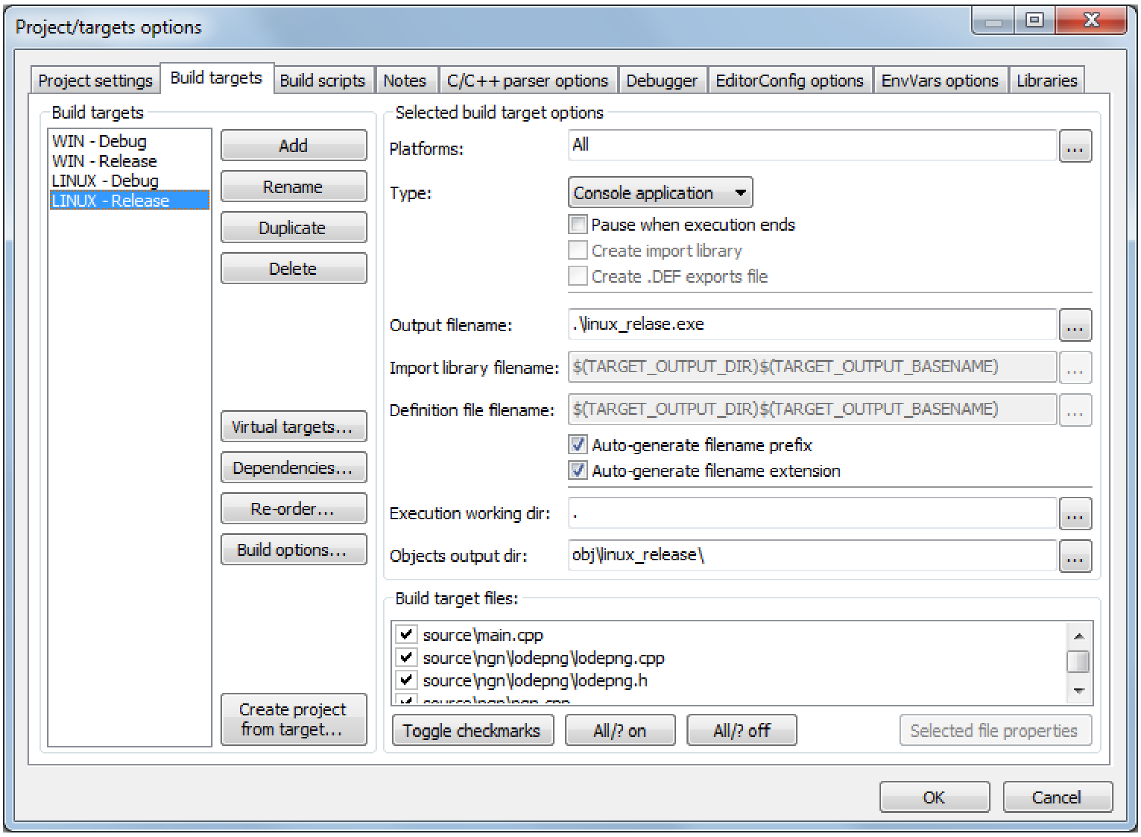
☑ Auto-generate filename prefix
☑ Auto-generate filename extension

Execution working dir: .

Objects output dir: obj\linux_debug\

Build target files:

☑ source\main.cpp
☑ source\ngn\lodepng\lodepng.cpp
☑ source\ngn\lodepng\lodepng.h



Project/targets options

Build targets

WIN - Debug
WIN - Release
LINUX - Debug
LINUX - Release

Selected build target options

Platforms: All

Type: Console application

☐ Pause when execution ends
☐ Create import library
☐ Create .DEF exports file

Output filename: .\linux_relase.exe

Import library filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

Definition file filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

☑ Auto-generate filename prefix
☑ Auto-generate filename extension

Execution working dir: .

Objects output dir: obj\linux_release\

Build target files:

☑ source\main.cpp
☑ source\ngn\lodepng\lodepng.cpp
☑ source\ngn\lodepng\lodepng.h

Optionally, the library includes various templates with all these options pre-configured, both for Windows and Linux/Raspbian. If you use these templates when creating projects, you will only need to perform the library installation step.

For Linux Mint and Raspbian environments, there are also automated installation scripts with the latest versions of the N'gine library, as well as the used versions of SDL2, SDL2_ttf, and SFML3. These versions will be installed in the corresponding system folders and may replace existing versions of these libraries.

- Revision date: Friday, December 8, 2023.