

## N'GINE - Configuración de CODE::BLOCKS

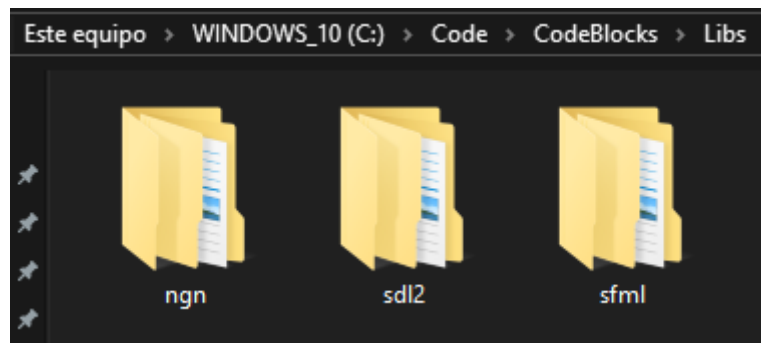
### Instalación de las librerías - Windows

Empezaremos descargando las librerías adicionales desde sus páginas oficiales:

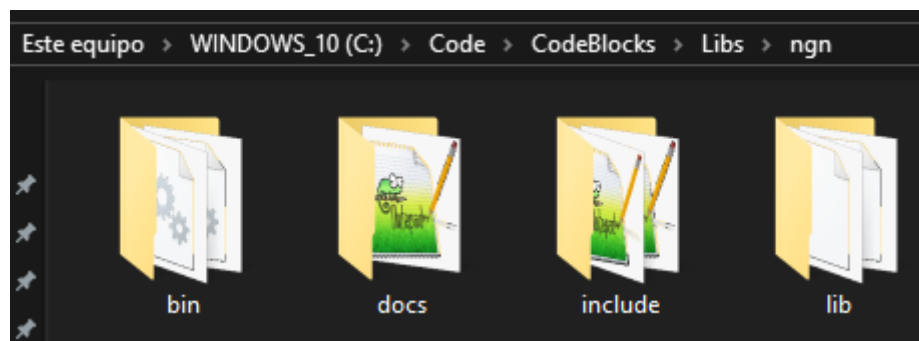
SDL2: <https://www.libsdl.org/download-2.0.php>

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

A continuación, crearemos una carpeta llamada “Libs” dentro de la carpeta donde tengamos instalado CODE::BLOCKS. En el interior de esta carpeta, crearemos las carpetas “ngn”, “sdl2” y “sfml”.

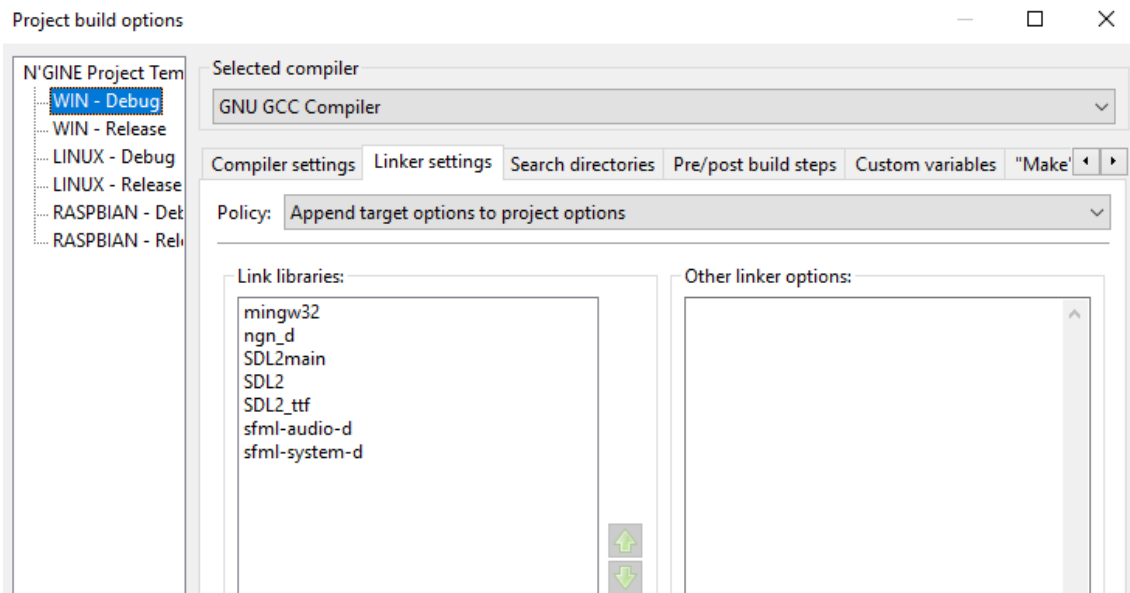


Copiaremos en cada carpeta los directorios “include”, “lib” y opcionalmente, “bin” y “docs” correspondientes a cada librería.

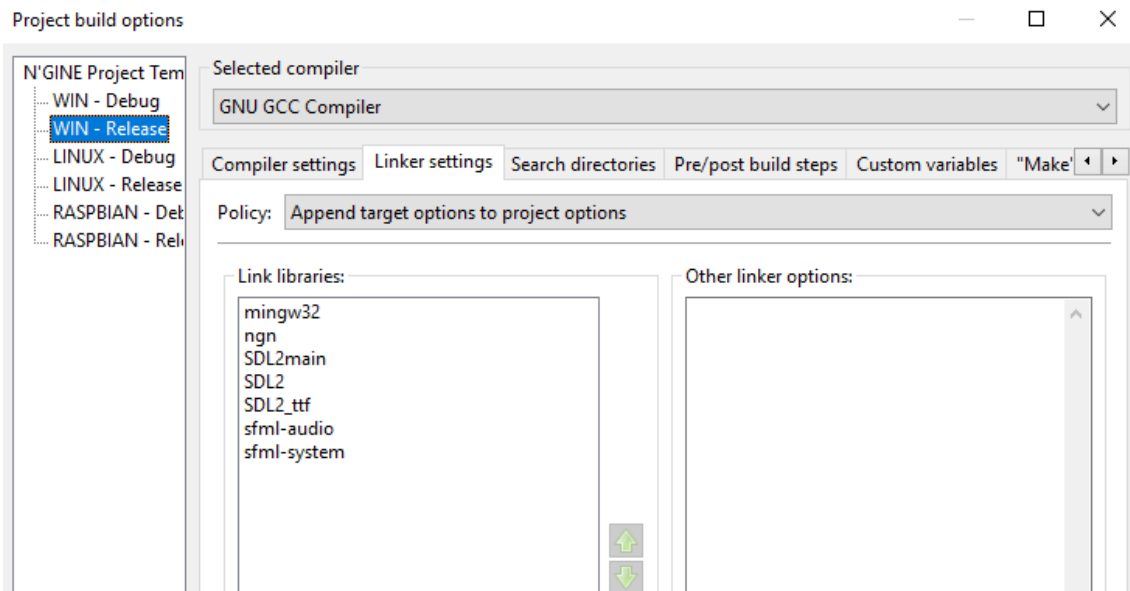


## Configuración del proyecto - CODE::BLOCKS

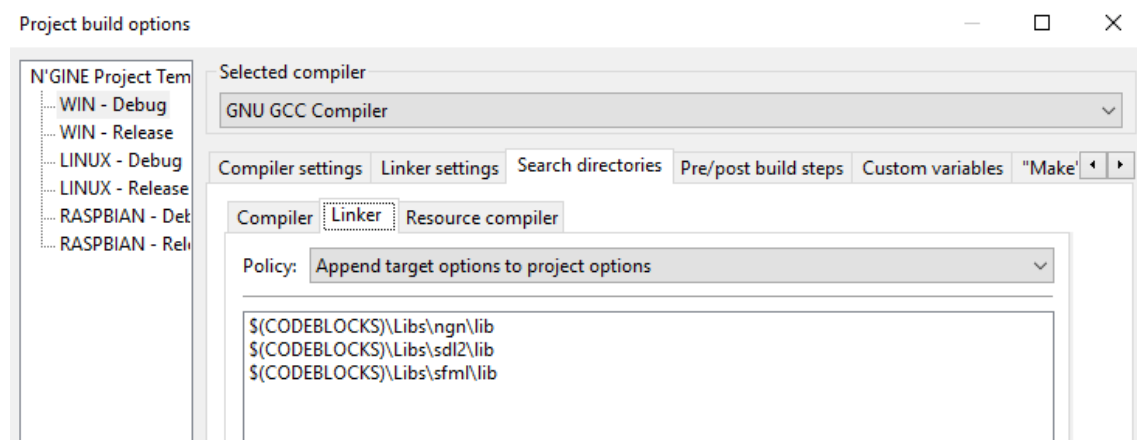
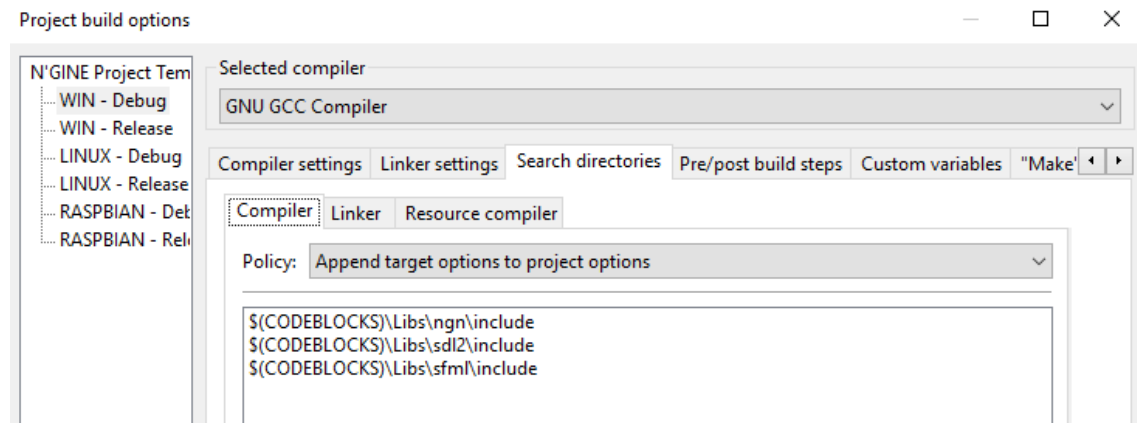
En “Project build options”, en la sección “WIN-Debug”, configuraremos los siguientes parámetros:



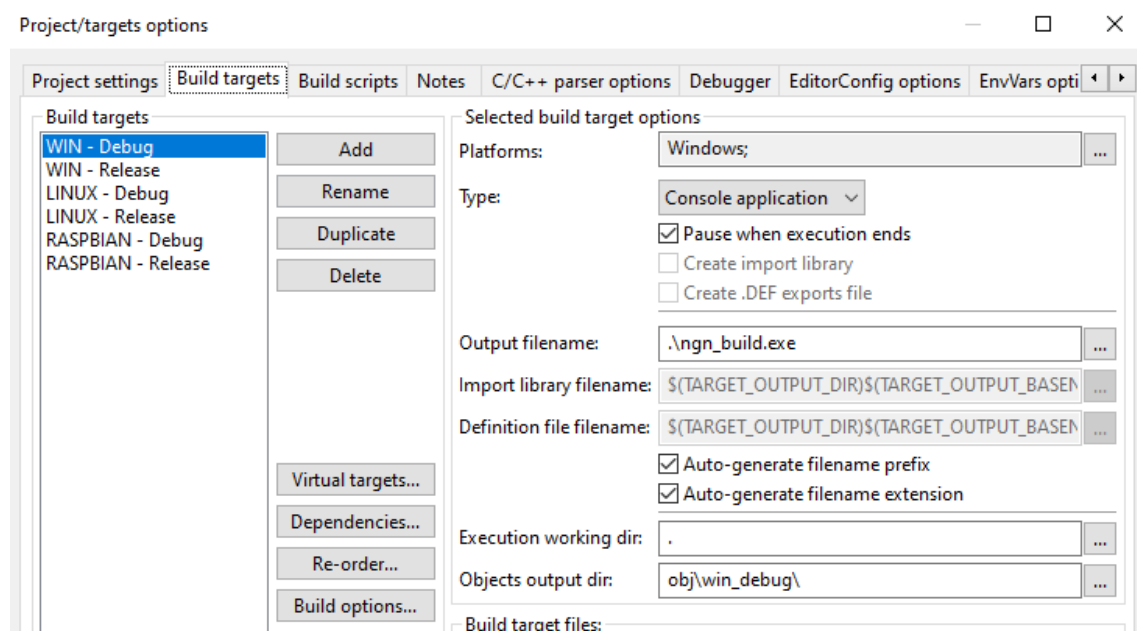
En “Project build options”, en la sección “WIN-Release”, configuraremos los siguientes parámetros:

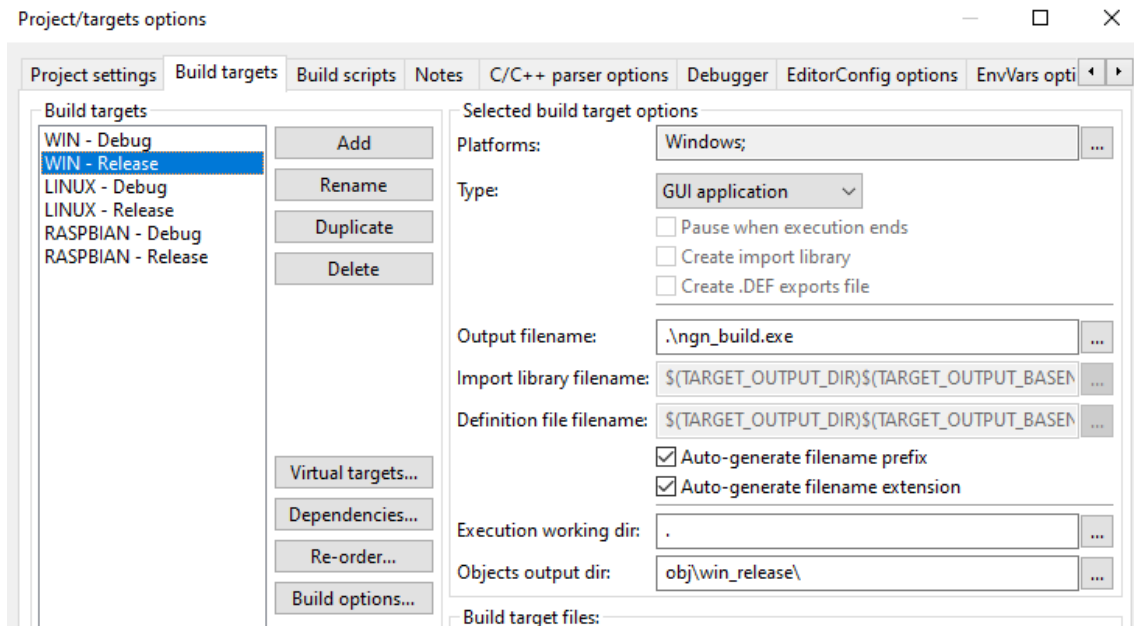


En “Project build options” configuraremos tanto en la sección “WIN-DEBUG” como en “WIN-RELEASE” los siguientes parámetros:



En “Project/target options”, configuraremos los siguientes parámetros:





## **Actualización del compilador MinGW sobre CODE::BLOCKS 20 - (Windows)**

La versión 20 de CODE::BLOCKS incluye no incluye versión más reciente del compilador MinGW (MinGW-W64 project, version 8.1.0, 32/64 bit, SEH). Si hemos instalado la versión de CODE::BLOCKS que incorpora este compilador o si no disponemos de este instalado (instalación de CODE::BLOCKS limpia), deberemos instalar o cambiar la versión del compilador. Para dicha instalación o actualización deberemos de seguir los siguientes pasos:

1 - Descargaremos la utilidad MSYS2 desde su web oficial:

<https://www.msys2.org/#installation>

La propia página web contiene un tutorial para realizar la instalación de la utilidad.

2 - Una vez instalada, procederemos a la instalación de la última versión del compilador MinGW. Desde la consola que se abre con el icono "MSYS2 MINGW64" ejecutaremos los siguientes comandos:

```
pacman -Syu
```

Y seguiremos las instrucciones de la pantalla para actualizar el repositorio de paquetes.

Una vez completada la instalación, veremos algo parecido a esto:

```
msys2-launcher-1.5-3-x86_64      96.4 KiB   227 KiB/s  00:00 [#####] 100%
libxcrypt-4.4.37-1-x86_64       76.9 KiB   184 KiB/s  00:00 [#####] 100%
Total (13/13)                    2.9 MiB   1845 KiB/s  00:02 [#####] 100%
(13/13) checking keys in keyring [#####] 100%
(13/13) checking package integrity [#####] 100%
(13/13) loading package files [#####] 100%
(13/13) checking for file conflicts [#####] 100%
(13/13) checking available disk space [#####] 100%
:: Processing package changes...
( 1/13) upgrading bash-completion [#####] 100%
( 2/13) upgrading libiconv [#####] 100%
( 3/13) upgrading libxcrypt [#####] 100%
( 4/13) upgrading libssh2 [#####] 100%
( 5/13) upgrading libedit [#####] 100%
( 6/13) upgrading libcurl [#####] 100%
( 7/13) upgrading curl [#####] 100%
( 8/13) upgrading libargp [#####] 100%
( 9/13) upgrading getent [#####] 100%
(10/13) upgrading libhogweed [#####] 100%
(11/13) upgrading libnettle [#####] 100%
(12/13) upgrading msys2-launcher [#####] 100%
(13/13) upgrading nettle [#####] 100%
:: Running post-transaction hooks...
(1/1) Updating the info directory file...

NightFox@FOXCAVE MINGW64 ~
$
```

3 - Ahora instalaremos la última versión del compilador MinGW, tecleando el siguiente comando en la consola:

```
pacman -S mingw-w64-x86_64-toolchain
```

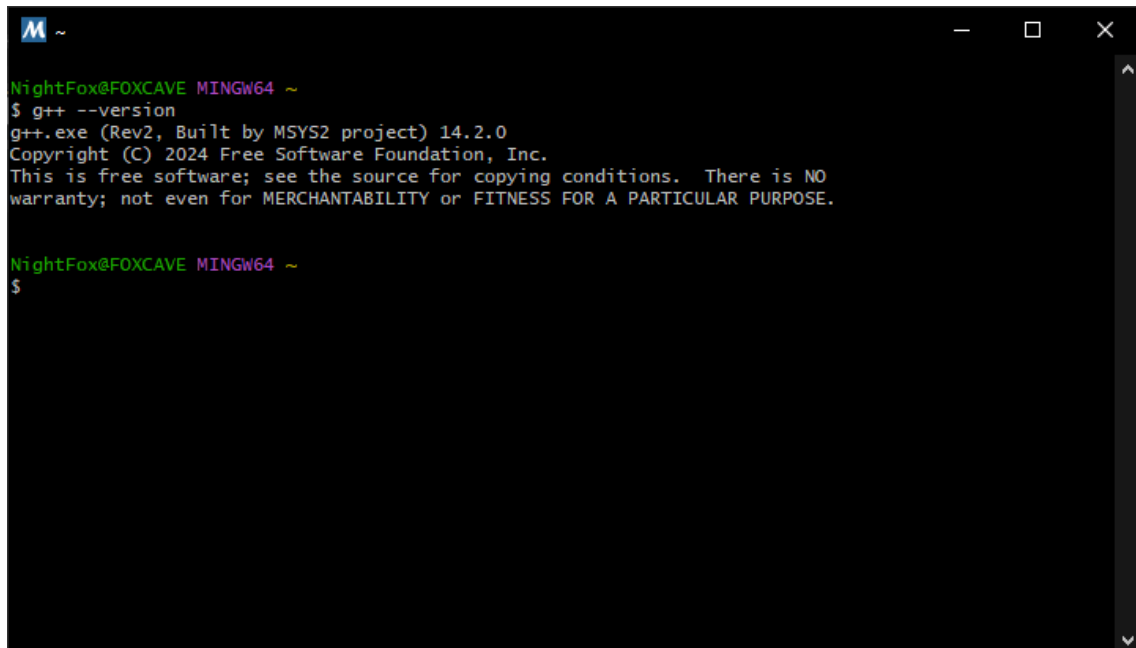
Al finalizar, deberíamos ver algo parecido a esto:

```
Optional dependencies for mingw-w64-x86_64-openssl
mingw-w64-x86_64-ca-certificates
(25/40) installing mingw-w64-x86_64-termcap [#####] 100%
(26/40) installing mingw-w64-x86_64-readline [#####] 100%
(27/40) installing mingw-w64-x86_64-sqlite3 [#####] 100%
Optional dependencies for mingw-w64-x86_64-sqlite3
mingw-w64-x86_64-tcl: for sqlite3_analyzer [pending]
(28/40) installing mingw-w64-x86_64-tcl [#####] 100%
(29/40) installing mingw-w64-x86_64-tk [#####] 100%
(30/40) installing mingw-w64-x86_64-xz [#####] 100%
(31/40) installing mingw-w64-x86_64-tzdata [#####] 100%
(32/40) installing mingw-w64-x86_64-python [#####] 100%
(33/40) installing mingw-w64-x86_64-xxhash [#####] 100%
(34/40) installing mingw-w64-x86_64-gdb [#####] 100%
Optional dependencies for mingw-w64-x86_64-gdb
mingw-w64-x86_64-python-pygments: for syntax highlighting
(35/40) installing mingw-w64-x86_64-gdb-multiarch [#####] 100%
Optional dependencies for mingw-w64-x86_64-gdb-multiarch
mingw-w64-x86_64-python-pygments: for syntax highlighting
(36/40) installing mingw-w64-x86_64-libmangle-git [#####] 100%
(37/40) installing mingw-w64-x86_64-make [#####] 100%
(38/40) installing mingw-w64-x86_64-pkgconf [#####] 100%
(39/40) installing mingw-w64-x86_64-tools-git [#####] 100%
(40/40) installing mingw-w64-x86_64-winstorecompat-git [#####] 100%

NightFox@FOXCAVE MINGW64 ~
$
```

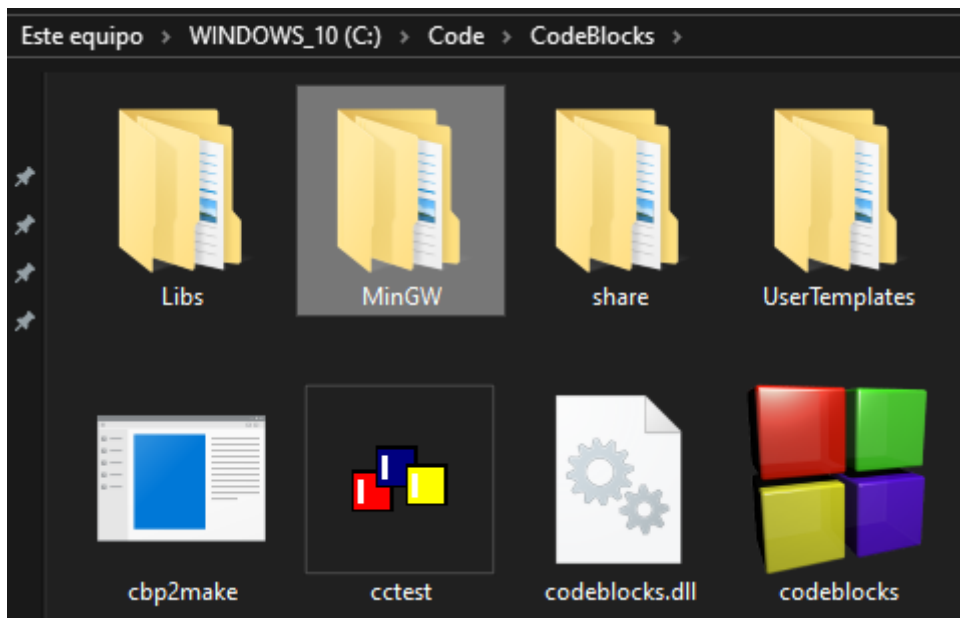
4 - Verificaremos la versión instalada del compilador, mediante el comando:

```
g++ --version
```

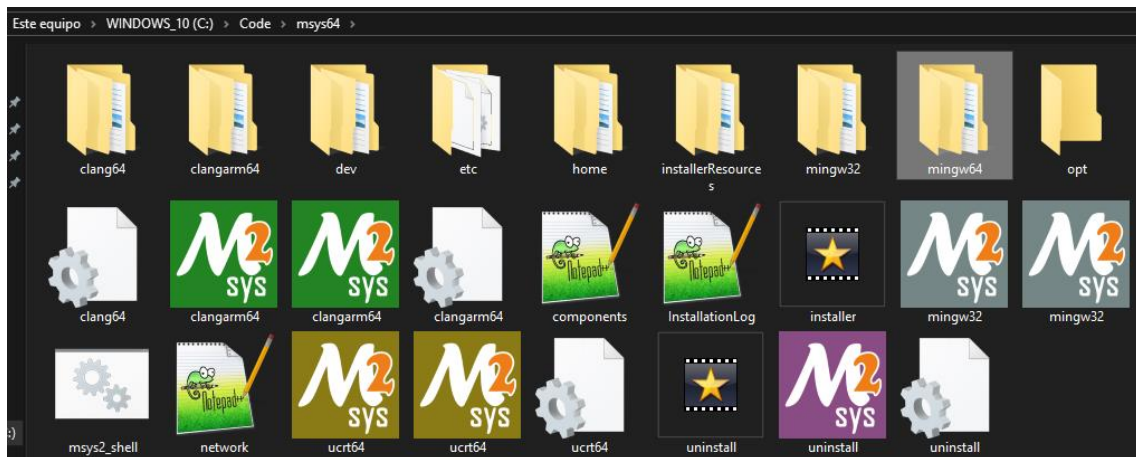


```
NightFox@FOXCAVE MINGW64 ~  
$ g++ --version  
g++.exe (Rev2, Built by MSYS2 project) 14.2.0  
Copyright (C) 2024 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
NightFox@FOXCAVE MINGW64 ~  
$
```

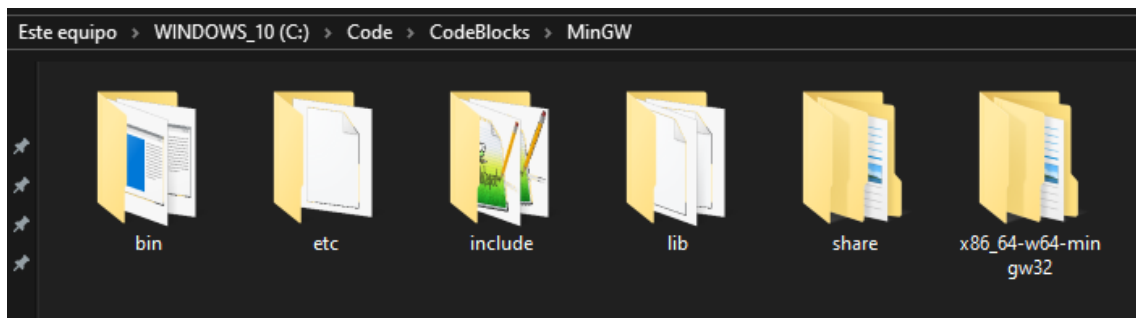
5 - En la carpeta donde tengamos instalado el programa CODE::BLOCKS, crearemos una carpeta llamada MinGW. Si esta ya existe, borraremos todo su contenido.



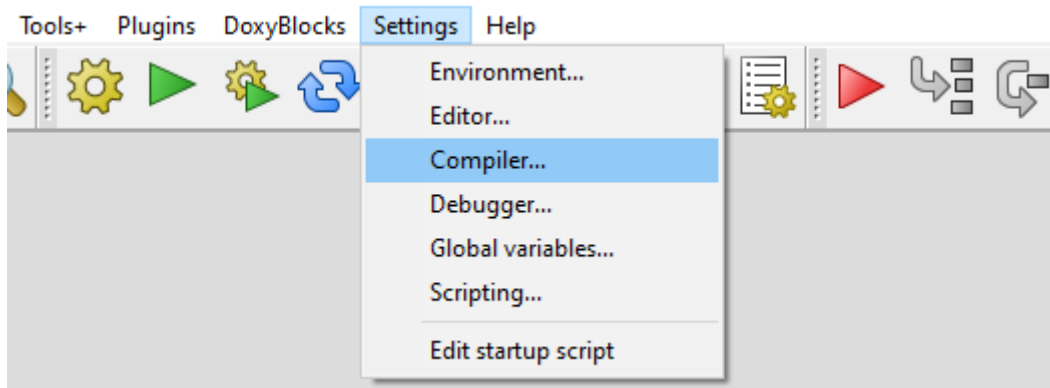
6 - Localizaremos la carpeta de instalación de la utilidad MSYS2 y buscaremos la carpeta “mingw64”.



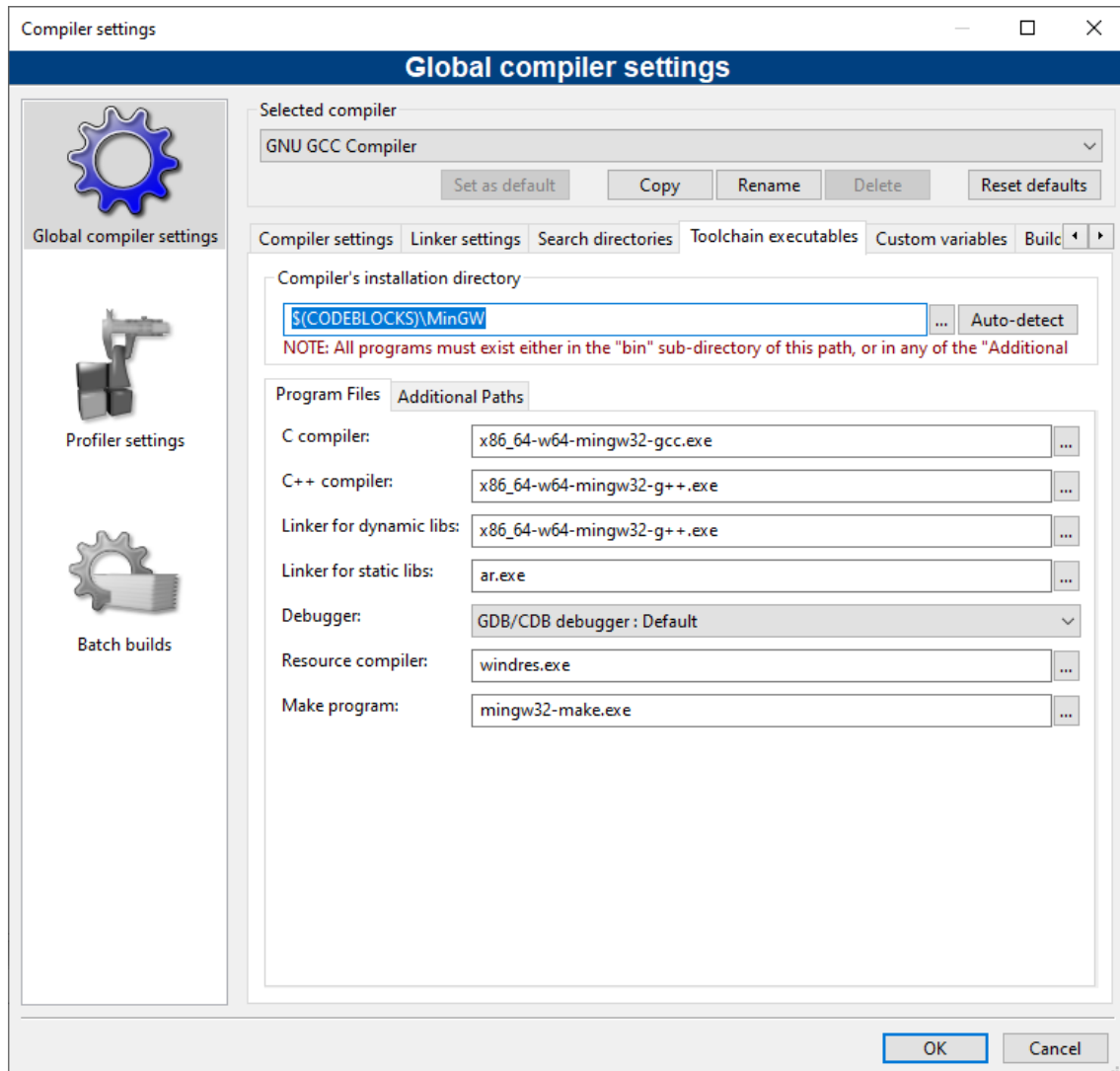
7 - Copiaremos todo el contenido de esta carpeta y lo pegaremos en el interior de la carpeta MinGW que hemos creado anteriormente en el directorio de instalación de CODE::BLOCKS.



8 - Ejecutaremos CODE::BLOCKS para que detecte el compilador que acabamos de instalar. Si no lo hace de manera automática, iremos **SETTINGS, COMPILER**:



Nos colocaremos en la pestaña **TOOLCHAIN EXECUTABLES** e introduciremos la ruta donde estén los archivos del compilador. Si queremos que sea portable, podemos usar la variable de sistema **\$(CODEBLOCKS)** como ruta. El botón **AUTO-DETECT** debería detectar los ejecutables necesarios.



Hecho esto, CODE::BLOCKS debería de compilar sin problemas.



## **Instalación de las librerías - Linux y Raspberry PI OS**

Desde el terminal, buscaremos las librerías SDL2 con el siguiente comando:

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

A continuación, instalaremos las librerías con este otro:

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

Buscaremos las librerías SDL2-TTF con el siguiente comando:

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

A continuación, instalaremos las librerías con este otro:

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

Buscaremos las librerías SFML con el siguiente comando:

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

A continuación, instalaremos las librerías con este otro:

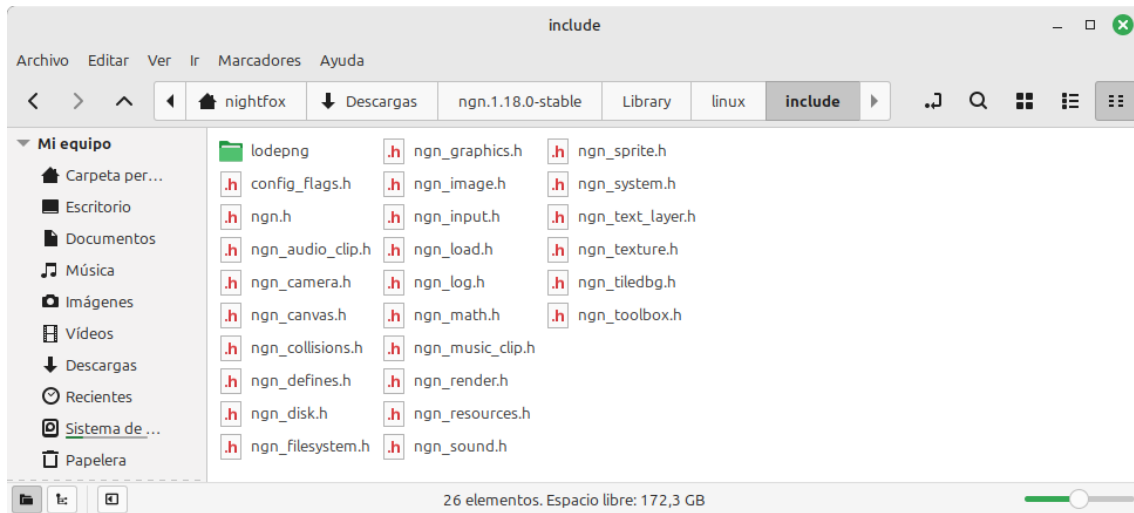
```
sudo apt update  
sudo apt install libsFML-dev
```

Es posible que falten algunas dependencias del compilador de C++. En ese caso las instalaremos con los siguientes comandos:

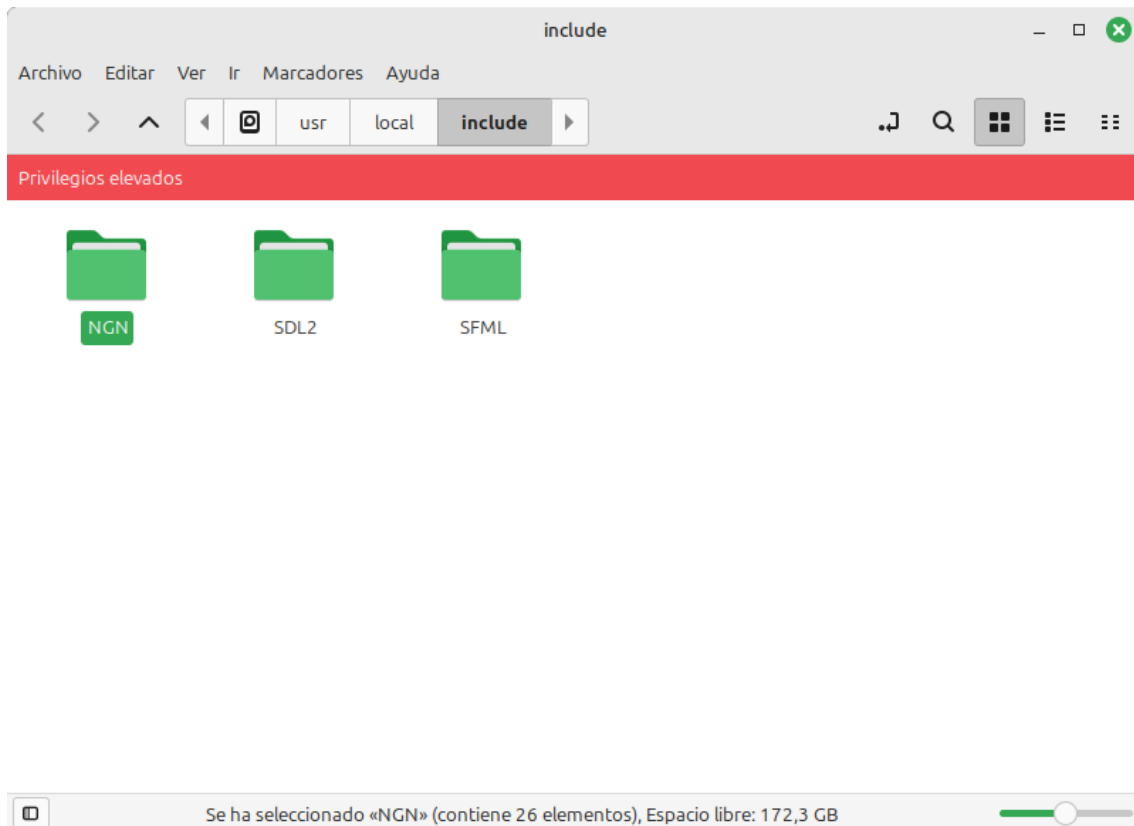
```
sudo apt update  
sudo apt install build-essential
```

Para la instalación de la librería N'gine, seguiremos los siguientes pasos:

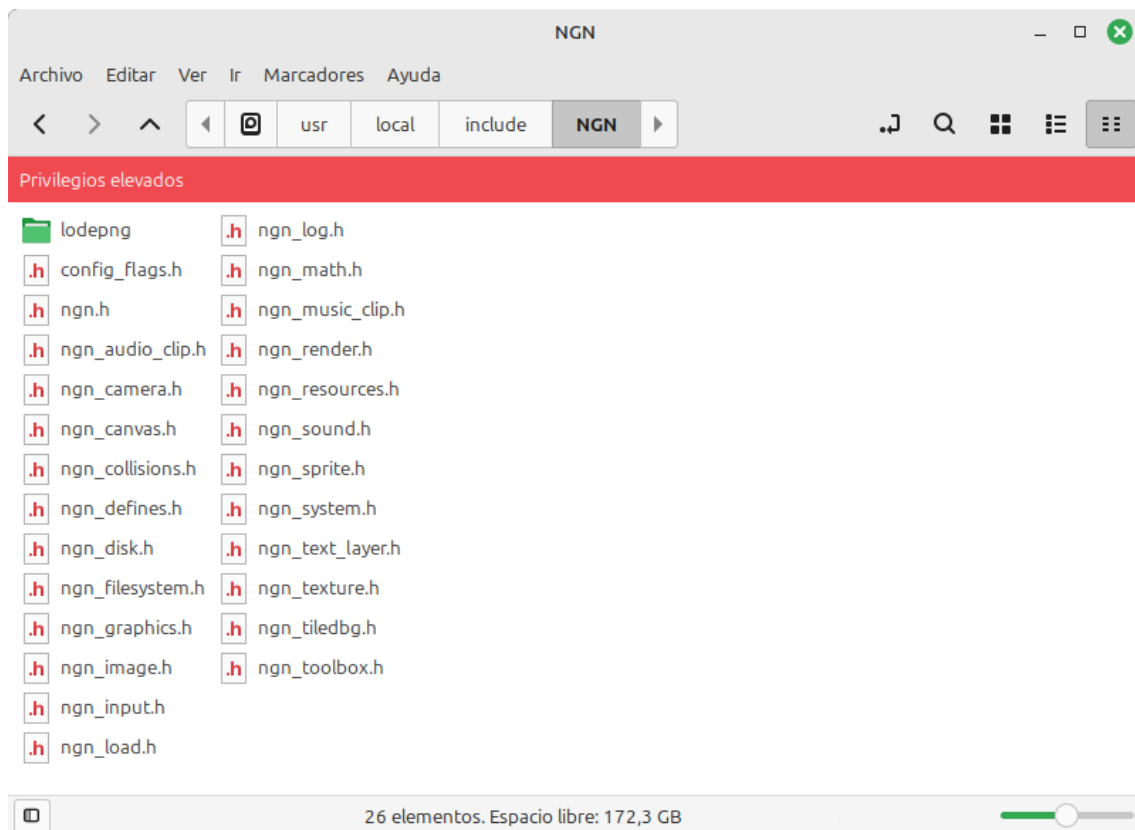
Copiaremos los archivos INCLUDE (.h) contenidos en la librería:



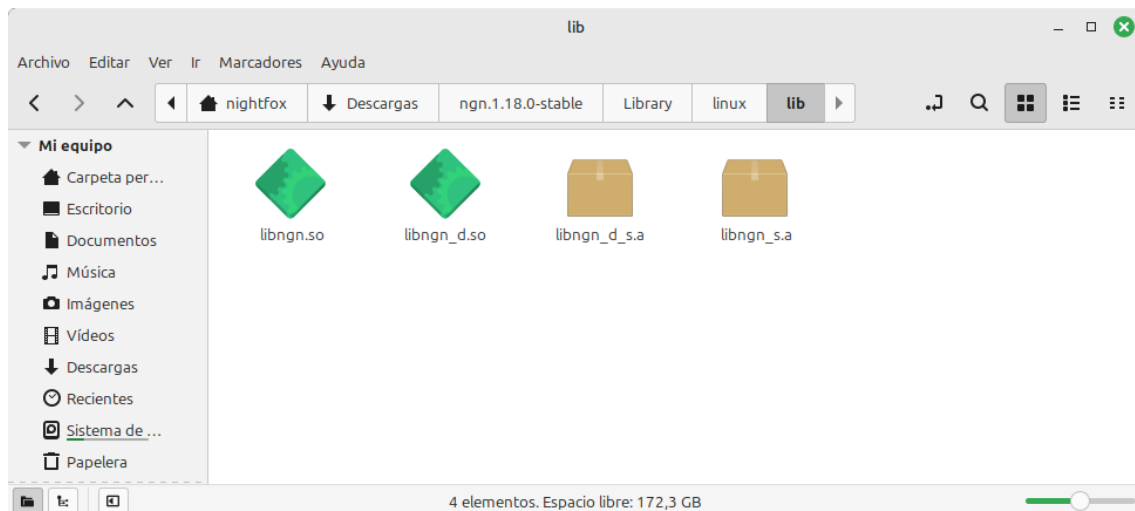
Con privilegios elevados, nos colocaremos en la carpeta del sistema “/usr/local/include” y crearemos una carpeta con el nombre “NGN” (en mayúsculas):



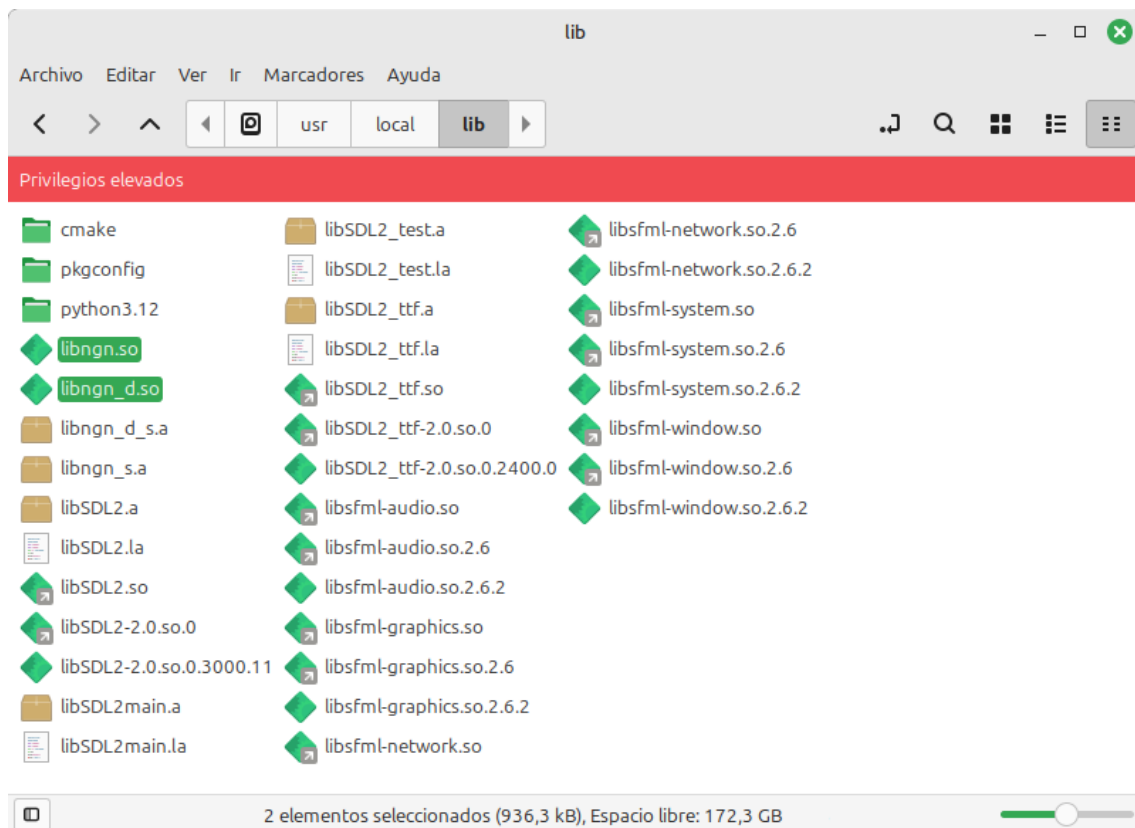
A continuación, pegaremos los archivos .h en su interior:



Ahora copiaremos los archivos binarios de la librería (.a y .so) de la carpeta lib:



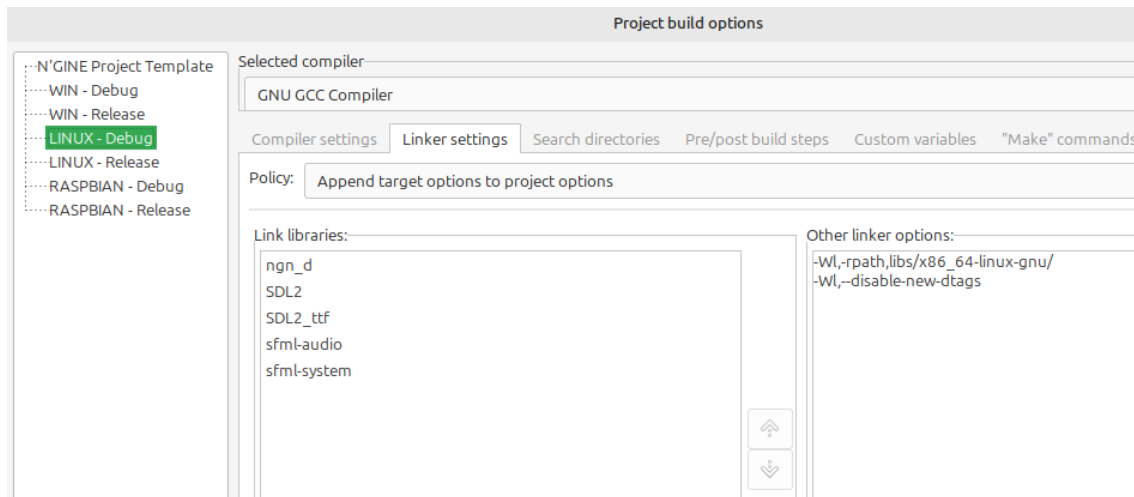
Nos colocaremos en la carpeta del sistema “/usr/local/lib” con privilegios elevados y pegaremos los archivos .a y .so en el interior de esta carpeta:



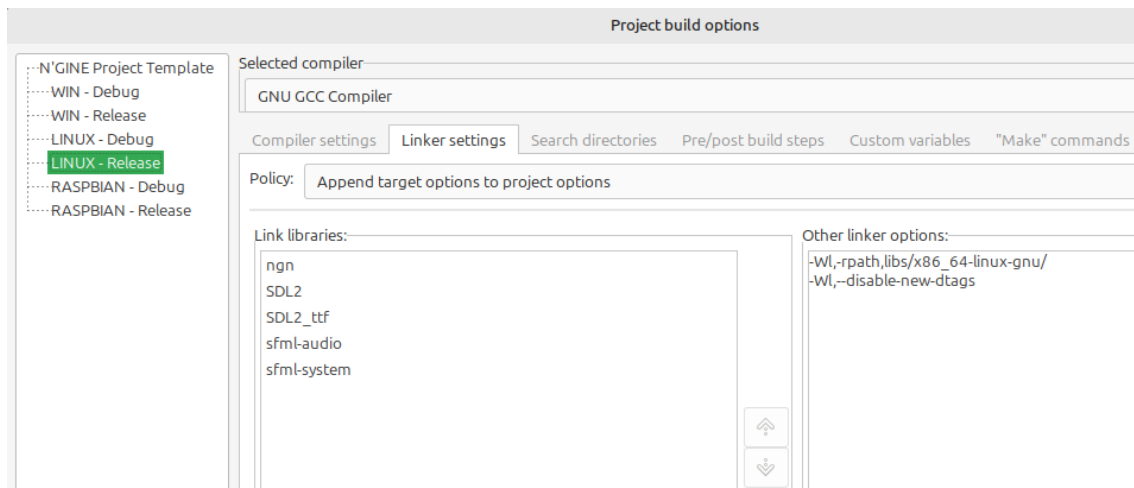
## Configuración del proyecto - CODE::BLOCKS

*En este caso, además, asumiremos que has compilado manualmente las librerías SDL2, SDL2\_ttf y SFML para mantenerlas actualizadas con las últimas versiones, por lo que estarán ubicadas en el directorio /usr/local.*

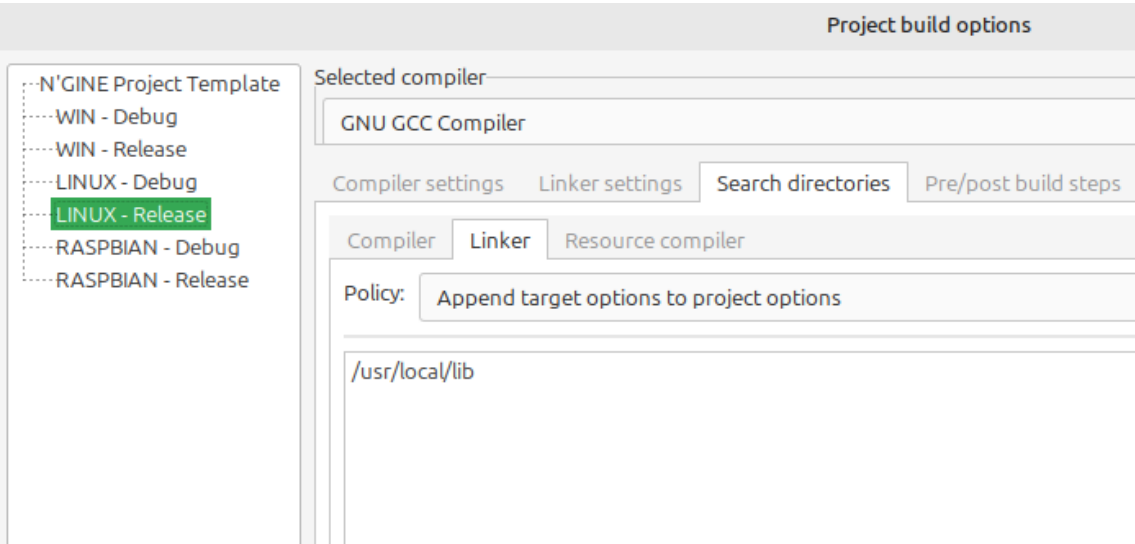
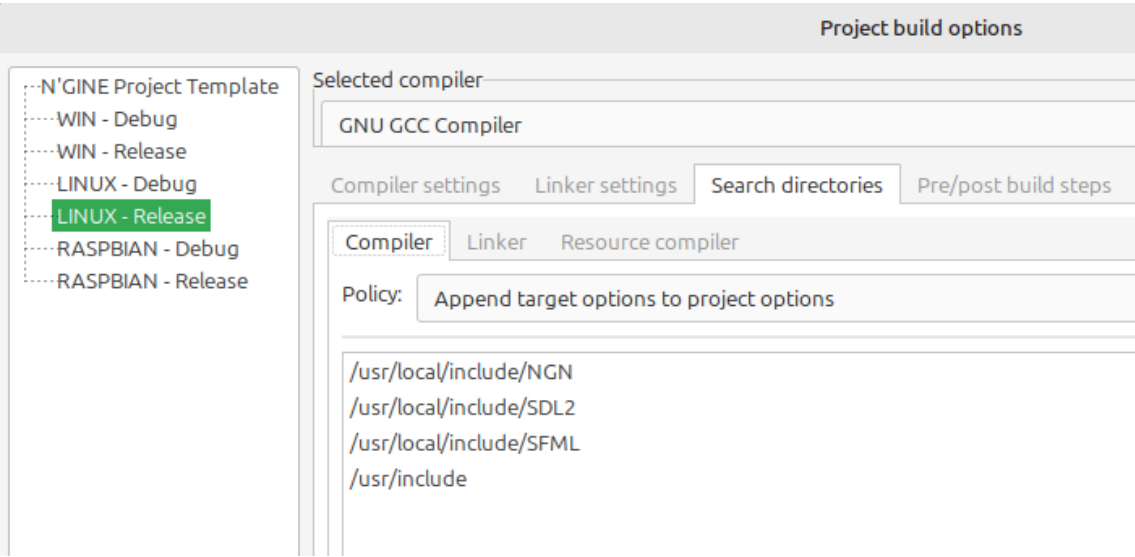
En “Project build options”, en la sección “LINUX-Debug”, configuraremos los siguientes parámetros:



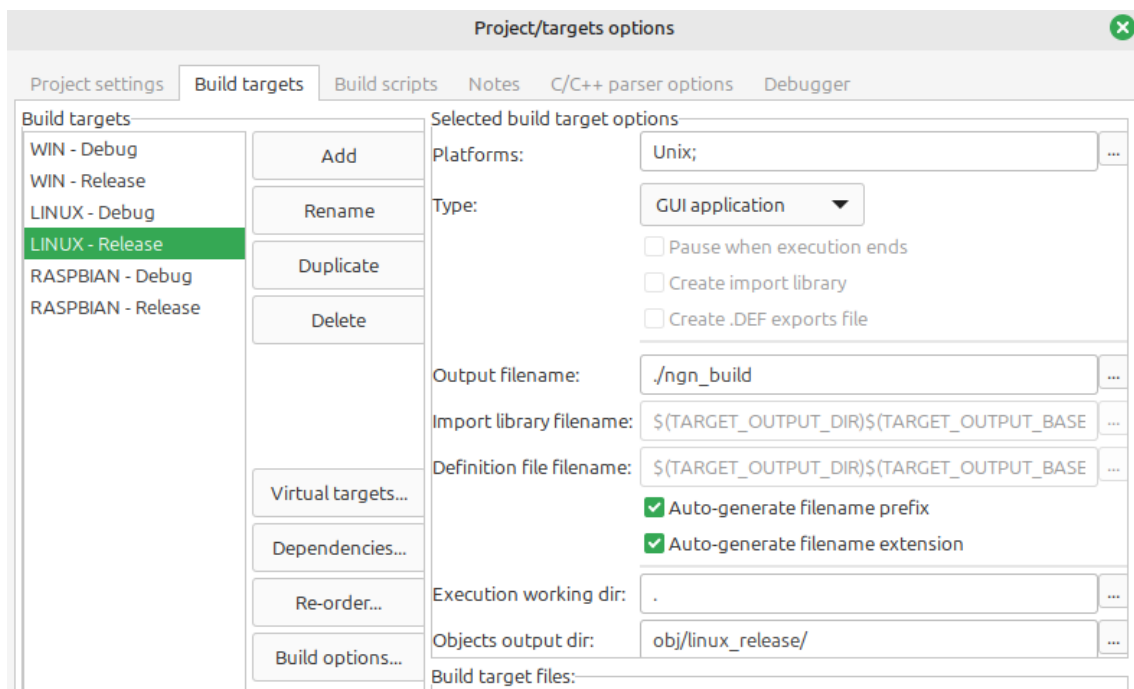
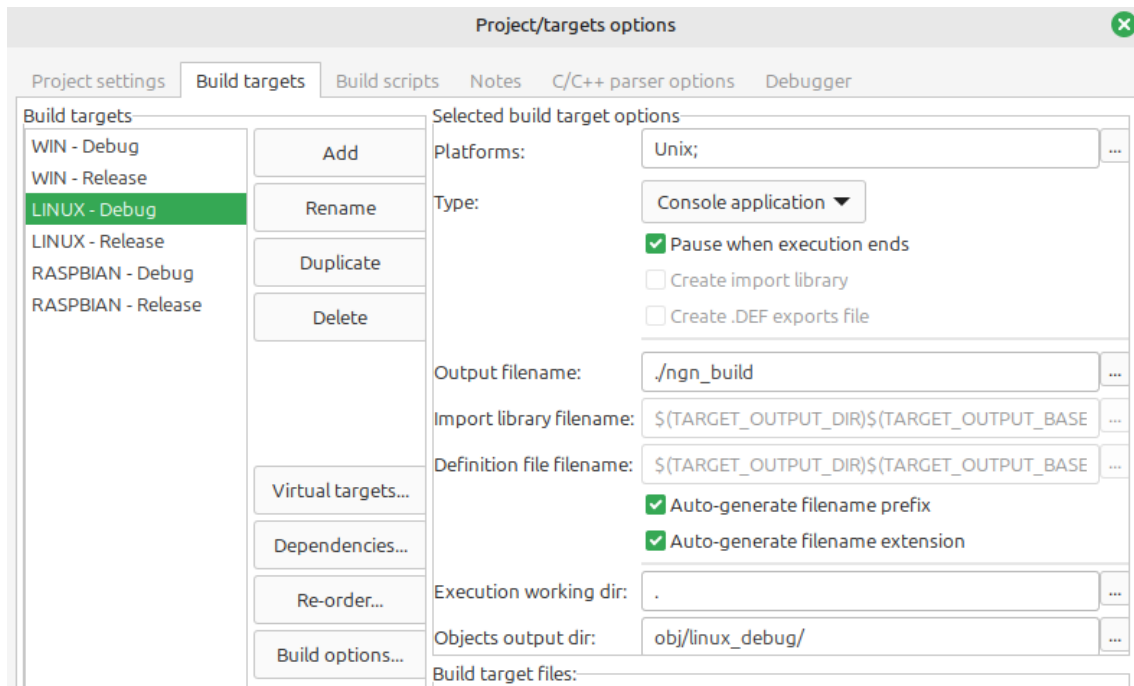
En “Project build options”, en la sección “LINUX-Release”, configuraremos los siguientes parámetros:



En “Project build options” configuraremos tanto en la sección “LINUX-Debug” como en “LINUX-Release” los siguientes parámetros:



En “Project/target options” configuraremos los siguientes parámetros:



Opcionalmente, la librería incluye diversas plantillas con todas estas opciones ya configuradas, tanto para Windows como para Linux y Raspberry PI OS. En caso de usar estas plantillas en la creación de los proyectos, solo será necesario realizar el paso de instalación de las librerías.

Para los entornos Linux Mint y Raspberry PI OS se incluyen también unos scripts de instalación automática con las ultimas versiones de la librería N'gine, así como las versiones usadas de SDL2, SDL2\_ttf y SFML2. Estas versiones se instalarán en las carpetas “/usr/local” correspondientes y puede que eliminen las versiones existentes de dichas librerías.

- Fecha de revisión: lunes, 13 de enero del 2025.