How config CLANG Compiler (32 and 64 bits) (into MSYS2) into CodeBlocks

Full name of tutorial : How config CLANG Compiler (32 and 64 bits) (+ MSYS2) into Code::Blocks on Windows 11 64 bits.

```
Code::Blocks : the best and great free IDE for Windows, Linux and ... Mac OS
```

Presentation of CLANG into MSYS2

During first run of CB on Windows, this IDE detect automatically some compilers, or present one list of them pre-configured.
It's very good functionnality, but, sometimes, you must "force" these configurations proposed by default to run correctly.
This tuto describe how configure one major compiler C\C++ on Windows : CLANG/LLVM Compiler (32 and 64 bits) included in MSYS2.

CLANG is rigourously a C/C++ (very good) compiler without libraries or "include files" needed to success generation.
On Windows systems, it's mandatory to associate CLANG with another development environment like MSVC + SDK Windows, or MinGW32/64 (most possibilities).

CLANG/LLVM Compiler (32 and 64 bits) is available directly in package MSYS2 (if you want choose it).
Open an "MSYS2 MSYS" console in the list of installed applications on your system and type :
"pacman -S mingw32/mingw-w64-i686-llvm mingw-w64-x86_64-llvm"

NB : If you want update all your applications/packages into MSYS2, use this "magic command" :
"pacman -Syuu"

Configuration of CLANG included in MSYS2 into CB

Normally, next run of IDE Code::Blocks must detect CLANG/LLVM compiler and propose it into list of available compilers on system. Choose "Settings" in main menu and after select

"Compilers", and verify if you find "LLVM Clang Compiler" in this list or an compiler near of it in name "LLVM Clang Compiler (64 bits)".
Then click to "Copy" button and after "Rename" button to change result name of "new" compiler like "LLVM Clang Compiler W64 (64 bits)" (by example).

An after, you must verify field "Compiler's installation directory" that must contain
"C:\msys64\mingw64" (subdirectory .\bin search of binaries by default)

You must verify list of tools like this (in subtab "Program Files") :
- C Compiler : clang.exe
- C++ Compiler : clang++.exe
- Linker for dynamic libs : clang++.exe
- Linker for statics libs : llvm-ar.exe
- Debugger : GDB (default)
- Resource compiler : llvm-rc.exe
- Make program : mingw32-make.exe

And, it's not all, you must verify in tab "Search Directories" and select "Compiler", or "Linker" or "Resource Compiler" subtabs.

For compiler, search directories of "include files" is (and it's the same for "Resource Compiler") C:\msys64\mingw64\include, and for linker, search directories of "lib files" is : C:\msys64\mingw64\lib

To terminate with 64 bits version of CLANG/LLVM, you must select in tab "Compiler settings" (or in zone "Other Compiler options") :"-m64". CLANG/LLVM must position automatically this option, but it's a precaution.

Result of command "C:\msys64\mingw64\bin\clang++.exe --version", must be :
clang version 18.1.8
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: C:/msys64/mingw64/bin

Now, you must copy "LLVM Clang W64 (64 bits)" in list of available compilers into CB (menu "Settings" submenu "Compilers"), and rename it by "LLVM Clang W32 (32 bits)" by example.

New configurations for this compiler in version 32 bits are next :

In tab "Toolchain executable", you must change value by :
"C:\msys64\mingw32" (subdirectory .\bin search of binaries by default)

In tab "Search directories", for compiler, search directories of "include files" is (and it's the same for "Resource Compiler") C:\msys64\mingw32\include,, and for linker, search directories of "lib

files" is : C:\msys64\mingw32\lib

And, to terminate with 32 bits version of CLANG/LLVM, you must select in tab "Compiler settings" (or in zone "Other Compiler options") : "-m32". CLANG/LLVM must position automatically this option, but it's a precaution.

Result of command "C:\msys64\mingw32\bin\clang++.exe --version", must be :
clang version 18.1.8
Target: i686-w64-windows-gnu
Thread model: posix
InstalledDir: C:/msys64/mingw32/bin

Test of "simple" code with CLANG included in MSYS2 into CB

And, with simply source "hellowworld.c", you can test generation of program into IDE CB, choosing "create new project" in main windows of CB, and choose "console application" with no source proposed by default, because named "main.c" by default, and choose compiler "LLVM Clang Compiler W64 (64 bits)".
You can select good directory/source with option "add file" after first creation of project into CB.

One time project created, you can generate it with selecting main menu "Build" and choose submenu "Rebuild..." (or CTRL-F11).

Pleasure of programming is open for you, your imagination is illimited, at your keyboard ! Enjoy !

PS : source file "hellowworld.c" :

```
/* Basic example in language C : hellowworld.c */

#include <stdio.h>

int main(int argc, char *argv[]) {
/* printf() displays the string inside quotation */
printf("Hello, World!");
return 0;
}
```

PS2 : Use of CLANG compiler included in MSYS2 in command line (just to illustrate)

You can also use CLANG compiler included in MSYS2 package in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set PATH=C:\msys64\mingw64\bin;%PATH%
REM set PATH=C:\msys64\mingw32\bin;%PATH% if you want use CLANG + MinGW32 (32bits)

REM Compile + link in one pass
clang hellowworld.c -o hellowworld.exe
REM Compile + link in two pass
clang -c hellowworld.c -o hellowworld.obj --target=x86_64-w64-windows-gnu
clang hellowworld.obj -o hellowworld.exe -Wl,-subsystem=console --target=x86_64-w64-windows-gnu
```

Continue with use of CLANG compiler, and don't forgive, at the end of your work, to return to initial state :

```
set PATH=%PATHSAV%
```

And, with precedent example, you can also generate version 32 bits with "clang" but, you must change value of PATH with good
directory described before (don't forgive to change "--target=i686-w64-windows-gnu" during call of "clang" if two pass).

But, it's much easy to use CLANG/LLVM included in MSYS2 directly into CB IDE especially with complex C program (many C sources and many subdirectories ...) , and multiple targets by example : main DLL and console program to test this DLL, ... -)

PS3 : Syntaxe of CLANG/LLVM compiler

Syntax and list of options of CLANG/LLVM compiler "clang" is very ... long ...
To simplify, it's preferable to list these options in text file like this "clang -help > command_clang.txt" and search that it you interest in this text file.

You can consult many documentation on site : https://clang.llvm.org/docs/UsersManual.html