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

Full name of tutorial : How config CLANG Compiler (32 and 64 bits) (+ package WinLibs) 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 package Winlibs

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 Winlibs.

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 ...).

Installation of CLANG/LLVM Compiler (32 and 64 bits) is included in package WINLIBS (if you want choose it).

Packages are available on site : https://winlibs.com/

Two files are needed to download in version 32 and 64 bits :
"winlibs-x86_64-posix-seh-gcc-14.1.0-llvm-18.1.8-mingw-w64ucrt-12.0.0-r3.7z" (64 bits) and "winlibs-i686-posix-dwarf-gcc-14.1.0-llvm-18.1.8-mingw-w64ucrt-12.0.0-r3.7z" (32 bits) (last version available mid 2024)

After download, click on these files to install required components on your system.
Version 32 bits is installed on directory "C:\mingw32" by default, and version 64 bits is installed on directory "C:\mingw64" by default, and it's a recommandation, and version of compiler gcc is 14.1.0 (14.2.0 not available during generation of packages).

Configuration of CLANG included in package Winlibs into CB

Search if you find "LLVM CLANG Compiler" in this list of available compilers by choose main menu "Settings" and after menu "Compilers".
You must first copy it and rename this ident of compiler in "LLVM Clang Winlibs (64 bits)" (by example).

An after, you must verify field "Compiler's installation directory" that must contain
- "C:\mingw64\bin"

You must verify list of tools like this (in tab "Program Files") :
- C Compiler : clang.exe
- C++ Compiler : clang++.exe
- Linker for dynamic libs : clang++.exe
- Linker for statics libs : llvm-lib.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" are (and it's the same for "Resource Compiler") :
C:\mingw64\x86_64-w64-mingw32\include and
C:\mingw64\include

For "Linker", search directories of "lib files" are :
C:\mingw64\x86_64-w64-mingw32\lib and
C:\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:\mingw64\bin\clang.exe --version", must be :
(built by Brecht Sanders, r3) clang version 18.1.8
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: C:/mingw64/bin

Now, you must copy "LLVM Clang Winlibs (64 bits)" in list of available compilers into CB (menu "Settings" submenu "Compilers"), and rename it by "LLVM Clang Winlibs (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:\mingw32\bin"

In tab "Search Paths", changes are mandatory in subtab "Compiler" and "Resource compiler" : replace
"C:\mingw64\x86_64-w64-mingw32\include" by "C:\mingw32\i686-w64-mingw32\include" and "C:\mingw64\include" by "C:\mingw32\include"

And, changes in tab "Linker" are next :
"C:\mingw32\i686-w64-mingw32\lib" and ""C:\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.

Test of "simple" code with CLANG included in package Winlibs 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 Winlibs (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).

You can repeat this generation after change compiler into CB to point to "LLVM Clang Winlibs (32 bits)" and use CTRL-F11 to rebuild.

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 package Winlibs in command line (just to illustrate)

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

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

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
clang hellowworld.obj -o hellowworld.exe
```

Continue with use of MSVC 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.

But, it's much easy to use CLANG/LLVM included in Winlibs 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 : Syntax of tools CLANG/LLVM

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