How config MinGW64 GCC C_C++ compiler into CodeBlocks

Name of tutorial : How config MinGW64 GCC C/C++ compiler into Code::Blocks on Windows 11 64 bits.

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

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 compilers GCC included in MSYS2 MinGW32/MinGW64 environment on Windows.

MinGW : A native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header
files for building native Windows applications; includes extensions to the MSVC runtime to support C99 functionality.
All of MinGW's software will execute on the 64bit Windows platforms.

How to install MinGW64/MinGW32 on Windows 11 64 bits ?

You can download last version of MinGW64/MinGW32 on Internet site :
https://github.com/niXman/mingw-builds-binaries/releases

Warning, many versions of MinGW64/MinGW32 are available on this site, and it's important to understand differences beetween their :
one category is based on UCRT (run-time C universal),
another category is based on MSVCRT (run-time C owned by Microsoft Corporation)
and another differentiations into versions is based on "multithread" package :
"posix" ("native POSIX" : pthread)
"win32" (portage of Posix thread on Win32 : winpthread)
"MCF" (new MCF thread model : mcfgthread)

To understand theses differences, it's important to note that :
"posix" : enable C++11/C11 multithreading features. Makes libgcc depend on libwinpthreads, so that even if you don't directly
call pthreads API, you'll be distributing the winpthreads DLL. There's nothing wrong with

distributing one more DLL
with your application.
"win32" : No C++11 multithreading features.
"MCF" : new threading included in MinGW by Brecht Sanders during october 2022

The pros of winpthreads are :
winpthreads supports Windows XP; mcfgthread only supports Vista,
winpthreads provides more complete POSIX APIs, such scheduler and RW locks; mcfgthread only provides those required by libgcc
and libstdc++.

On the other hand, the pros of mcfgthread are :
mcfgthread is generally more efficient, outperforming native SRWLOCKs; winpthreads' condition variable and thread-specific
storage access is slow,
mcfgthread provides slim mutex and condvar, which take up storage as pointers, consume no additional resource, and require
no cleanup,
mcfgthread provides __cxa_finalize() as per Itanium ABI.

All versions are declined in 32 bits or 64 bits version.

To focalize on 64 bits version, you can download file "x86_64-13.2.0-release-win32-seh-ucrt-rt_v11-rev1.7z" and after decompress
it on directory C:\niXman\MinGW64, version of GCC 64 bit installed after is 13.2.0.
Why not C:\MinGW64 ? Because package "WinLibs" need to conserve C:\mingw32 and C:\mingw64 free to use (mandatory).

You must configure this new compiler into CB by selecting main menu "Settings" then submenu "Compiler..." into IDE interface,
then, choose "GNU GCC Compiler (default)", copy it by button "Copy" and rename this like "MinGW64 GCC Compiler" by example,
with button "Rename".

Then, you choose tab" "Toolchain executable" to position good environment like this :

Toolchain executable :
C:\niXman\MinGW64 (subdirectory \bin will be searched automatically to access to binaries listed after)
compilateur C : gcc.exe
compilateur C++ : g++.exe
linker for dynamic lib : g++.exe

linker for static lib : gcc-ar.exe
debugger : gdb.exe
resource compiler : windres.exe
make program : mingw32-make.exe

Then, you choose tab" "Search directories" to position good environment like this :

```
to compiler :                    C:\niXman\MinGW64\x86_64-w64-mingw32\include
and C:\niXman\MinGW64\include
to linker :                      C:\niXman\MinGW64\x86_64-w64-mingw32\lib
and C:\niXman\MinGW64\lib
to resource compiler :  C:\niXman\MinGW64\x86_64-w64-mingw32\include and
C:\niXman\MinGW64\include
```

Then, you can type this into command console Windows : "C:\niXman\MinGW64\bin\g++.exe" --version; result here :

g++.exe (x86_64-win32-seh-rev1, Built by MinGW-Builds project) 13.2.0
Copyright (C) 2023 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.

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" is proposed by default,
and choose compiler "MinGW64 GCC Compiler".
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).
If, you apply all of precedent instructions, compile and link of your program with this new compiler must be succeeded.

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 : You can also use compiler GCC in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set PATH=C:\niXman\MinGW64\bin;%PATH%
REM Next instructions are not mandatory, but you can set var if you want.
REM set CLANG=C:\niXman\MinGW64\x86_64-w64-
mingw32\include;C:\niXman\MinGW64\include
REM set LIBRARY_PATH=C:\niXman\MinGW64\x86_64-w64-
mingw32\lib,C:\niXman\MinGW64\lib

REM Generate console application in one pass
gcc hellowworld.c -o hellowworld.exe
REM Generate console application in two pass
gcc -c hellowworld.c -o hellowworld.obj
gcc hellowworld.obj -o hellowworld.exe -Wl,--subsystem console
```

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

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

But, it's much easy to use GCC of MinGW64 directly into CB IDE especially with complex C program (many C sources
and many subdirectories ...) -)

PS3 : Command "gcc" present a very "verbose" list of options, but to resume principal and useful options, you can use
by example :
"-m16" Generate i386 16 bits object or executable.
"-m32" Generate i386 32 bits object or executable.
"-m64" Generate x64 64 bits object or executable.
"-c" Compile and assemble, but do not link.
"-D var[=value]" Define variable to be use by préprocessor, and optionnally affect an value at this variable.
"-o " Place the output into .
"-I " Add directory to search include files
"-L " Add directory to search library files

"-shared" Create a shared library.
"-llibrary" Give name of library used by linker.
"-pthread" Link with the POSIX threads library.
"-Wl," Pass comma-separated on to the linker. Example of useful options :
-t trace all input files used by linker
--kill-at not add the function name decorations at-sign and number for stdcall functions
--add-stdcall-alias export functions with the stdcall decoration suffix (@nn) and also without this suffix
--dll create DLL on Win32 systems
--export-all-symbols all global symbols in the objects used to build a DLL will be exported by the DLL
--output-def file create a def file of all exported sysmbols
--out-implib file create a import library in parallel of creation of shared library
--subsystem ident create target based on "ident" that can be valued by "native, windows, console, posix, or xbox"

You can consult an summary of these options on site :
https://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
And for "ld" (linker of gcc), you can consult too : http://sourceware.org/binutils/docs-2.16/ld/Options.html