How config MSYS2 MinGW32_MinGW64 GCC C_C++ compiler into CodeBlocks

Full name of tutorial : How config MSYS2 MinGW32/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
```

Presentation of MinGW32/MinGW64 included in package 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 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.

MSYS2 : It's a portage of MSYS to actualised all packages included in MinGW32 and MinGW64 with latest versions of software.

How to install MSYS2 MinGW32/MinGW64 on Windows 11 64 bits ?

By example, you can install MSYS2 available on this site https://www.msys2.org/, last version dated july 2024 :
"msys2-x86_64-20240727.exe"

On Windows 11 system, 64 bits, MSYS2 is installed by default on directory : C:\msys64.

After first install, you must add all needed software by use of tool "pacman" in console MSYS2 by open this in list of installed software on Windows 11 : clic in task bar on "windows" icon, then choose "all" on the rigth of the new screen and select application "MSYS2" and submenu "MSYS2 MSYS".

Open this console (or command file) and type next commands ("-toolchain" because install many useful complements) :

```
pacman -S mingw-w64-i686-toolchain
pacman -S mingw-w64-x86_64-toolchain
pacman -S mingw-w64-ucrt-x86_64-toolchain
pacman -S mingw-w64-x86_64-doxygen, grep, sed, bash, mingw-w64-x86_64-ninja,
msys/make mingw32/mingw-w64-i686-clang,
mingw-w64-x86_64-clang
```

So, with MSYS2, you can use multiple development environment, 32 or 64 bits (ARM64 not mandatory on X32/X64 architecture) :

Name Directory Compiler Architecture RT C base Default lib C++

- MSYS /usr gcc x86_64 cygwin libstdc++
- UCRT64 /ucrt64 gcc x86_64 ucrt libstdc++
- CLANG64 /clang64 llvm/clang x86_64 ucrt libc++
- CLANGARM64 /clangarm64 llvm/clang aarch64 ucrt libc++
- CLANG32 /clang32 llvm/clang i686 ucrt libc++
- MINGW64 /mingw64 gcc x86_64 msvcrt libstdc++
- MINGW32 /mingw32 gcc i686 msvcrt libstdc++

Warning : To update with latest versions of software after first install, you must rerun "pacman" in console MSYS2 MSYS like this : "pacman -Syuu". It's suffisant to update all packages and softwares installed on your configuration. Simply.

Configuration of MinGW32/MinGW64 included in package MSYS2 into CB

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)" and rename this like "MSYS2 MinGW32 32bit" by example.

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

Toolchain executable :
C:\msys64\mingw32 (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:\msys64\mingw32\i686-w64-
mingw32\include and C:\msys64\mingw32\include
- to linker :                    C:\msys64\mingw32\i686-w64-mingw32\lib     and
C:\msys64\mingw32\lib
- to resource compiler :         C:\msys64\mingw32\i686-w64-mingw32\include and
C:\msys64\mingw32\include
```

Then, you can type this into command console Windows : "C:\msys64\mingw32\bin\g++.exe --version"; result here :

g++.exe (Rev1, 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.

To terminate, you must reiterate last instructions to configure MinGW64 included in MSYS2 environment.

You must configure this new compiler into CB by selecting main menu "Settings" then submenu "Compiler..." into IDE interface, then, choose "MSYS2 MinGW32 32bit" and rename this like "MSYS2 MinGW64 64bit" by example.

After, choose tab" "Toolchain executable" to position good environment like this

Change Toolchain executable to :
C:\msys64\mingw64 (subdirectory \bin will be searched automatically to access to binaries listed after)

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

```
- to compiler :                         C:\msys64\mingw64\x86_64-w64-
mingw32\include and C:\msys64\mingw64\include
- to linker :                   C:\msys64\mingw64\x86_64-w64-mingw32\lib
and C:\msys64\mingw64\lib
- to resource compiler :        C:\msys64\mingw64\x86_64-w64-mingw32\include
and C:\msys64\mingw64\include
```

To verify, you can repeat instruction with "very simply" code "hellowworld.c" like with MinGW32, only choose good compiler to generate program.

Test of "simple" code with MinGW32/MinGW64 includes in package MSYS2 into CB

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 "MSYS2 MinGW32 32bit".

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), and save your project (or "save everythings").

To test 64 bits, return into main menu "Project" and in submenu "Build options" to change the compiler to "MSYS2 MinGW32 64bit" and rebuild with 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 : Use of compilers GCC of MinGW32/MinGW64 included in package MSYS2 on command line (just to illustrate)

You can also use compiler GCC of MinGW64 include in MSYS2 in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set PATH=C:\msys64\mingw64\bin;%PATH%
REM Next instructions are not mandatory, but you can set var if you want.
REM set CLANG=C:\msys64\MinGW64\x86_64-w64-
mingw32\includeC:\msys64\mingw64\include
REM set LIBRARY_PATH=C:\msys64\mingw64\x86_64-w64-
mingw32\lib,C:\msys64\mingw64\include

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%
```

And, with precedent example, you can also use version GCC 32 bits of MSYS2.

But, it's much easy to use Digital Mars Compiler C/C++ 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 : Principal syntax of tools GCC

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 preprocessor, 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 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