How config TDM MinGW32 MinGW64 into CodeBlocks

Full name of tutorial : How config TDM MinGW32/MinGW64 into Code::Blocks on Windows 11 64 bits.

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

Presentation of TDM MinGW32/MinGW64

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 TDM MinGW32 and MinGW64 into Code::Blocks on Windows 11 64 bits.

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 GNU GCC Compiler C/C++ included TDM MinGW32/MinGW64 (version gcc 10.3.0, 64 bits) on Windows 11 64 bits ?

You can download last version of TDM MinGW32 and TDM MinGW64 on Internet site :
https://jmeubank.github.io/tdm-gcc/download/

Two files for download (if you want 32 bits and 64 bits version) :
tdm64-gcc-10.3.0-2.exe : 64+32-bit MinGW-w64 edition. Includes GCC C/C++, GNU binutils, mingw32-make, GDB (64-bit), the MinGW-w64 runtime libraries and tools, and the windows-default-manifest package.
tdm-gcc-10.3.0.exe : 32-bit-only MinGW.org edition. Includes GCC C/C++, GNU binutils,

mingw32-make, GDB (32-bit), the MinGW.org mingwrt and w32api packages, and the windows-default-manifest package.

Click on these executables to install two version of TDM MinGW32 and TDM MinGW64 on your Windows system :
tdm64-gcc-10.3.0-2.exe : default installed directory on C:\TDM-GCC-64
tdm-gcc-10.3.0.exe : default installed directory on C:\TDM-GCC-32

Configuration of TDM MinGW32/MinGW64 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 "TDM GNU GCC Compiler 64bit" by example, if it's not detected.

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

Toolchain executable :
C:\TDM-GCC-64 (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 (default)
- resource compiler : windres.exe
- make program : mingw32-make.exe

You must add another directory into tab "Toolchain executable" with subtab "Additional paths" : C:\TDM-GCC-64\gdb64\bin.
The name of executable associated with gdb is "gdb.exe", good news (see configuration of TDM MinGW32 to understand why).

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

```
- to compiler :                              C:\TDM-GCC-64\include
- to linker :                          C:\TDM-GCC-64\lib
- to resource compiler :         C:\TDM-GCC-64\include
```

Then, you can type this into command console Windows : "C:\TDM-GCC-64\bin\g++.exe --version"; result here :

g++.exe (tdm64-1) 10.3.0

For TDM MinGW32, you must return into menu "Settings" of CB and select "TDM GNU GCC Compiler 64bit" compiler for copy it first,
and then, first copy this compiler, and, to terminate, rename to "TDM GNU GCC Compiler 32bit" by example.

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

Toolchain executable :
C:\TDM-GCC-32 (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 (default)
- resource compiler : windres.exe
- make program : mingw32-make.exe

You must add another directory into tab "Toolchain executable" with subtab "Additional paths" : C:\TDM-GCC-32\gdb32\bin.
But, the name of executable associated with gdb is "gdb32.exe" not "gdb.exe" and you can't change this name into interface CB.
But, to resolve this, you can copy, in this directory, "gdb32.exe" to "gdb.exe", and "gdb32server.exe" to "gdbserver.exe".
Or create two symlinks pointed on these two executables.

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

```
- to compiler :                    C:\TDM-GCC-32\include
- to linker :                      C:\TDM-GCC-32\lib
- to resource compiler :        C:\TDM-GCC-32\include
```

Test of "simple" code with TDM MinGW32/MinGW64 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", and choose compiler "TDM GNU GCC Compiler 64bit".
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").

You can test with precedent project console C create into CB, after change compiler to "TDM GNU GCC Compiler 32bit" and rebuild it.

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 TDM MinGW32/MinGW64 directly on command line (just to illustrate)

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

```
set PATHSAV=%PATH%
set PATH=C:\TDM-GCC-64\bin;%PATH%
REM set PATH=C:\TDM-GCC-32\bin;%PATH% to use MinGW32 issued of TDM

REM Generate console application in one pass
gcc hellowworld.c -o hellowworld.exe
```

REM Generate console application in two pass, compile only first, and link to terminate.
gcc -c hellowworld.c -o hellowworld.obj
gcc hellowworld.obj -o hellowworld.exe -Wl,--subsystem=console

Continue with use of TDM 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 compiler of MinGW64 included in Red-Panda 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.
"-mwindows" Create GUI application on Win32 systems (entry point of program "WinMain", not "main")
"-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 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"

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