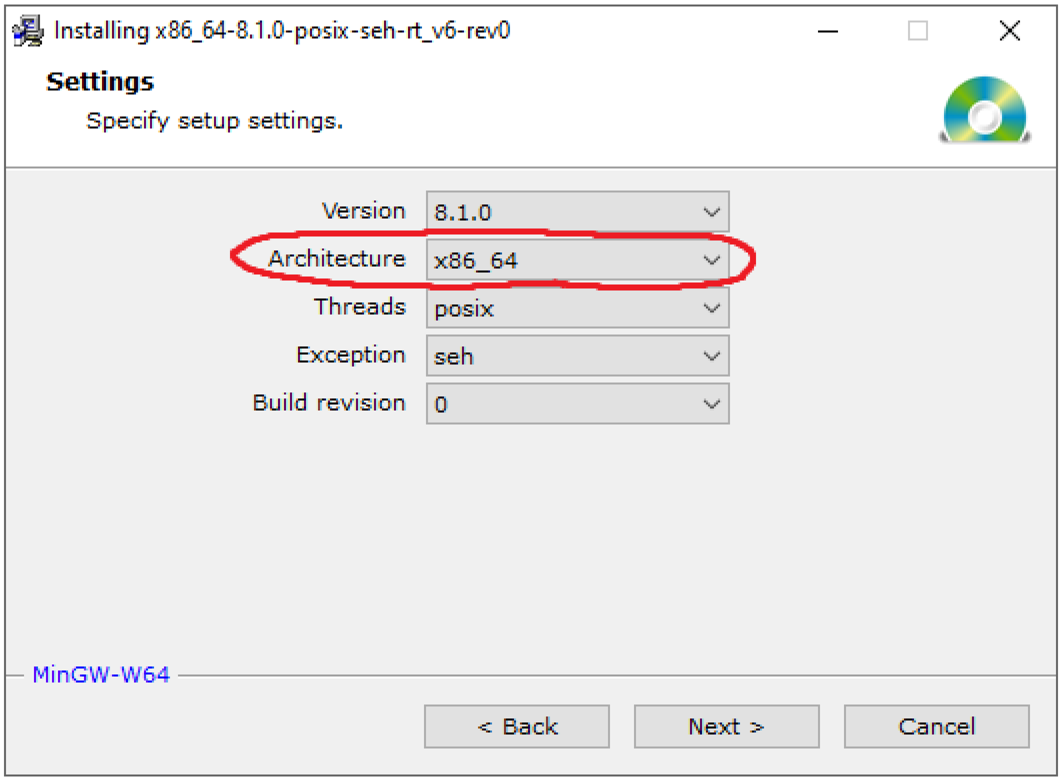
How to Setup SDL2 on Windows for C/C++

This tutorial will go through the process of setting up the SDL2 library on Windows for C/C++ development with mingw-w64, which is a port of the GCC compiler for Windows.

Our example will be written in C using a 64-bit compiler but this works exactly the same for C++ and one could easily use a 32-bit compiler instead.

Step 1: Installing mingw-w64

The first step is to download mingw-w64, and during the install, make sure to install the 64-bit compiler *x86_64*, as shown below.


Installing x86_64-8.1.0-posix-seh-rt_v6-rev0

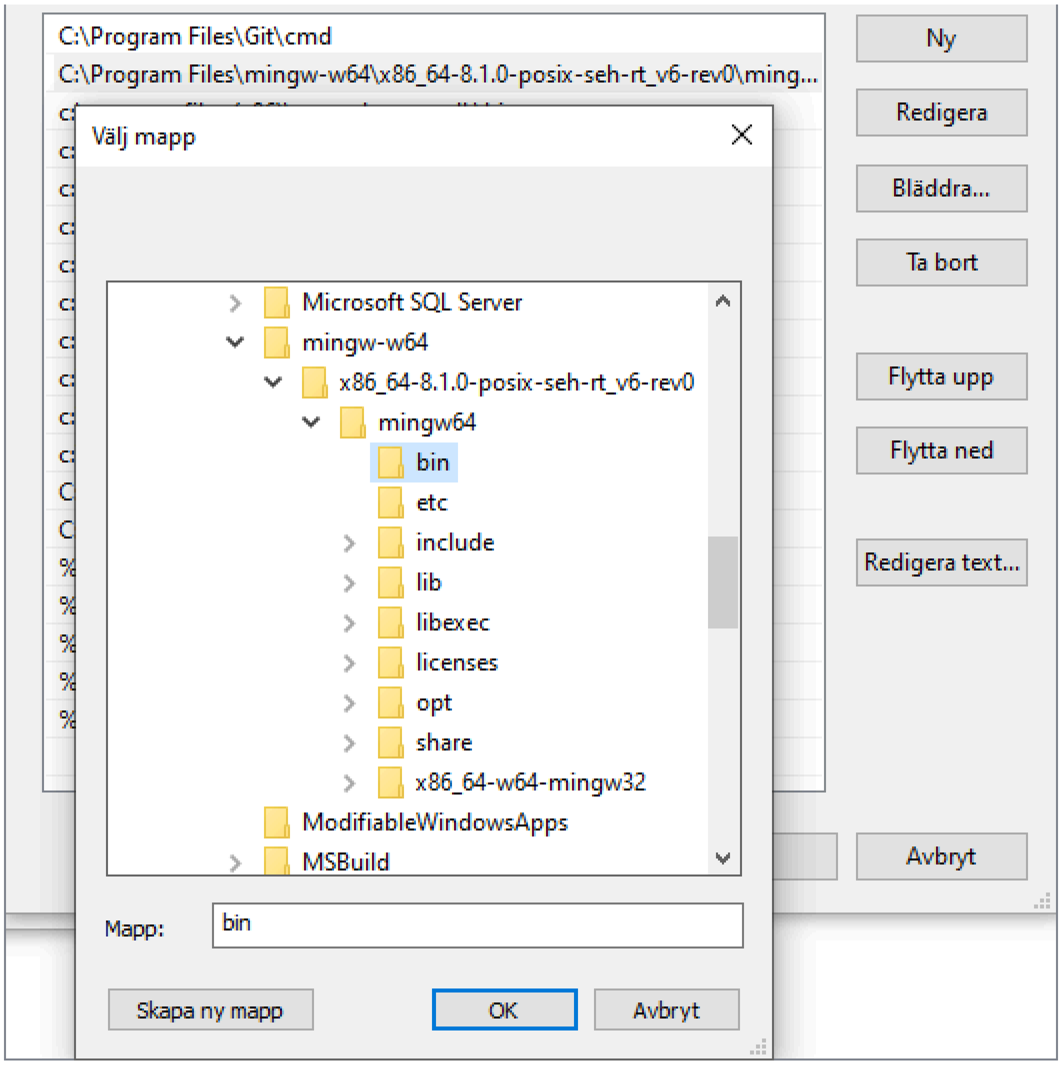
Settings

Specify setup settings.

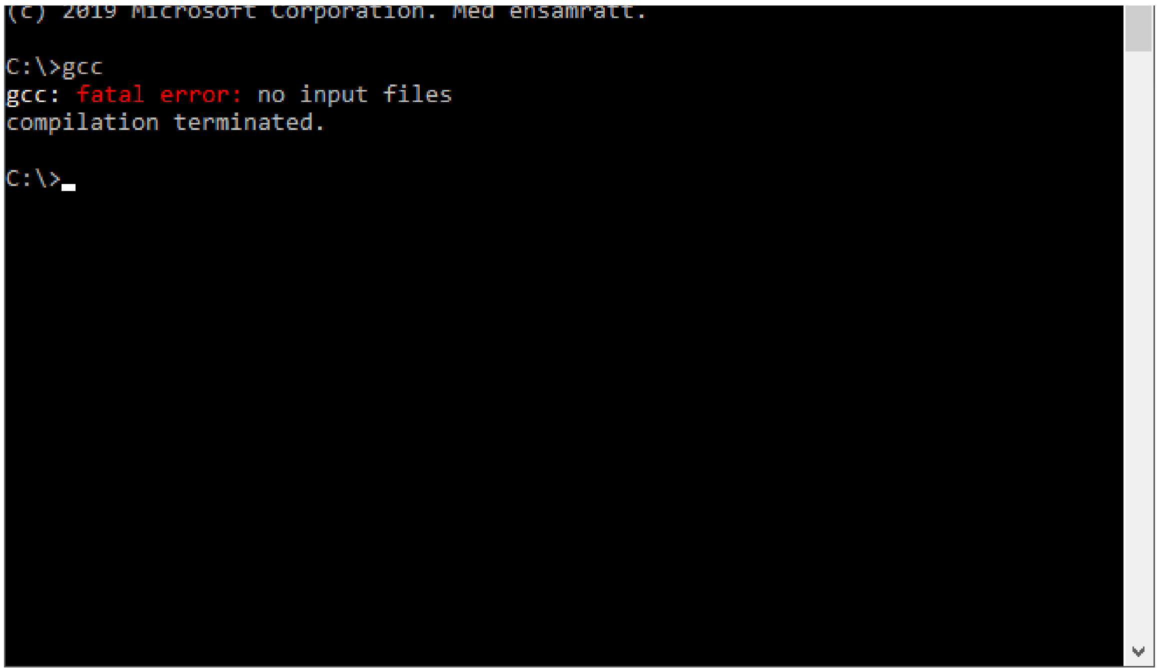
Version	8.1.0
Architecture	x86_64
Threads	posix
Exception	seh
Build revision	0

After the installer is done we need to add the mingw directory to PATH.

Open the start menu and search for "Edit the system environment variables" -> click "Environment Variables" -> select "Path" under System variables and click "Edit" -> add mingw64\bin



Now we can make sure everything works correctly by opening cmd and typing in the gcc command.



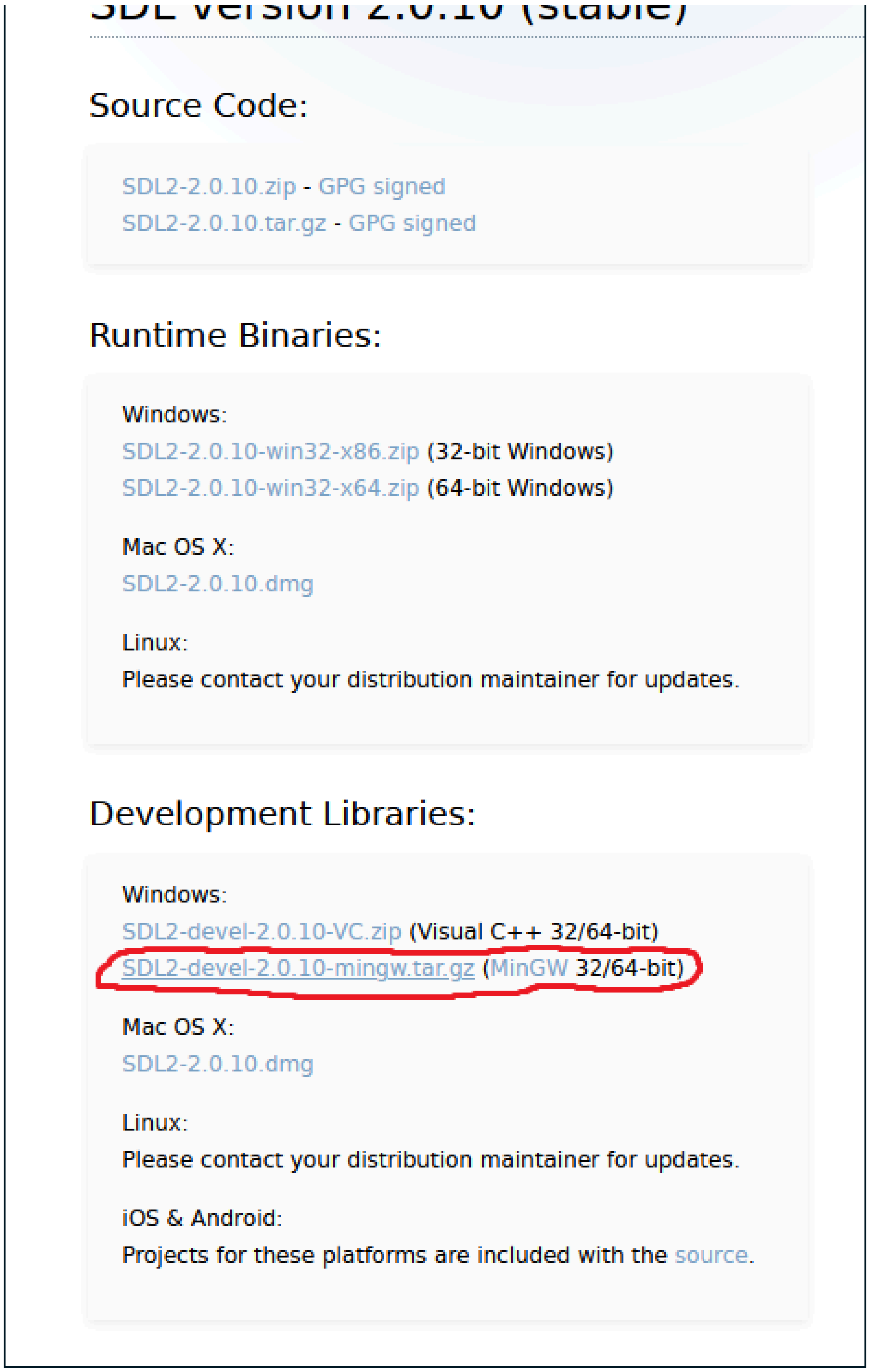
```
(c) 2019 Microsoft Corporation. Med ensamrätt.

C:\>gcc
gcc: fatal error: no input files
compilation terminated.

C:\>_
```

Step 2: Installing SDL2

Go to the SDL2 download page and download the latest development library for Windows using MinGW.


SDL version 2.0.10 (stable)

Source Code:

SDL2-2.0.10.zip - GPG signed
SDL2-2.0.10.tar.gz - GPG signed

Runtime Binaries:

Windows:
SDL2-2.0.10-win32-x86.zip (32-bit Windows)
SDL2-2.0.10-win32-x64.zip (64-bit Windows)

Mac OS X:
SDL2-2.0.10.dmg

Linux:
Please contact your distribution maintainer for updates.

Development Libraries:

Windows:
SDL2-devel-2.0.10-VC.zip (Visual C++ 32/64-bit)
SDL2-devel-2.0.10-mingw.tar.gz (MinGW 32/64-bit)

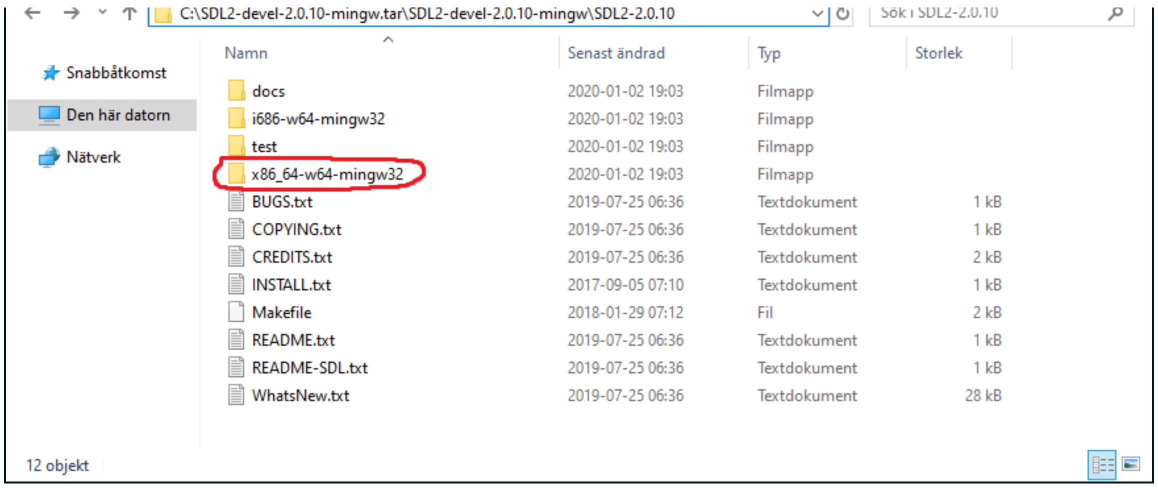
Mac OS X:
SDL2-2.0.10.dmg

Linux:
Please contact your distribution maintainer for updates.

iOS & Android:
Projects for these platforms are included with the source.

(this tutorial uses SDL2-devel-2.0.10-mingw.tar.gz)

After extracting the contents using for example 7-Zip, copy the folder *"x86_64-w64-mingw32"*, to where you want to store the library.
Note that this is still the 64-bit version of the library.

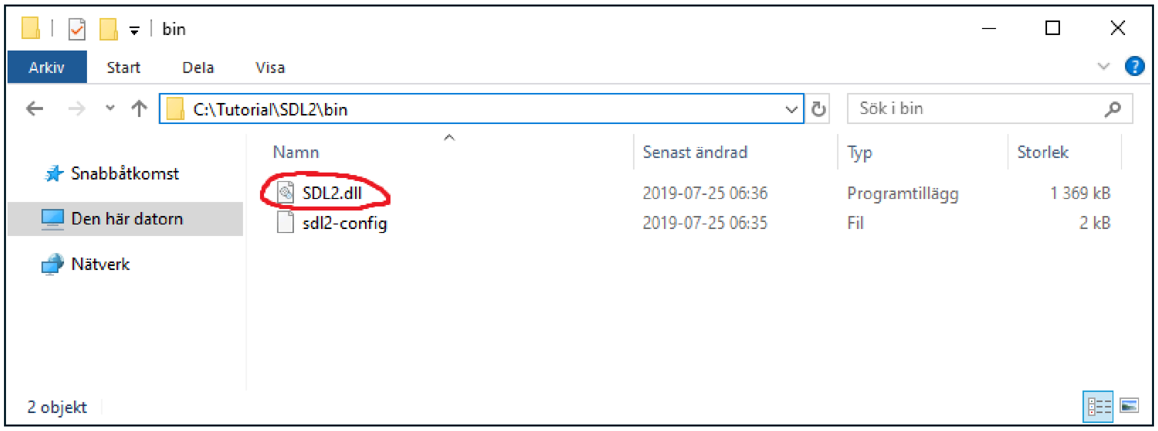

C:\SDL2-devel-2.0.10-mingw.tar\SDL2-devel-2.0.10-mingw\SDL2-2.0.10

Namn	Senast ändrad	Typ	Storlek
docs	2020-01-02 19:03	Filmapp	
i686-w64-mingw32	2020-01-02 19:03	Filmapp	
test	2020-01-02 19:03	Filmapp	
x86_64-w64-mingw32	2020-01-02 19:03	Filmapp	
BUGS.txt	2019-07-25 06:36	Textdokument	1 kB
COPYING.txt	2019-07-25 06:36	Textdokument	1 kB
CREDITS.txt	2019-07-25 06:36	Textdokument	2 kB
INSTALL.txt	2017-09-05 07:10	Textdokument	1 kB
Makefile	2018-01-29 07:12	Fil	2 kB
README.txt	2019-07-25 06:36	Textdokument	1 kB
README-SDL.txt	2019-07-25 06:36	Textdokument	1 kB
WhatsNew.txt	2019-07-25 06:36	Textdokument	28 kB

12 objekt

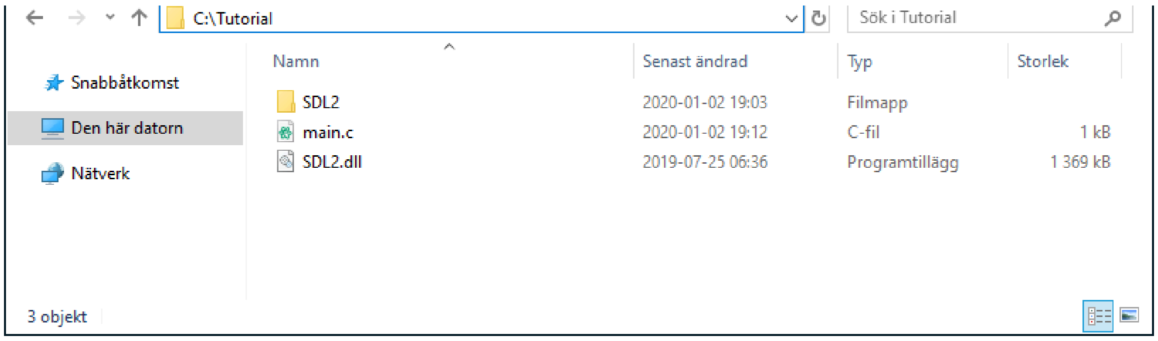
This folder contains the include- and library files needed for compiling, as well as the *SDL2.dll* file that we need to distribute along with the final compiled .exe file.
For the sake of this tutorial, we will rename this folder *SDL2* and copy it into our project folder *C:\Tutorial* (which only has an empty *main.c* file).

Now go into the *SDL2/bin* folder and copy the *SDL2.dll* file to where your *main.c* file is located (or *main.cpp* if you are writing in C++).


C:\Tutorial\SDL2\bin

Namn	Senast ändrad	Typ	Storlek
SDL2.dll	2019-07-25 06:36	Programtillägg	1 369 kB
sdl2-config	2019-07-25 06:35	Fil	2 kB

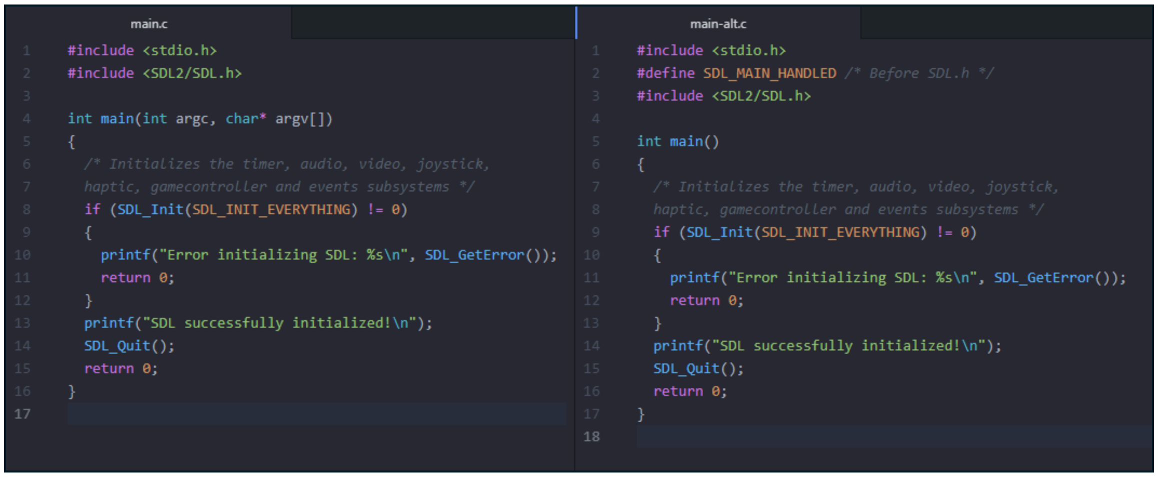
2 objekt


C:\Tutorial

Namn	Senast ändrad	Typ	Storlek
SDL2	2020-01-02 19:03	Filmapp	
main.c	2020-01-02 19:12	C-fil	1 kB
SDL2.dll	2019-07-25 06:36	Programtillägg	1 369 kB

Step 3: Creating a Basic C/C++ Program

We will now make a very simple C program that initializes SDL, and then terminates. There are two ways to do this, as illustrated below.


main.c

```c
#include <stdio.h>
#include <SDL2/SDL.h>

int main(int argc, char* argv[])
{
  /* Initializes the timer, audio, video, joystick,
  haptic, gamecontroller and events subsystems */
  if (SDL_Init(SDL_INIT_EVERYTHING) != 0)
  {
    printf("Error initializing SDL: %s\n", SDL_GetError());
    return 0;
  }
  printf("SDL successfully initialized!\n");
  SDL_Quit();
  return 0;
}
```

main-alt.c

```c
#include <stdio.h>
#define SDL_MAIN_HANDLED /* Before SDL.h */
#include <SDL2/SDL.h>

int main()
{
  /* Initializes the timer, audio, video, joystick,
  haptic, gamecontroller and events subsystems */
  if (SDL_Init(SDL_INIT_EVERYTHING) != 0)
  {
    printf("Error initializing SDL: %s\n", SDL_GetError());
    return 0;
  }
  printf("SDL successfully initialized!\n");
  SDL_Quit();
  return 0;
}
```

The method on the left is the recommended way, and is what we will use. Because of the way SDL works, the main method should be written as:

```
int main(int argc, char* argv[])
{
...
}
```

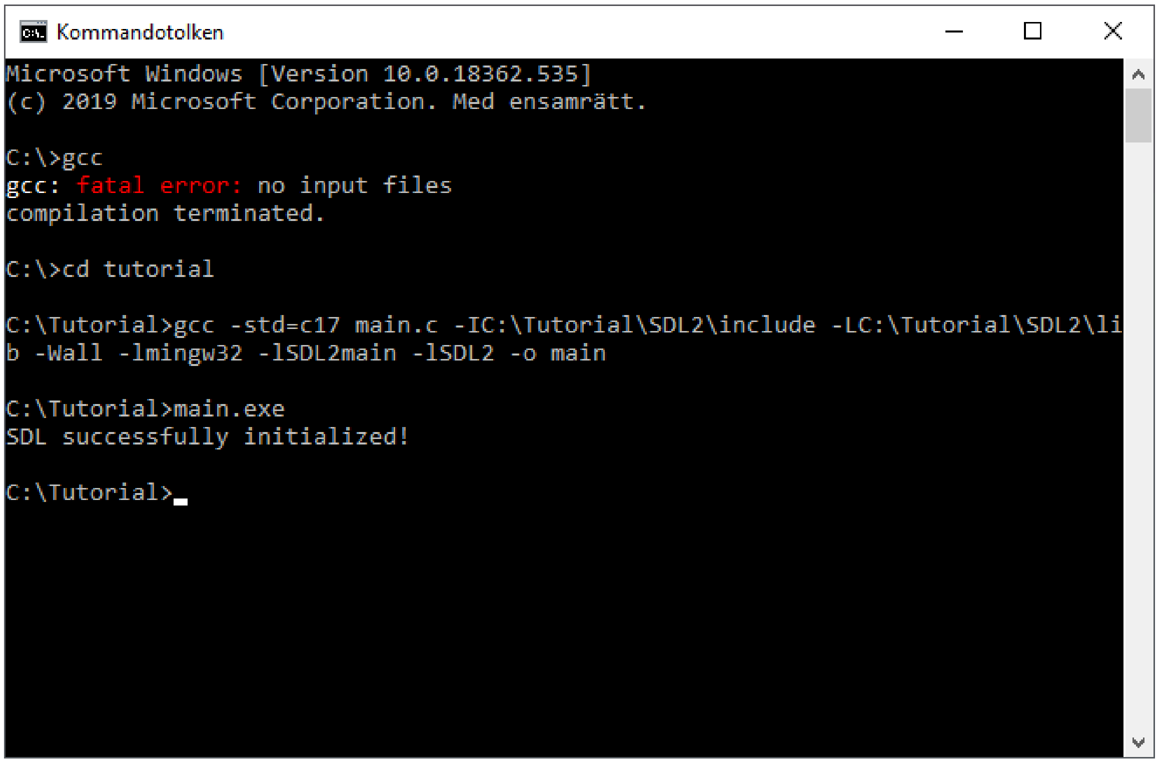
If it's not on this form, we have to define the macro *SDL_MAIN_HANDLED* before including the *SDL.h* header.

gcc -std=c17 main.c -I{Path to SDL2\include} -L{Path to SDL2\lib} -Wall -lmingw32 -lSDL2main -lSDL2 -o main

or if we are writing a C++ program:

g++ -std=c17++ main.cpp -I{Path to SDL2\include} -L{Path to SDL2\lib} -Wall -lmingw32 -lSDL2main -lSDL2 -o main

This will create a main.exe file in the project directory.



```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. Med ensamrätt.

C:\>gcc
gcc: fatal error: no input files
compilation terminated.

C:\>cd tutorial

C:\Tutorial>gcc -std=c17 main.c -IC:\Tutorial\SDL2\include -LC:\Tutorial\SDL2\li
b -Wall -lmingw32 -lSDL2main -lSDL2 -o main

C:\Tutorial>main.exe
SDL successfully initialized!

C:\Tutorial>_
```

As we can see everything works. Now some explanation about the flags.
-std=c17 means that the compiler uses the most recent C standard, ISO/IEC 9899:2018, known as both C17 and C18. Because of this, the flag -std=c18 is equivalent. Worth noting is that C17 was pretty much a bugfix version of C11, and the fixes are also applied to C11 in GCC - so the only difference from using -std=c11 is the value of __STDC_VERSION__.
See also: GCC Language Standards for C

-I is for include and *-L* is for linking the library.

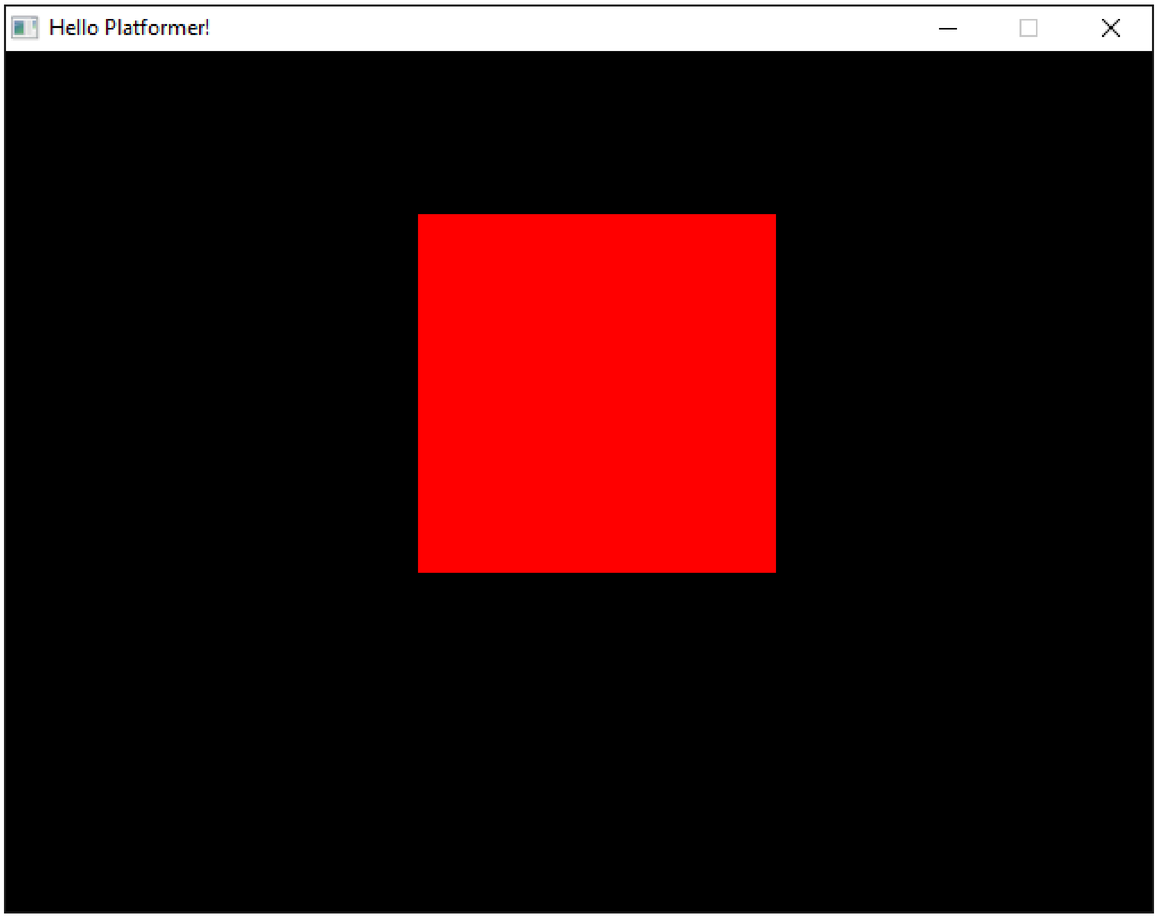
-Wall enables many compiler warning messages, it is not required but it is recommended. Also recommended is adding the -Wextra tag for more warnings, which we skip in this tutorial since we are

-lmingw32 is required, but don't get fooled by the name - we are still compiling a 64-bit program (which you can check by making sure that the value of *(8 * sizeof(void *))* is 64).

-lSDL2main and *-lSDL2* are also required.

Step 4: A Platformer in C

Now we are done with the setup and can therefore start using SDL2 for development in C/C++, so I will include some example code to get a basic object moving on the screen. Here is platformer.c!



For more Game development in C:
Writing 2D Games in C using SDL by Thomas Lively

contact@matsson.com