Compiling C modules

Table of Contents

modules in C

C programming projects in industry have more than one source file because it usually takes more than one programmer to complete the job. However, only one of those C source files in an application contains `int main() { }`. Other C source files are known conceptually as modules. A module's source code contains functions() which work independently or together with other modules. A "main" program calls those functions.

Source files making up an application are grouped together in a Visual Studio IDE Project or in the same folder/workspace when using Visual Studio Code or other development tools including command line compilation.

A typical C application has .h header files, .c module files, and a single main.c source file which calls functions in the modules.

TL;DR for macOS

- Visual Studio Code or Xcode are good choices for C development.

TL;DR for Windows

- Visual Studio IDE is the professional's choice for C development on Windows.

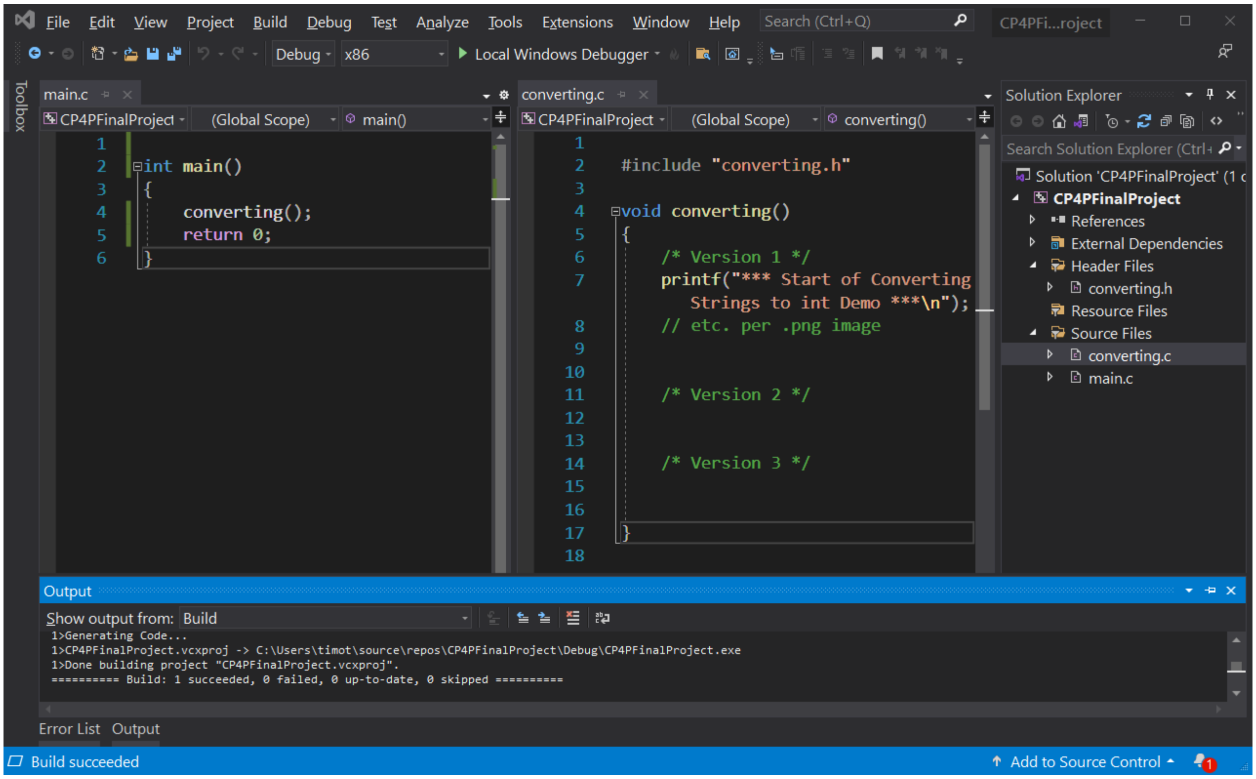
TL;DR for command line compilation

- Microsoft Windows cl compiler: https://youtu.be/rqLbyj0TnIg (see notes below)
- macOS gcc compiler: included with Visual Studio Code or Xcode. See https://youtu.be/we2Oc4WQ7FM
- The gcc compiler is native to the Unix / Linux world. If you are going all hardcore in Windows 10+, do it in the Windows Subsystem for Linux where gcc is very happy.
- You can install gcc to run under Windows. You can also walk across Canada. In both cases, there are easier ways to get there. If you must, see the MinGW-w64_WinLibs.docx file.

helloWorld C source to test a compilation

```
/*
helloWorld : the canonical test of any programming language thanks to K&R.
*/
#include <stdio.h>     // Standard Input/Output
int main(void)         // mainline – only one in an application
{
   // console output as proof of compiler installation and operation

   // call a function in the standard input/output library.
   printf("Hello, World!\nThis is a compiler test.\n");

   return 0;
}
```

project C source template example



main.c:

```c

int main()
{
    converting();
    return 0;
}
```

converting.c:

```c

 #include "converting.h"

void converting()
{
    /* Version 1 */
    printf("*** Start of Converting
       Strings to int Demo ***\n");
    // etc. per .png image


    /* Version 2 */


    /* Version 3 */


}

```

Show output from: Build

```
1>Generating Code...
1>CP4PFinalProject.vcxproj -> C:\Users\timot\source\repos\CP4PFinalProject\Debug\CP4PFinalProject.exe
1>Done building project "CP4PFinalProject.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Your project or workspace/folder/directory contains three files:

- *moduleName*.h header file
- *moduleName*.c function file
- main.c with `int main() {` which calls the function inside *moduleName*.c }

Command line compilation of Final Project source files

To compile your module for unit testing on

Microsoft, use cl — *see next page*

macOS or Linux, use gcc

```
> gcc  moduleName.c  main.c  -o main
```

e.g.

```
> gcc  converting.c  main.c  -o main
```

To compile all modules into a program for Integration testing,
specify all the module names:

```
> gcc moduleA.c moduleB.c moduleC.c moduleD.c main.c -o main
```

e.g.

```
> gcc fundamentals.c manipulating.c converting.c tokenizing.c  main.c -o main
```

To compile a module only and make it runnable: (works only in gcc)

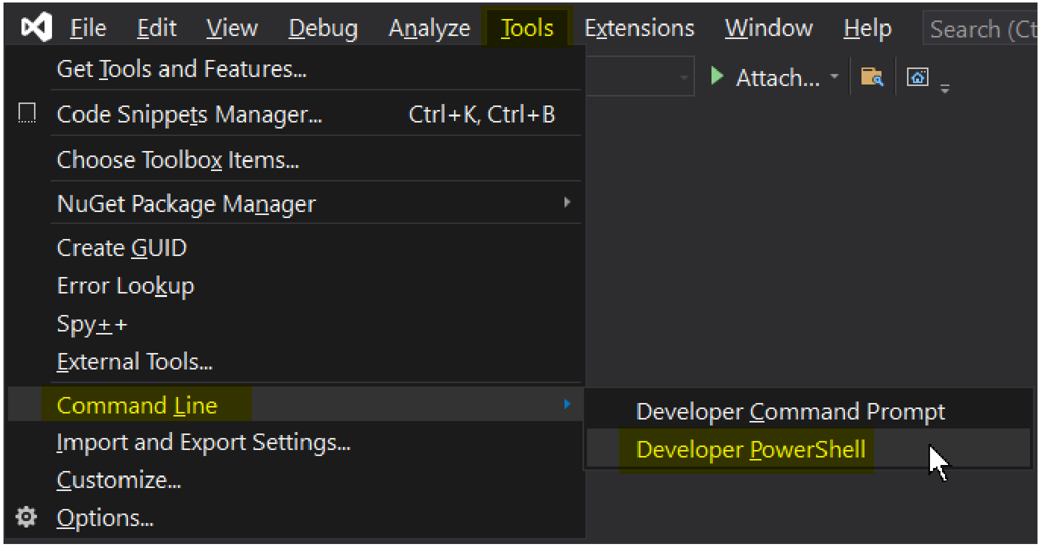
```
   > gcc -nostartfiles     moduleA.c  -o module
```

e.g.

```
gcc -nostartfiles  converting.c  -o converting
```

running the Microsoft cl compiler

The `cl` compiler runs only from a Visual Studio developer command prompt.
VS-IDE or VS Code > menu > View > Terminal [Ctrl + `] shows the terminal.
Alternatively, access through Visual Studio IDE:

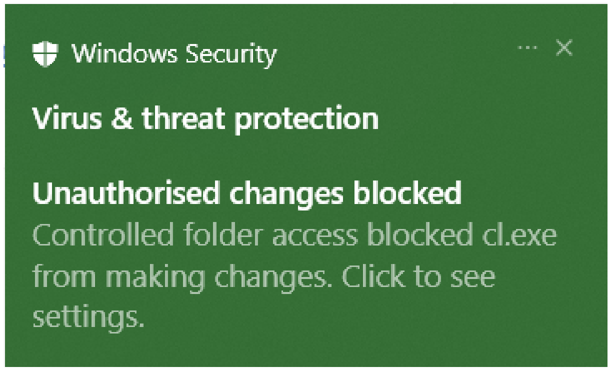


```
**********************************************************************
** Visual Studio 2022 Developer PowerShell v17.3.5
**********************************************************************
PS C:\Users\me\source\repos\CP4PFinalProject>

   cd "C:\Users\me\Documents\Seneca\CPR101\Final" –> as required

PS C:\Users\me\ Documents\Seneca\CPR101\Final>
   cl .\moduleName.c  .\main.c  /link  /out:main.exe
   cl .\converting.c  .\main.c  /link  /out:main.exe
   cl .\converting.c ### source requires main() to call function()

Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30136 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

converting.c
main.c
Generating Code...
Microsoft (R) Incremental Linker Version 14.29.30136.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:converting.exe
/out:main.exe
converting.obj
```

```
main.obj
PS C:\Users\timot\source\repos\CP4PFinalProject> .\main.exe
*** Start of Converting Strings to int Demo ***
...
```

Windows Security

One time only:

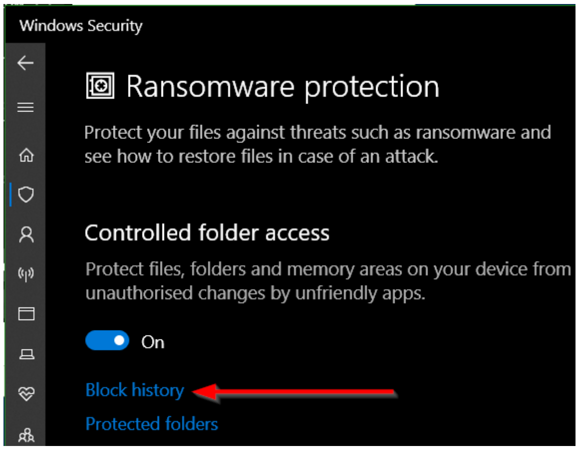


Windows Security

Virus & threat protection

Unauthorised changes blocked

Controlled folder access blocked cl.exe from making changes. Click to see settings.



Windows Security

Ransomware protection

Protect your files against threats such as ransomware and see how to restore files in case of an attack.

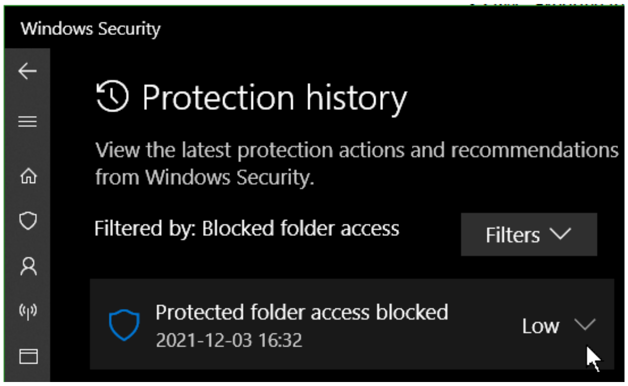
Controlled folder access

Protect files, folders and memory areas on your device from unauthorised changes by unfriendly apps.

On

Block history

Protected folders



Windows Security

Protection history

View the latest protection actions and recommendations from Windows Security.

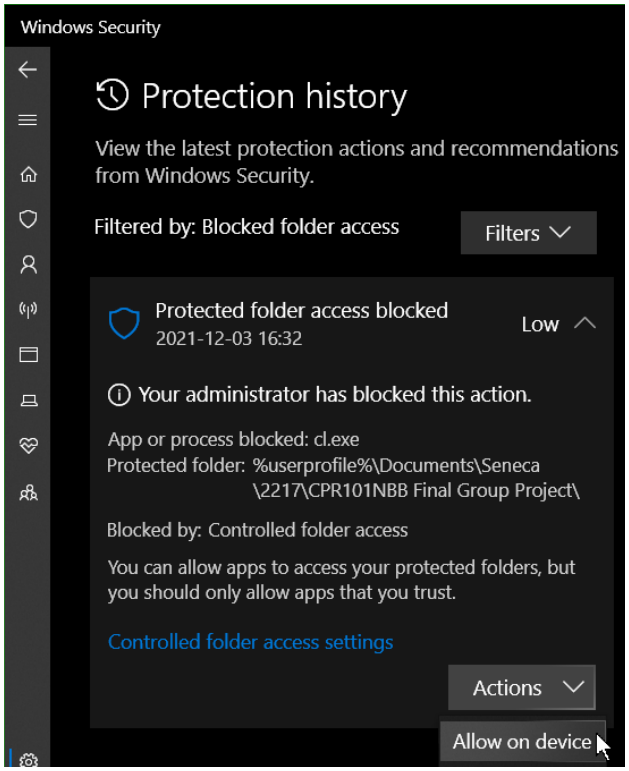
Filtered by: Blocked folder access

Filters

Protected folder access blocked

2021-12-03 16:32

Low



Windows Security

Protection history

View the latest protection actions and recommendations from Windows Security.

Filtered by: Blocked folder access

Filters

Protected folder access blocked

2021-12-03 16:32

Low

Your administrator has blocked this action.

App or process blocked: cl.exe

Protected folder: %userprofile%\Documents\Seneca\2217\CPR101NBB Final Group Project\

Blocked by: Controlled folder access

You can allow apps to access your protected folders, but you should only allow apps that you trust.

Controlled folder access settings

Actions

Allow on device