Beginning C++ Compilers

An Introductory Guide to Microsoft C/C++ and MinGW Compilers

—

Berik I. Tuleuov
Ademi B. Ospanova

Apress®

Beginning C++ Compilers

Berik I. Tuleuov • Ademi B. Ospanova

Beginning C++ Compilers

An Introductory Guide to Microsoft C/C++ and MinGW Compilers

Apress®

Berik I. Tuleuov
Nur-Sultan, Akmolinskaia, Kazakhstan

Ademi B. Ospanova
Nur-Sułtan, Kazakhstan

ISBN-13 (pbk): 978-1-4842-9562-5
ISBN-13 (electronic): 978-1-4842-9563-2
https://doi.org/10.1007/978-1-4842-9563-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Berik dedicates the book to the memory of his father, Iglik K. Tuleuov. He expresses gratitude to his family for their endless patience while working on this book, colleagues and Apress editors for their kindness.

Contents

About the Authors

Berik I. Tuleuov is Senior Lecturer at L. N. Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan. He's a researcher and mathematician using computers for scientific computations and designing algorithms. He runs a topic on The AIFC Tech Hub (a meeting point for global startups, entrepreneurs, investors, industry's top experts and talent pool) about Microsoft C/C++ compilers. This forum has more than two million registered participants. He regularly takes part in academic and industry conferences, mainly on computer science topics. Interests include programming languages, algorithms and data structures, concurrent (parallel) programming, scientific programming, (La)TeX Typesetting System, and data visualization.

Ademi B. Ospanova is an Associate Professor in the Department of Information Security at L. N. Gumilyev Eurasian National University. She is the author of many courses in the field of IT technologies. She is developer of educational programmes of all levels of the university on information security. In the educational process and projects she uses her own software and libraries in C/C++, C#, Java, Prolog, R, Python, Solidity, works in Mathematica, Maple, Sage packages. She also has her own website, including hosting on her own server.

She manages grant and initiative research projects, and her Masters and PhD students are winners of national scientific competitions.

She also gives courses and consultations on cryptography and programming to specialists from various companies.

About the Technical Reviewer

Sedat Akleylek received the B.Sc. degree in mathematics majored in computer science from Ege University, Izmir, Turkey, in 2004, and the M.Sc. and Ph.D. degrees in cryptography from Middle East Technical University, Ankara, Turkey, in 2008 and 2010, respectively. He was a Postdoctoral Researcher at the Cryptography and Computer Algebra Group, TU Darmstadt, Germany, between 2014 and 2015. He was an Associate Professor at the Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey, between 2016 and 2022. He has been Professor at the Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey, in 2022. He has started to work at the Chair of Security and Theoretical Computer Science, University of Tartu, Tartu, Estonia since 2022. His research interests include the areas of post-quantum cryptography, algorithms and complexity, architectures for computations in finite fields, applied cryptography for cyber security, malware analysis, IoT security, and avionics cyber security. He is a member of the Editorial Board of IEEE Access, Turkish Journal of Electrical Engineering and Computer Sciences, Peerj Computer Science, and International Journal of Information Security Science.

Acknowledgments

Ademi would like to dedicate the book to her fragile but strong mother who cares for her entire family. She provides Ademi with warmth and the opportunity to do her job.

Introduction

Anyone who wants to start programming in the C/C++ languages needs two things in general: a computer and two programs called a C/C++ language compiler and a source code editor (generally speaking, the so-called debugger — a program that helps find errors in the source code, but still it is not necessary). More, in principle, nothing is needed.

If any text editor (for example, Notepad on Windows) is in principle suitable as a source code editor, then the situation with compilers is not simple. On Windows, they must be installed, and in the vast majority of cases, when it comes to installing the C/C++ compiler, for some reason it means installing Microsoft Visual Studio, which requires a lot of computer resources. Meanwhile, Microsoft Visual Studio is not a compiler, but the so-called Integrated Development Environment (IDE), which includes, among other components, also a C/C++ language compiler.

As far as we know, there are no books devoted to installing C/C++ compilers[1], it is implicitly assumed that the user has the compiler either already installed, or its installation is standard and does not cause difficulties. However, there are many pitfalls here, and we will try to briefly describe the motives that prompted us to write our book.

Under Windows, usually installing a C/C++ compilers, especially Microsoft ones, takes quite a lot of time, because it comes with Microsoft Visual Studio for the vast majority of users. Installing Visual Studio requires usually about 40 GB of disk space and big amount of RAM, so it is impossible to use weak hardware. So we suggest an easy way to deploy Microsoft C/C++ compiler: no headache with disk space and hardware resources lack. Additionally, our means saves big amount of time since one can deploy software on removable devices, such as flash sticks, and use it easily in a portable way. We achieve this by using Enterprise Windows Driver Kit (EWDK), a single, large ISO file, which can be mounted as virtual device and used directly without any installation. EWDK contains everything from Visual Studio except IDE. EWDK also allows to use MASM64 (Microsoft Macro-Assembly) and C# compilers. With the aid of MSBuild System one can compile Visual Studio Projects (.vcxproj) and Solutions (.sln) even without Visual Studio! Analogously, MinGW compilers can be deployed from 7z/zip archives, simply by

[1] There are a lot of books on the C/C++ languages themselves.

unpacking into appropriate location. Briefly, both Microsoft C/C++ and MinGW compilers can be used as portable software. Notice that such approach does not require an administrative privileges at all.

It is Create Once, Use Many principle: one can deploy these compilers and auxiliary software on removable device and use everywhere, or just copy it to hard disk and use them from local disk. There is no need to re-install.

Also, users can use several versions of these compilers at the same time, they do not interfere each other. Using MSYS (Minimal SYStem, a port of GNU Autotools) allows to build under Windows many libraries originally designed for Unix-systems. These things important because standard installation procedure doesn't give such a flexibility: very often various versions of installed software conflict with each other, or it is impossible to install at all.

Our book is intended primarily for two categories of users:

- beginners to learn the C/C++ language, who don't want to spend time on the standard installation of MinGW and Microsoft C/C++ compilers, since in the first case one has to make a difficult and non-obvious choice between different builds of this compiler, and in the second — to solve computer resources lack and installation problems;
- advanced users who, generally speaking, are not professional programmers, but write small programs in standard C/C++, for, for example, scientific and technical calculations.

Of course, the approach we propose can also be useful for professional programmers, as it saves a lot of time; in addition, the created toolkit can be used in the future on a variety of computers without special preparation, since this toolkit is portable.

We describe the MinGW and Microsoft C/C++ compilers, the clang compiler is not covered (its popularity has not yet reached the level of first two). We set as our main goal the description of compilers, and not various IDEs, so they are practically not considered. Various advanced (lightweight) editors that have some IDE functions are considered as code editors. We do not consider the currently popular Visual Studio Code due to the fact that it is not lightweight (as we have already noted, the IDE is an auxiliary tool for us; however, if the user has enough memory installed on the computer, then nothing prevents using VS Code). We do not consider code debuggers, an interested reader can later deal with this topic himself.

Also, we do not consider WSL (Windows Subsystem for Linux), since this subsystem is designed to run Linux applications (and not all of them: for example, restrictions apply to GUI applications) under Windows, and we are fighting to build Open Source applications and libraries originally created for Unix systems for Windows. WSL, although it provides less resource consumption compared to virtualization, is still an additional layer that negatively affects the performance of applications running under it. The programs and libraries compiled under Windows using the tools we describe are native Windows applications and thus provide the highest performance.

For completeness of our research, we use a variety of versions of Windows, on different computers and virtual machines (GNOME Boxes, VMware). The system configurations on these platforms are different (different versions of Windows, different amounts of disk space, etc.), which explains why throughout the book the Programs, Soft, and User directories are located on various partitions of the computer's hard drive (and sometimes on removable devices). In part, we did this intentionally, because we wanted, in accordance with the spirit of our book, in every possible way to emphasize the flexibility and portability of the approach we use.

For the convenience of the reader, in the Appendixes we provide a number of tables to facilitate the use of the Microsoft C/C++ compiler. These tables are taken from the Microsoft website.

Files and Devices

1

General information on files and devices is given in this chapter. The concept of a file is fundamental to computer science, so it's important to be clear about it. The issues of effective user interaction with the computer are also discussed.

1.1 File Types and Formats

In general, a file is a piece of information with an assigned name that is stored on a computer media. The media could be a hard or SSD disk, compact or DVD disk, magnetic tape, flash sticks and cards, etc.

Most computers have at least one disk drive, HDD or SSD, installed permanently. Nowadays, laptops have no DVD drives at all. Almost all devices can be connected to a computer via USB; in this case, a user can use removable media: flash sticks and cards, external HDD/SSD disks, or CD/DVD media.

By type, computer files are divided into text and binary files. Text files contain bytes that have a visual representation, as well as bytes that serve to control (line feed, carriage return, tabulation). Simply put, the contents of a text file can be "read."

Microsoft Word files, although containing text information for the most part, are not text files, as they contain information about formatting, sizes and types of fonts used, and other meta-information, as well as other data, such as pictures. Generally, binary files can contain human-readable pieces too.

By format, files are divided into graphics, multimedia (video and audio), executables (programs), objects, archives, CD/DVD images, etc. On Windows, file formats are recognized by their so-called extension: three or more letters after the last period in the file name. For example, Adobe Acrobat Portable Document Format (PDF) files have the extension `'.pdf'`, and Microsoft Office documents have extensions `'.docx'`, `'.xlsx'`, etc. For many programs, file extensions are

B. I. Tuleuov, A. B. Ospanova, *Beginning C++ Compilers*,
https://doi.org/10.1007/978-1-4842-9563-2_1

not mandatory: for example, text editors can open the text file named “Readme”; however, some software require strong extensions for their files.

Most file formats are designed to be independent of processor platforms and operating systems. This means, for example, that the same pdf document can be opened on Windows, Linux, Mac OS, Android devices, and finally iOS. However, this is not the case for binary executables, which we will discuss as follows.

Note All files consist of bytes, both text and binary.

1.2 Executable and Batch Files

Before moving on, we need to make some clarifications. Most computer users, are dealing with a modern and convenient graphical shell that provides ease of work. The graphical shell of Windows is Explorer. The shell is an intermediary between the user and the operating system, allowing you to open files, run programs, etc. In the graphical shell, the mouse plays an important role, with which you can click the icons of programs and documents to launch and open them.

Note Mouse actions can be duplicated via the keyboard.

However, the graphical shell is not the only one—there is another one, which is called the command line. In the command line, the main role is played by another device—the keyboard. The command line is mainly used to run commands by typing the name of the command in the prompt and pressing the Enter key. The command can be either a system command (e.g., `set` or `echo`) or an executable program file, and you must specify the exact location (full path to this file) of this file. If such a file is not found due to nonexistence or an incorrect full path to it, the system will display a corresponding message:

```
E:\Test>mycommand
'mycommand' is not recognized as an internal or external command,
operable program or batch file.

E:\Test>
```

Commands can have parameters called options. Their number depends on the purpose of the command. As a rule, commands have built-in help, which in Windows is called like this:

```
E:\Test>echo /?
Displays messages, or turns command-echoing on or off.

  ECHO [ON | OFF]
  ECHO [message]
```

```
Type ECHO without parameters to display the current echo setting.

E:\Test>
```

Conventional notation `[option]` means that the `'option'` parameter in square brackets `[ ]` can be omitted (and the brackets are omitted), and the `'|'` in `ON | OFF` means that one of the two options must be selected (sometimes, it happens that options can be combined, or, conversely, some options may be incompatible with some other options; such cases are specifically discussed in the help of the command).

For open source utilities, help is usually called in the format

```
E:\Test>command --help
```

So how to find this command line? Very simple: Press the Win + R keys and type `cmd` (`cmd.exe` is also possible) and press the Enter key. Windows will bring up a command prompt window.

Note In earlier Windows versions, this file was called `COMMAND.COM`. In modern Windows versions, this file is called `CMD.EXE` and is located in the `C:\Windows\System32` directory.

Although the command line looks inconvenient, it undoubtedly has advantages over the graphical shell, especially when it comes to automating sequential actions, that is, for programming a certain sequence of operations. An example from the practice of one of the authors: It was required to extract all graphic files from a filled CD while maintaining the directory structure; there were many nested directories on the disk. In the graphical shell, this task is almost impossible to solve quickly, but on the command line, this task is solved in a couple of minutes, using the `xcopy` utility with the `/s` key (option).

Executable files contain program code that is executed by the central processor of a computer. Windows executable file formats include (by file extension) `'.exe'`, `'.com'`, `'.dll'`, `'.sys'`, `'.ocx'`, etc.

- **exe**: The main format of Windows executable program files.
- **com**: This extension belonged a long time ago to 16-bit MS-DOS programs that were small and could use a small amount of RAM. On purely 32-bit and 64-bit operating systems, such programs cannot run (they can only run in compatibility mode or in emulators). Currently, Microsoft has made a change where the format of binary executable files is determined not by the extension but by the content of the file, so that any `'.exe'` file can be renamed to a `'.com'` file without breaking functionality. This change was made for compatibility with older MS-DOS batch files that called older utilities with `'.com'` extensions, the newer versions of which are already larger than the limits of the old `'.com'` format. Here is a list of these util-

ities: `chcp.com`, `diskcomp.com`, `diskcopy.com`, `format.com`, `mode.com`, `more.com`, and `tree.com`.
- **dll**: Dynamic-link libraries; they can contain both reusable executable code and data. The vast majority of Windows code reside in such libraries.
- **sys**: Windows device drivers; these files are binaries, and they are created by compilers from their human-readable source code in high-level programming languages such as C/C++ and Assembly (parts of drivers).

Windows `'.exe'` files have so-called “magic bytes”: their first two bytes are always `'MZ'`.

Batch files on Windows have extensions `'.bat'` and `'.cmd'`.

Executable files in `'.exe'` and `'.com'` formats as well as batch files (`'.bat'`, `'.cmd'`) can be directly launched by the user. This can be done both in the graphical shell and on the command line. In the graphical shell, open the Explorer window, find the file to be launched, and double-click it. For the second method, launch the command prompt, type the full file name, and press the Enter key.

Notice that for binary executable files and batch files, when they are executed, their extensions can be omitted on the command line. In this regard, an interesting question arises: If the files `test.bat`, `test.cmd`, `test.com`, and `test.exe` are in the same directory, then which one will be executed when the `test` command is executed?

Answer: The order of execution is `'.com'`, `'.exe'`, `'.bat'`, `'.cmd'`; hence, `test.com` will be executed. To execute any other of them, you need to write its name in full with the extension, for example, `test.exe`. In general, the execution priority is determined by the `PATHEXT` environment variable and can be changed, which we will talk about later.

A batch file is a text file, each line of which consists of a single command that can have parameters. Thus, a batch file can execute several commands sequentially one after another, that is, by running one command, we actually execute a whole series of commands! Therefore, such files are sometimes called script files.

It is important to note that binary executables are not only operating system dependent but also processor architecture dependent: for example, Windows `'.exe'` files do not run on Linux or Mac OS; moreover, `'.exe'` files created for 64-bit Windows do not work in 32-bit Windows (the opposite is true: 32-bit Windows `'.exe'` files work in 64-bit Windows in compatibility mode). Likewise, Linux binaries don’t work on Windows.

Batch files are a bit more flexible in this regard: while Windows `'.bat'` and `'.cmd'` files don’t work on Linux, you can create script files on Windows that are cross-platform with some limitations. We will cover this in later chapters.

Note In Unix systems, every file can be made executable in terms of those systems.

1.3 System Commands

System commands are for executing common basic system commands. System commands are divided into internal and external ones. Internal commands are implemented in the `CMD.EXE` file; external commands are implemented as separate utilities located in the `C:\WINDOWS\System32` system directory. An example of an external command is the abovementioned `xcopy` (`xcopy.exe`), an advanced file and directory copying utility.
Help on these commands, as noted earlier, can be called with the `/?`.

We briefly describe here some useful commands:

cls	clears the console window (clear screen).
set	sets a value to an environment variable.
echo	echoes its argument value.
cd	changes the working directory (change directory).
dir	types the content of the current directory.
path	displays the value of the `PATH` environment variable.

`echo %PATH%` types the value of the `PATH` environment variable.

1.4 Mounting Devices

Sometimes, in Windows you have to change the letter of the CD/DVD drive or even the hard disk partition. This can be done through the Computer Management applet, which is invoked by right-clicking the `Computer` icon in the `Desktop`. Note that this requires administrator rights.

Much more interesting and useful is the `subst` command, which does not require administrator rights and allows you to mount a folder as a disk partition, assigning a given letter to this disk:

```
E:\Test>subst /?
Associates a path with a drive letter.

SUBST [drive1: [drive2:]path]
SUBST drive1: /D

 drive1:        Specifies a virtual drive to which you want to
  ↪  assign a path.
 [drive2:]path    Specifies a physical drive and path you want to
  ↪  assign
to a virtual drive.
 /D             Deletes a substituted (virtual) drive.

Type SUBST with no parameters to display a list of current
 ↪  virtual drives.

E:\Test>
```

For example, the command

```
E:\Test>subst X: E:\Test
```

will create disk `X:` in the system (if it, of course, did not exist) and mount all the contents of the `E:\Test` directory on this disk. We will use this command later when working with the Microsoft C/C++ compiler.

1.5 Virtual Devices

Some software is shipped in the ISO format. Examples are Linux distributions, Microsoft Enterprise Windows Driver Kit, and others. As we know, the ISO file is an image of CD/DVD media, and in the old days, the user had to burn this file to a blank CD/DVD disk and insert the disk into the drive. In our days, it is much more easier to handle such files— it suffices to use the so-called virtual devices. The user just creates such a device and mounts the ISO file on the device.

We work with DVD virtual devices. Virtual devices are created programmatically; no physical device is needed. Such devices can easily be created on Windows 7 with the aid of several software: DAEMON Tools Lite (www.daemon-tools.cc/products/dtLite), Alcohol 120% Free Edition (http://trial.alcohol-soft.com/en/downloadtrial.php, all Windows operating systems except 98/ME, for personal use only), etc. These tools require installation and system reboot.

In our opinion, the best program of this kind is WinCDEmu, an open source CD/DVD/BD emulator (https://wincdemu.sysprogs.org); this program has a portable version (https://wincdemu.sysprogs.org/portable/). Portable single executable file runs under all versions of Microsoft Windows (10/8.1/8/7/2008/Vista/2003/XP), on both x86 and x64 platforms. No system reboot is needed.

WinCDEmu is

- Free for any kind of use.
- Lite, about 670 KB only.
- Easy to use, just run the downloaded portable exe.

Of course, WinCDEmu requires administrator privilege to create virtual devices and mount ISO images.

On Windows 10 and up, no additional software of this kind is needed at all. Just right-click the ISO image on the Explorer window and select the “Mount” menu item. The system itself will create the device, assign it a letter, and mount the image there—no matter if the user has administrator privilege or not (no matter if the user has administrative rights or not).