Setting Up Your C Compiler on Windows

Microsoft Windows is one of the most popular desktop operating systems.

Before beginning, I highly recommend that you install **7-Zip** from `https://www.7-zip.org`/. 7-Zip will allow you to extract the various compression archive formats that library source code is distributed in.

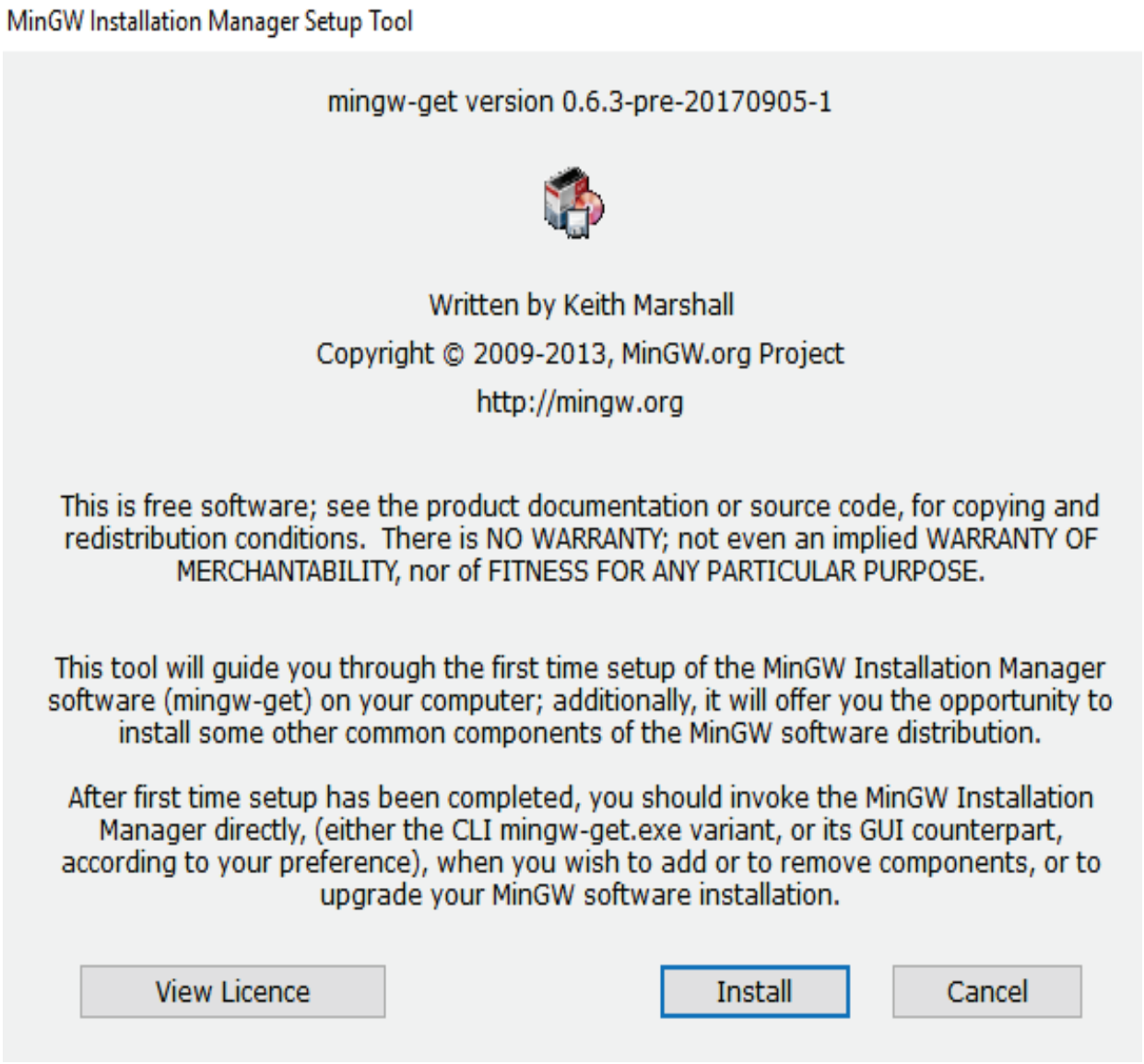
Let's continue and get MinGW, OpenSSL, and `libssh` set up on Windows 10.

Installing MinGW GCC

MinGW is a port of GCC to Windows. It is the compiler we recommend for this book.

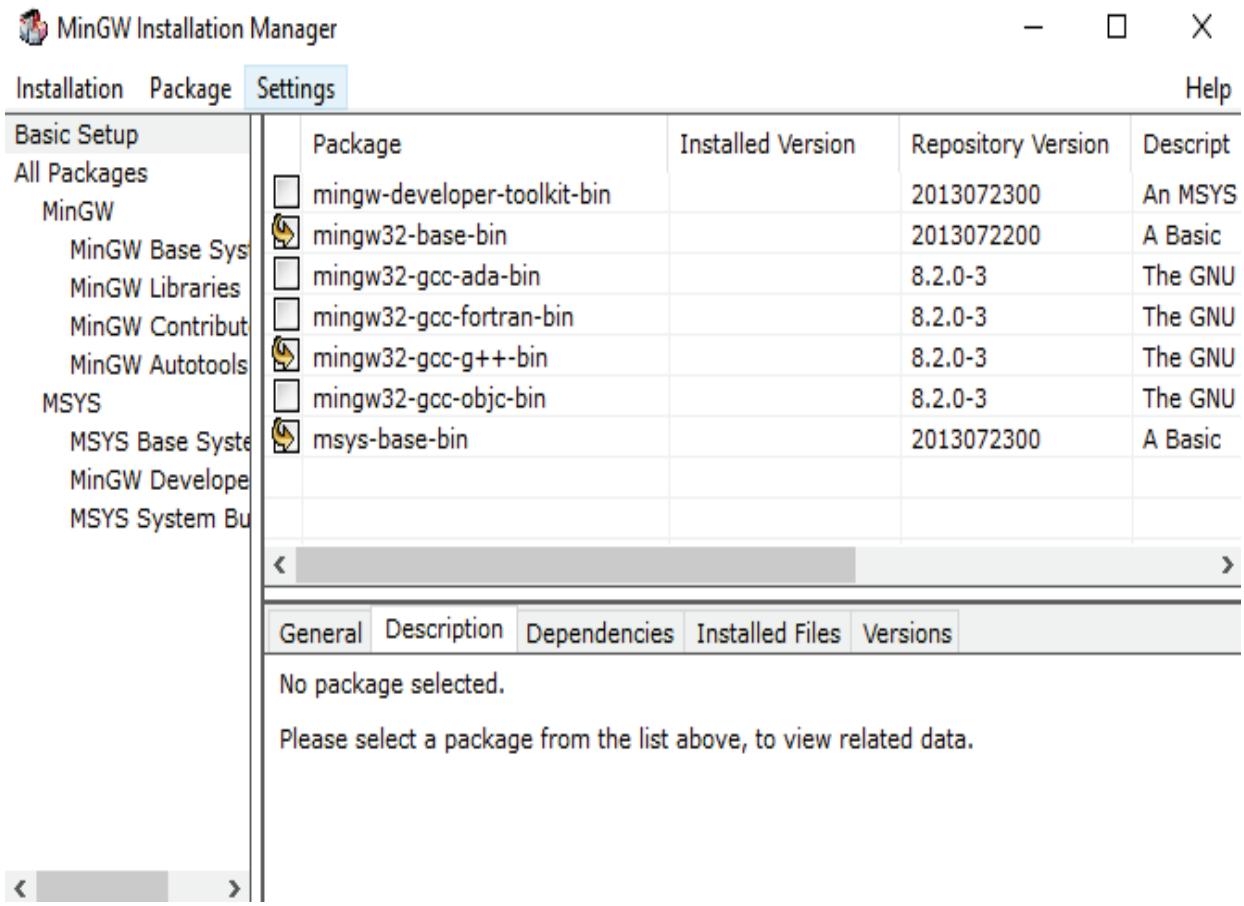
You can obtain MinGW from http://www.mingw.org/. Find the download link on that page and download and run the **MinGW Installation Manager** (**mingw-get**).

The MinGW Installation Manager is a GUI tool for installing MinGW. It's shown in the following screenshot:

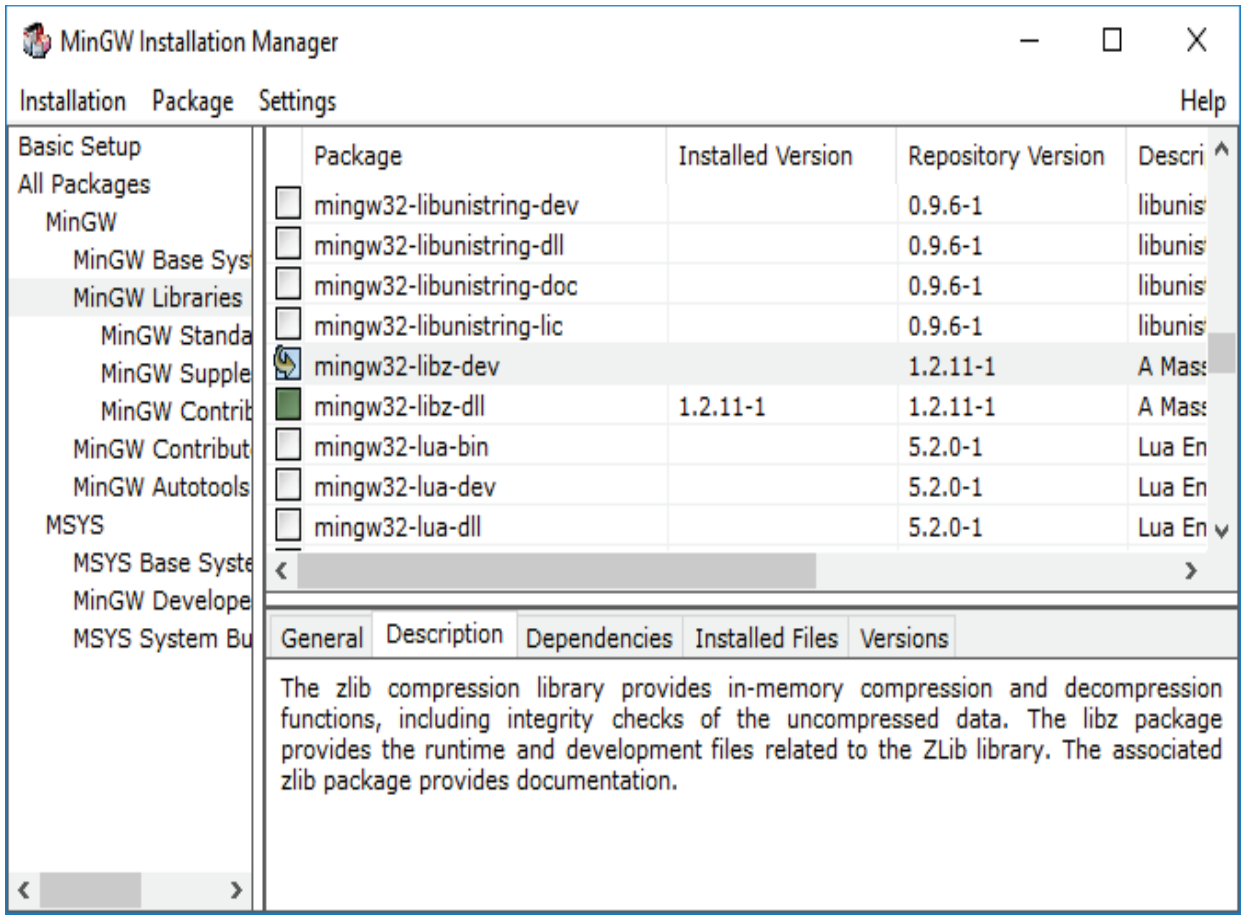


Click Install. Then, click Continue. Wait while some files download, and then click Continue once again.

At this point, the tool will give you a list of packages that you can install. You need to mark mingw32-base-bin, msys-base-bin, and mingw32-gcc-g++-bin for installation. This is shown in the following screenshot:



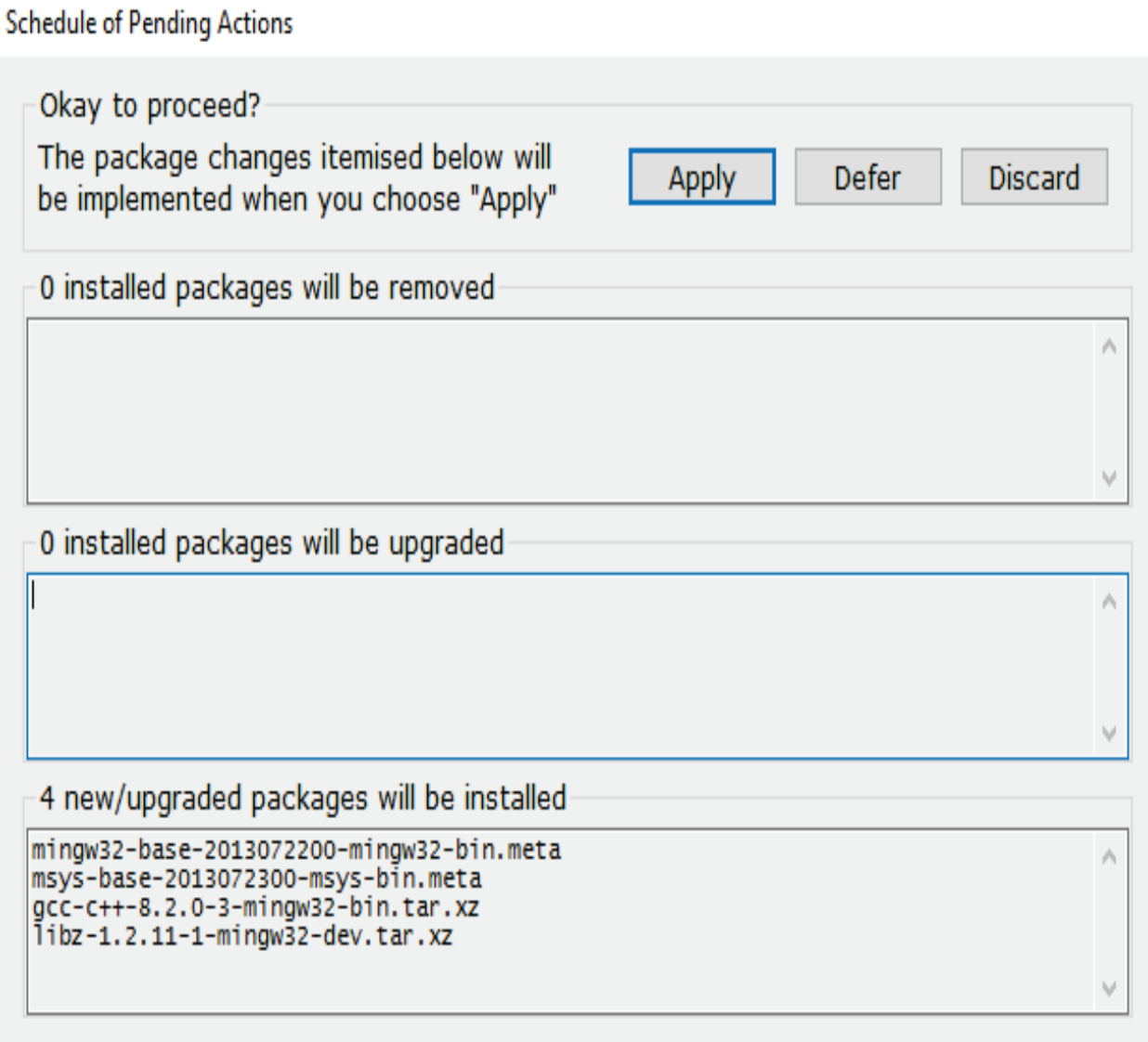
You will also want to select the mingw32-libz-dev package. It is listed under the MinGW Libraries section. The following screenshot shows this selection:



The `g++` and `libz` packages we've selected are required for building `libssh` later.

When you're ready to proceed, click Installation from the menu and select Apply Changes.

A new dialog will show the changes to be made. The following screenshot shows what this dialog may look like:

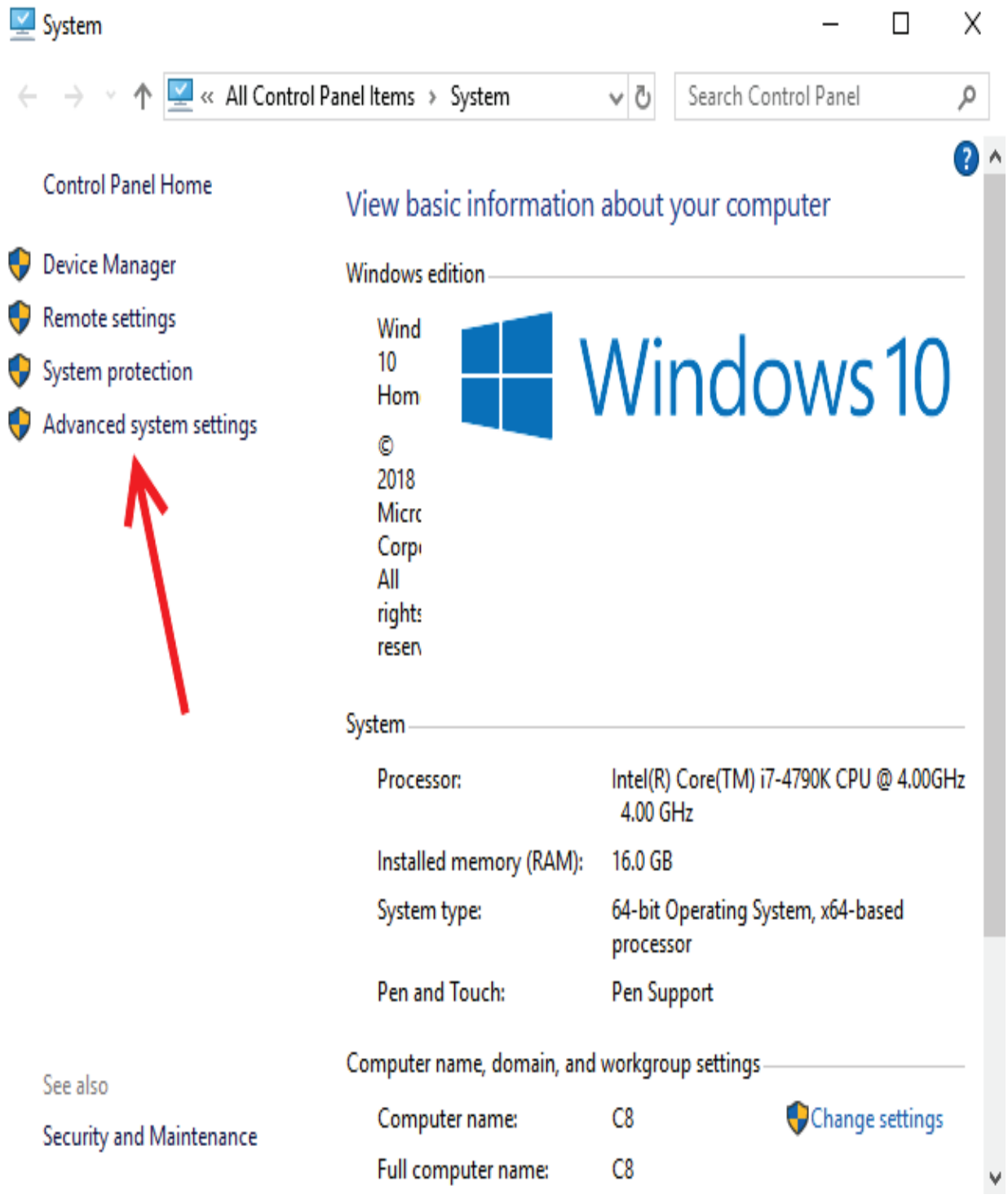


Click the Apply button to download and install the packages. Once the installation is complete, you can close the MinGW Installation Manager.

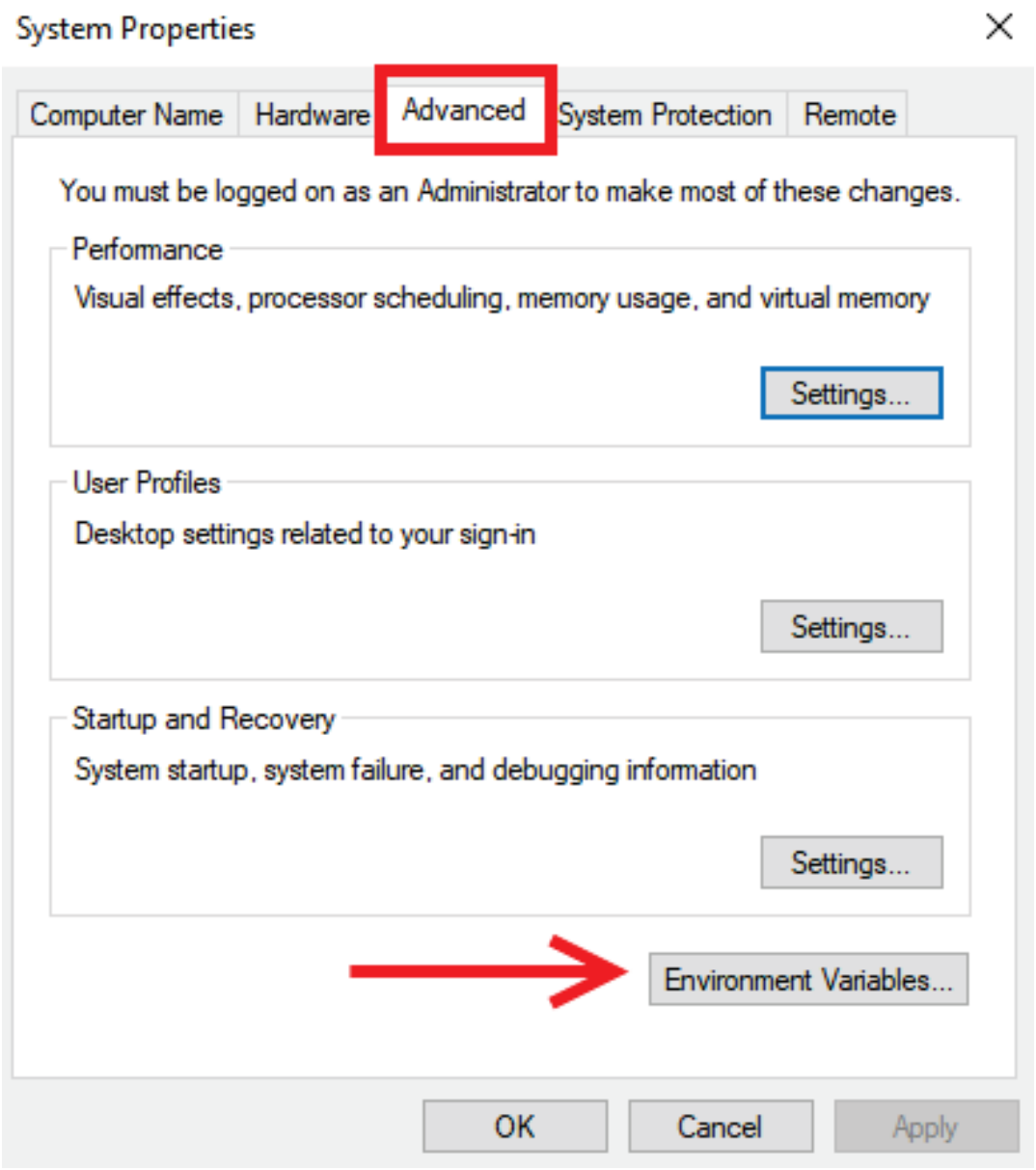
To be able to use MinGW from the command line easily, you will need to add MinGW to your PATH.

The steps for adding MinGW to your PATH are as follows:

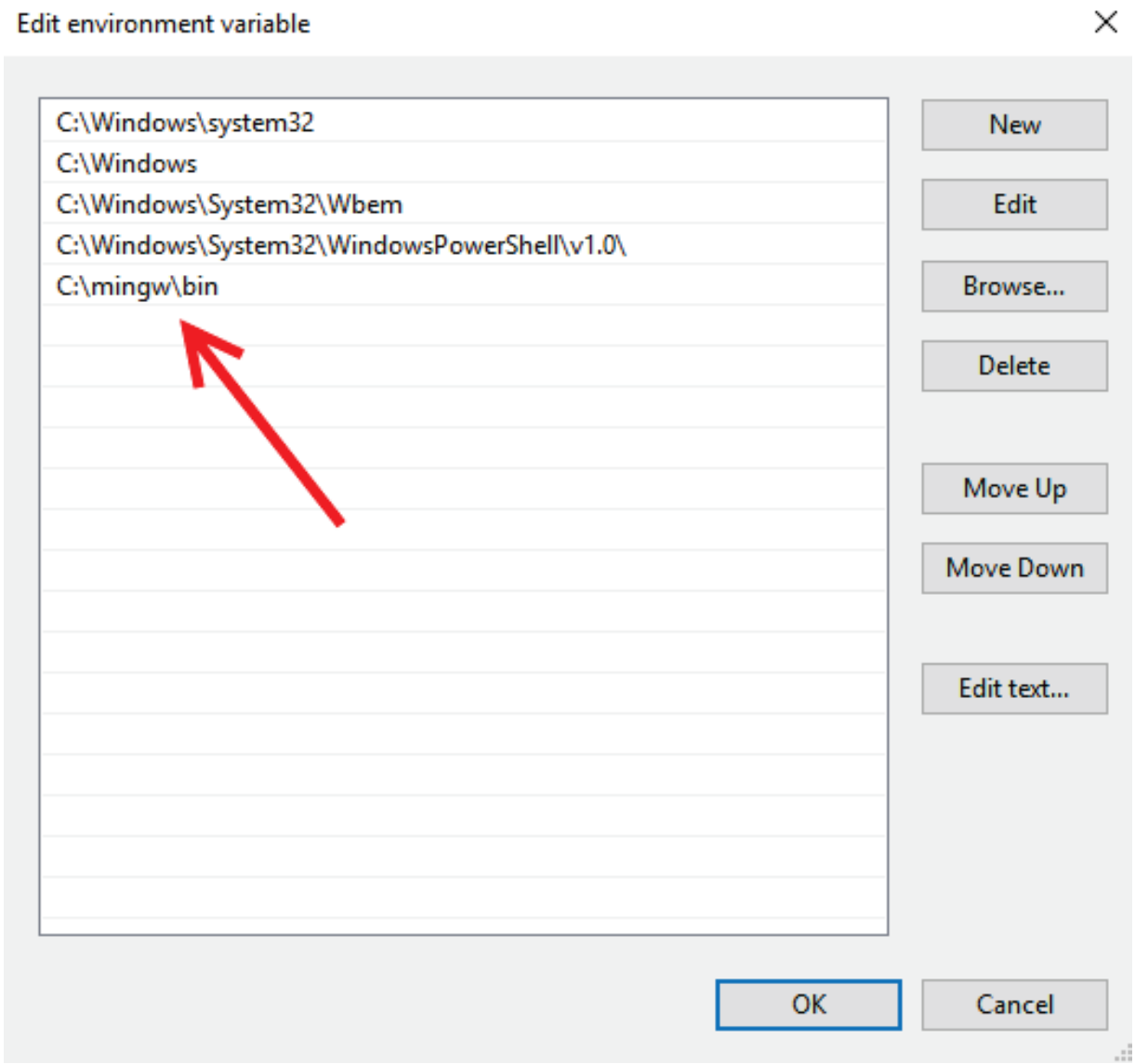
1. Open the System control panel (Windows key + *Pause*/*Break*).
2. Select Advanced system settings:



3. From the System Properties window, navigate to the Advanced tab and click the Environment Variables... button:

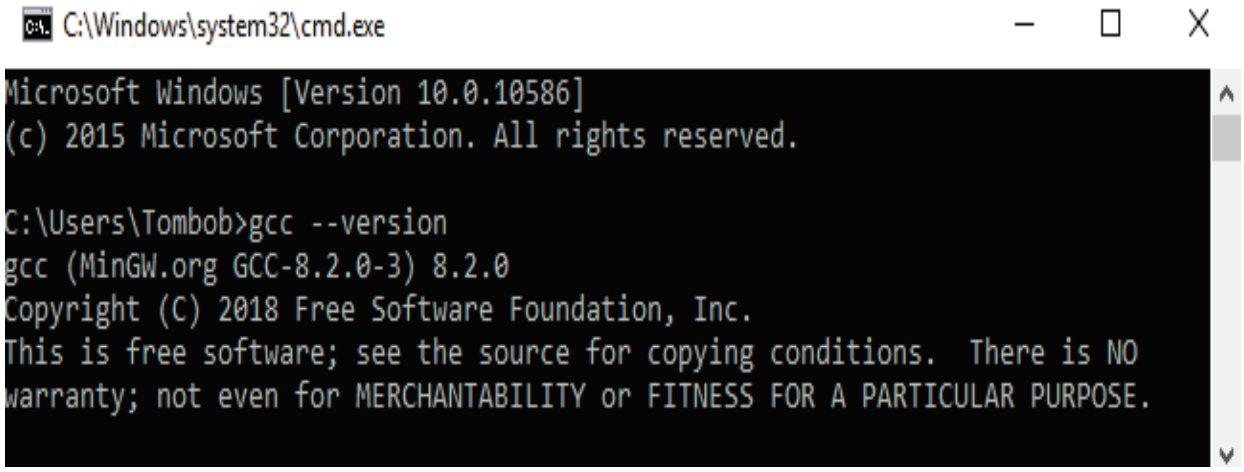


4. From this screen, find the `PATH` variable under System variables. Select it and press **Edit...**.
5. Click New and type in the MinGW path—`C:\mingw\bin`, as shown in the following screenshot:



6. Click OK to save your changes.

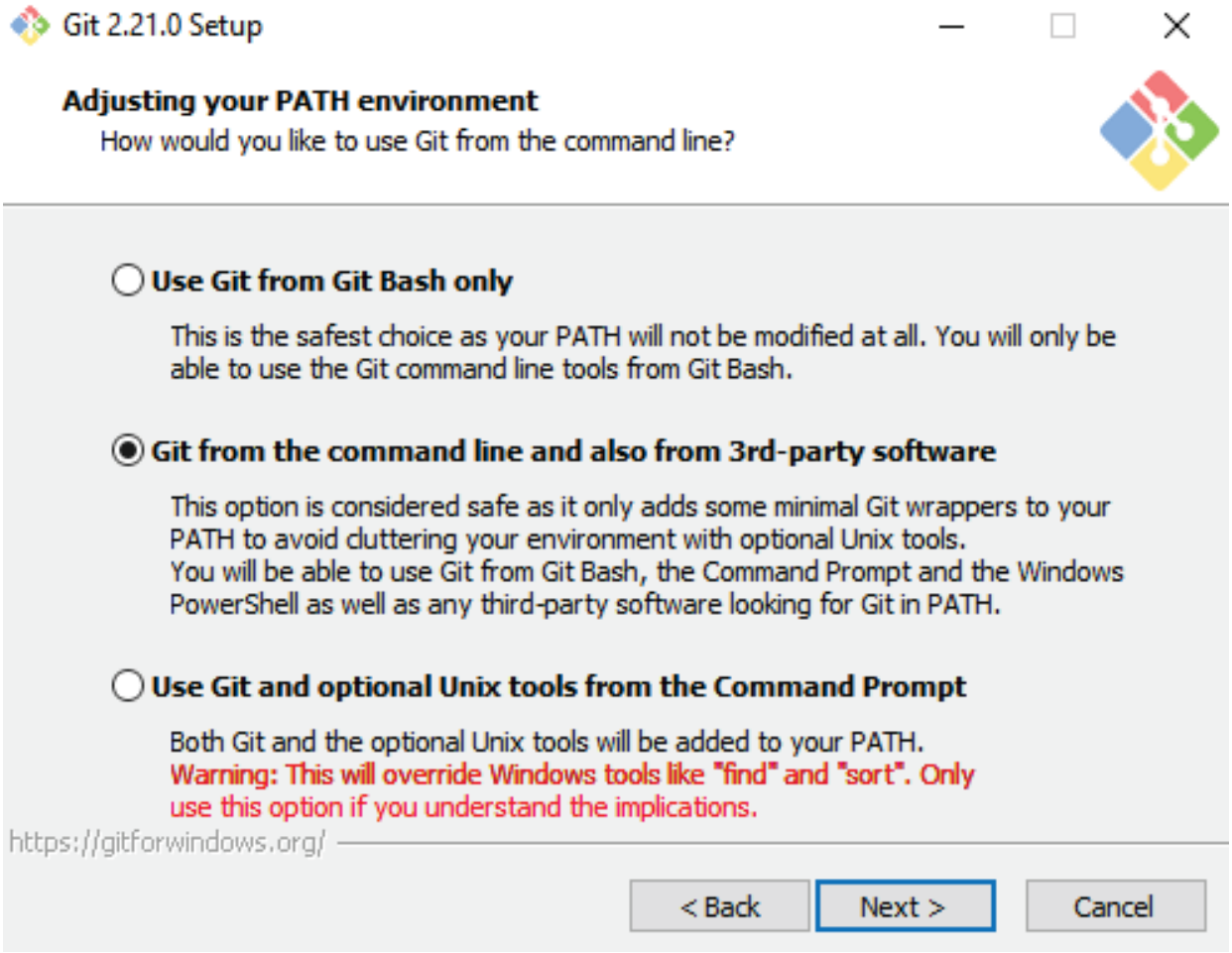
Once MinGW has been added to your `PATH`, you can open a new command window and enter `gcc --version` to ensure that `gcc` has been installed correctly. This is shown in the following screenshot:


C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\Tombob>gcc --version
gcc (MinGW.org GCC-8.2.0-3) 8.2.0
Copyright (C) 2018 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.

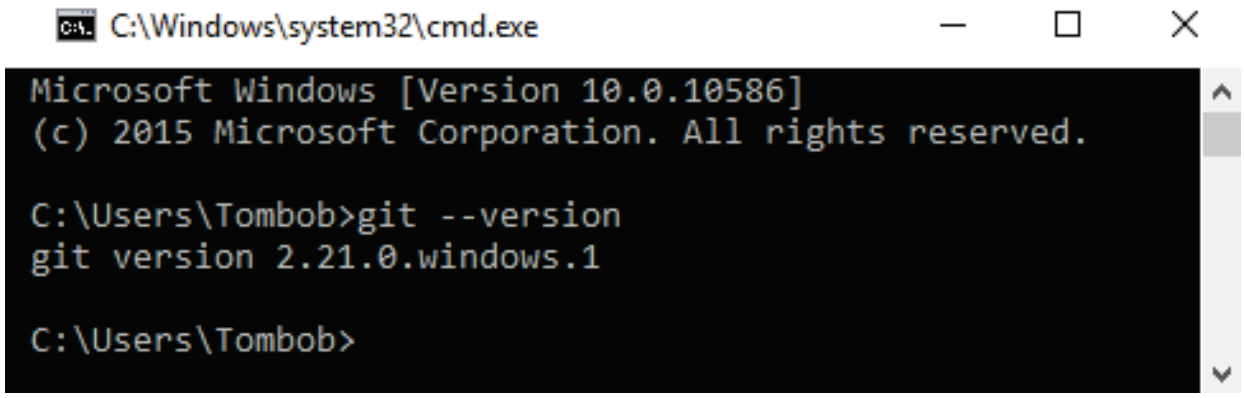
Installing Git

You will need to have the `git` version control software installed to download this book's code.

`git` is available from `https://git-scm.com/download`. A handy GUI-based installer is provided, and you shouldn't have any issues getting it working. When installing, be sure to check the option for adding `git` to your `PATH`. This is shown in the following screenshot:



After `git` has finished installing, you can test it by opening a new command window and entering `git --version`:


C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\Tombob>git --version
git version 2.21.0.windows.1
C:\Users\Tombob>

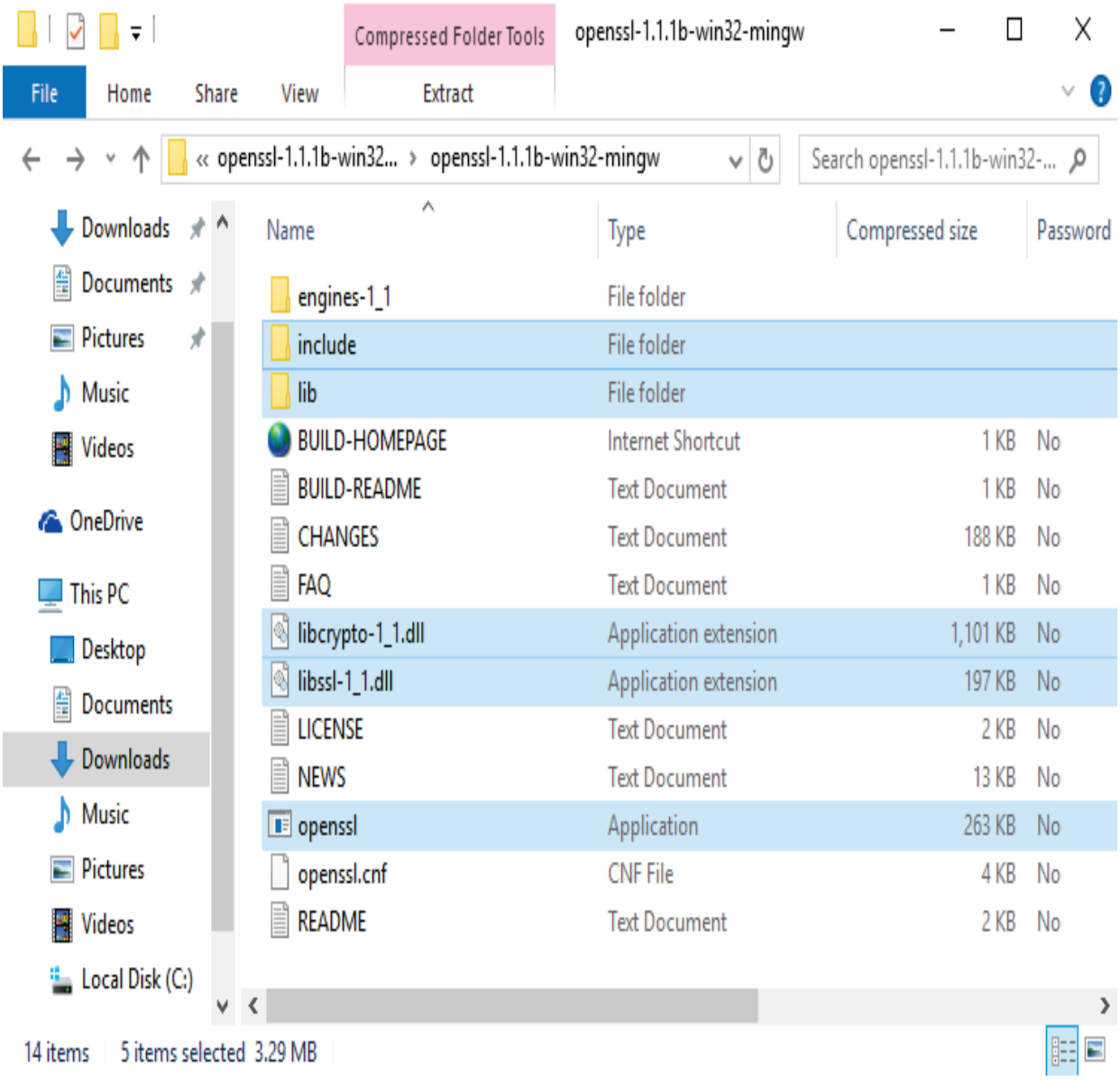
Installing OpenSSL

The OpenSSL library can be tricky to get going on Windows.

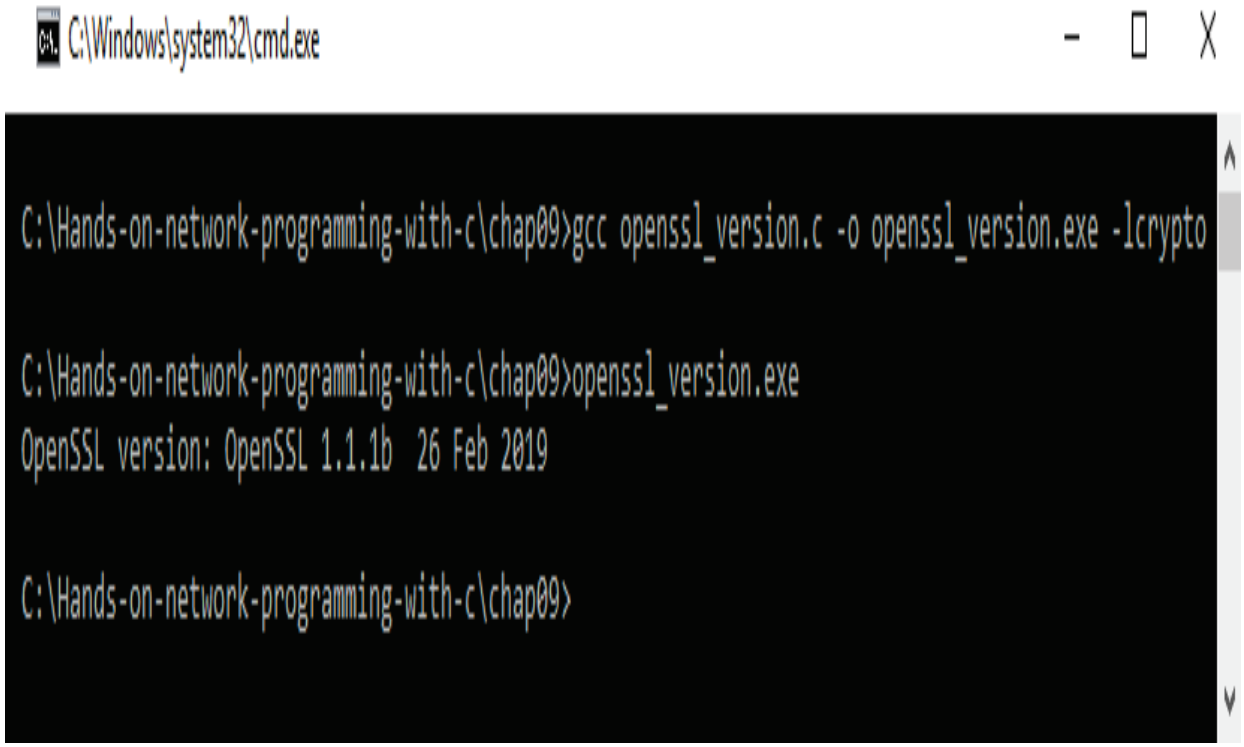
If you are brave, you can obtain the OpenSSL library source code directly from `https://www.openssl.org/source/`. You will, of course, need to build OpenSSL before it can be used. Building OpenSSL is not easy, but instructions are provided in the `INSTALL` and `NOTES.WIN` files included with the OpenSSL source code.

An easier alternative is to install prebuilt OpenSSL binaries. You can find a list of prebuilt OpenSSL binaries from the OpenSSL wiki at `https://wiki.openssl.org/index.php/Binaries`. You will need to locate binaries that match your operating system and compiler. Installing them will be a matter of copying the relevant files to the MinGW `include`, `lib`, and `bin` directories.

The following screenshot shows a binary OpenSSL distribution. The `include` and `lib` folders should be copied over to `c:\mingw\` and merged with the existing folders, while `openssl.exe` and the two DLL files need to be placed in `c:\mingw\bin\`:



You can try building `openssl_version.c` from `Chapter 9`, *Loading Secure Web Pages with HTTPS and OpenSSL*, to test that everything is installed correctly. It should look like the following:

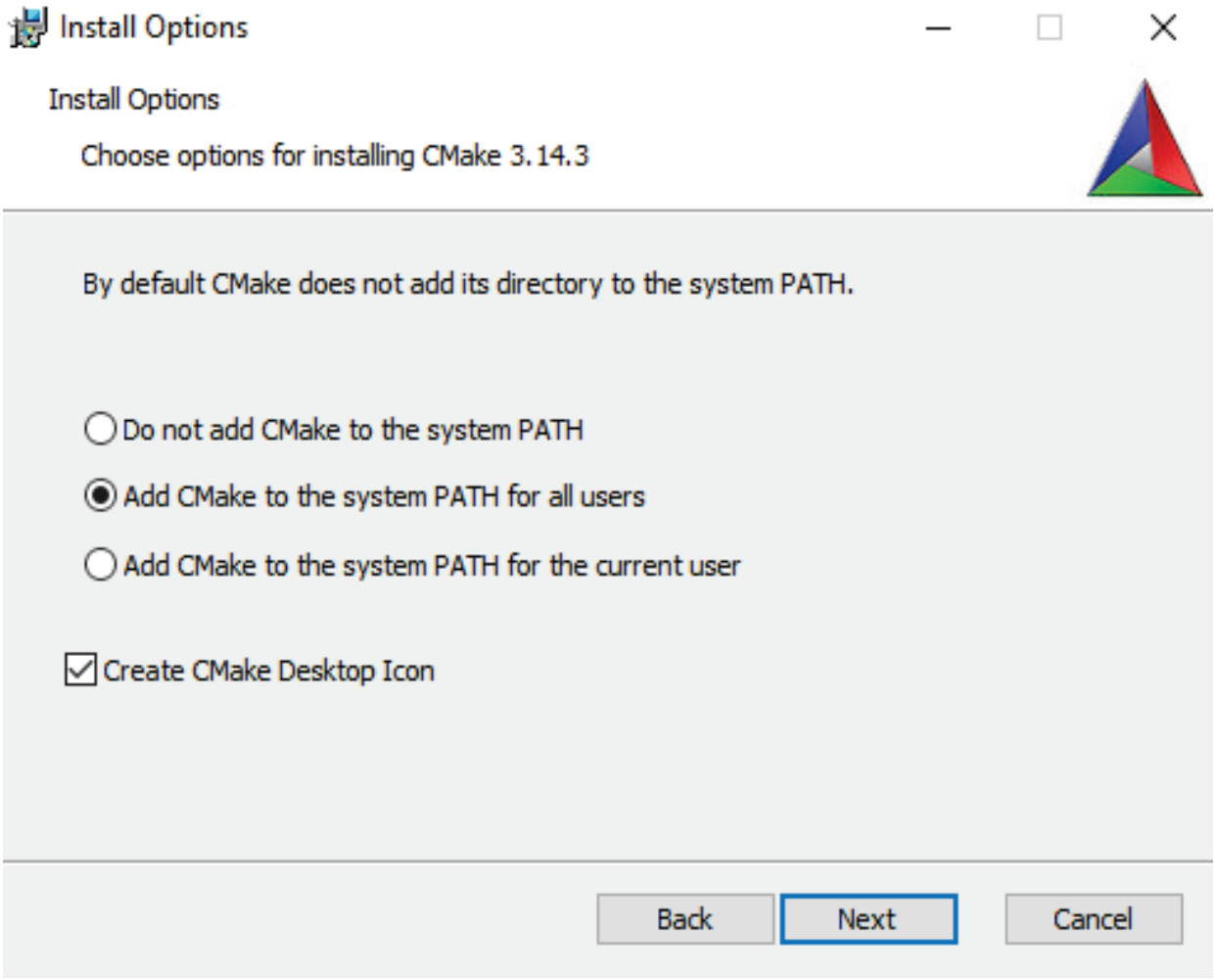

C:\Windows\system32\cmd.exe
C:\Hands-on-network-programming-with-c\chap09>gcc openssl_version.c -o openssl_version.exe -lcrypto
C:\Hands-on-network-programming-with-c\chap09>openssl_version.exe
OpenSSL version: OpenSSL 1.1.1b 26 Feb 2019
C:\Hands-on-network-programming-with-c\chap09>

Installing libssh

You can obtain the latest `libssh` library from `https://www.libssh.org/`. If you are proficient in installing C libraries, feel free to give it a go. Otherwise, read on for step-by-step instructions.

Before beginning, be sure that you've first installed the OpenSSL libraries successfully. These are required by the `libssh` library.

We will need CMake installed in order to build `libssh`. You can obtain CMake from `https://cmake.org/`. They provide a nice GUI installer, and you shouldn't run into any difficulties. Make sure you select the option to add CMake to your `PATH` during installation:



Once you have the CMake tool and the OpenSSL libraries installed, navigate to the `libssh` website to download the `libssh` source code. At the time of writing, Version 0.8.7 is the latest, and it is available from https://www.libssh.org/files/0.8/. Download and extract the `libssh` source code.

Take a look at the included `INSTALL` file.

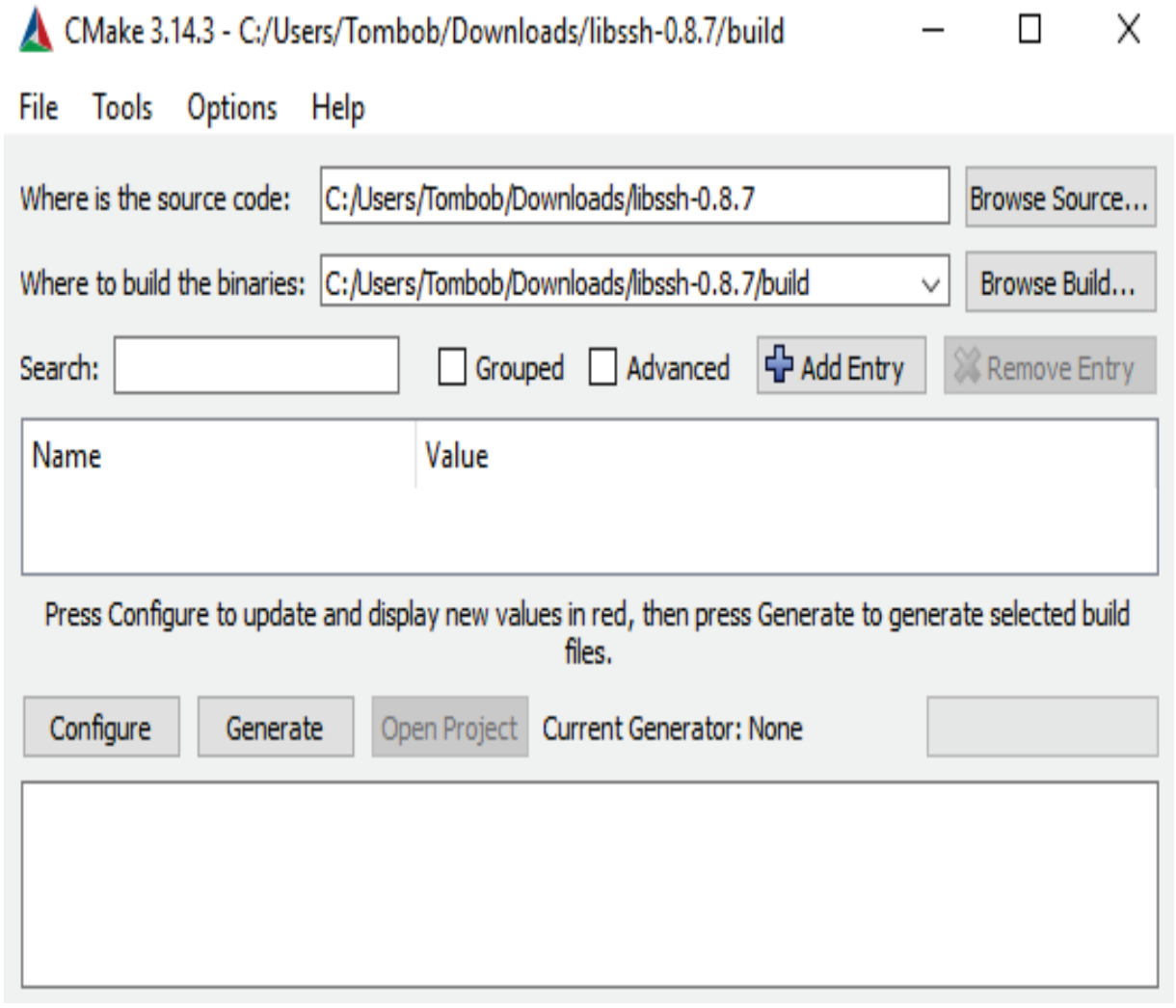
Now, open a command window in the `libssh` source code directory. Create a new `build` folder with the following commands:

```
mkdir build
cd build
```

Keep this command window open. We'll do the build here in a minute.

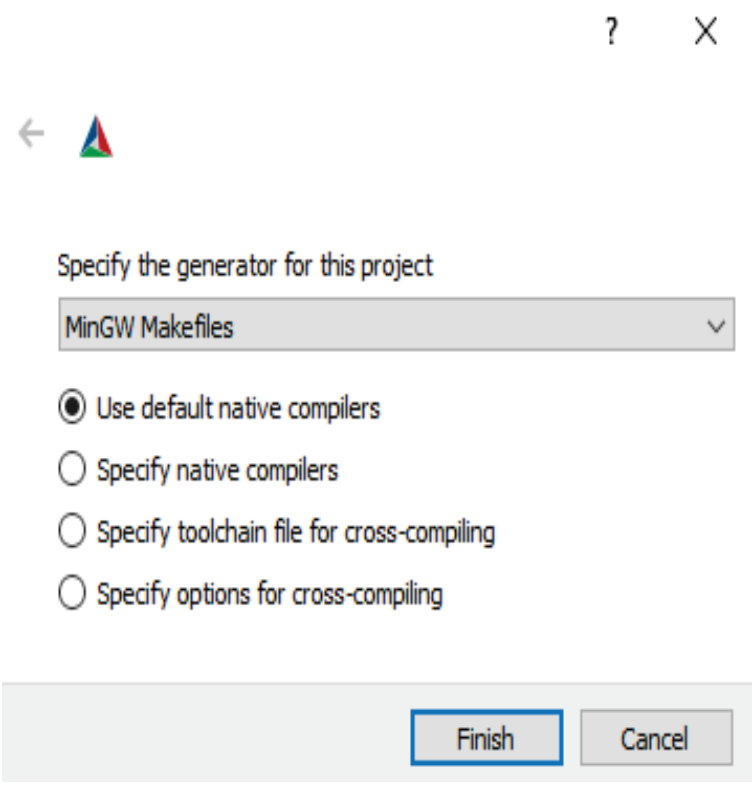
Start CMake 3.14.3 (**cmake-gui**) from the start menu or desktop shortcut.

You need to set the source code and build locations using the Browse Source... and Browse Build... buttons. This is shown in the following screenshot:



Then, click Configure.

On the next screen, select MinGW Makefiles as the generator for this project. Click Finish.



It may take a moment to process.

From the configuration options, make the following changes:

1. Uncheck WITH_NACL
2. Uncheck WITH_GSSAPI
3. Change `CMAKE_INSTALL_PREFIX` to `c:\mingw`

Then, click Configure again. It will take a moment. If everything worked, click Generate.

You should now be able to build `libssh`.

Go back to your command window in the build directory. Use the following command to complete the build:

```
mingw32-make
```

After the build completes, use the following command to copy the files over to your MinGW installation:

```
mingw32-make install
```

You can try building `ssh_version.c` from `Chapter 11`, *Establishing SSH Connections with libssh*, to test that everything is installed correctly. It should look like the following:

Alternatives

In this book, we recommend free software whenever possible. This is important for user freedom, and this is one reason we recommend GCC throughout the book.

In addition, to MinGW GCC, the Clang C compiler is also open source and excellent quality. The code in this book was also tested to run successfully using Clang on Windows.

Command-line tools such as GCC and Clang are often easier to integrate into the complicated workflows required for larger projects. These open source tools also provide better standards compliance than Microsoft's compilers.

That said, the code in this book also works with Microsoft's compilers. The code was tested for both Microsoft Visual Studio 2015 and Microsoft Visual Studio 2017.