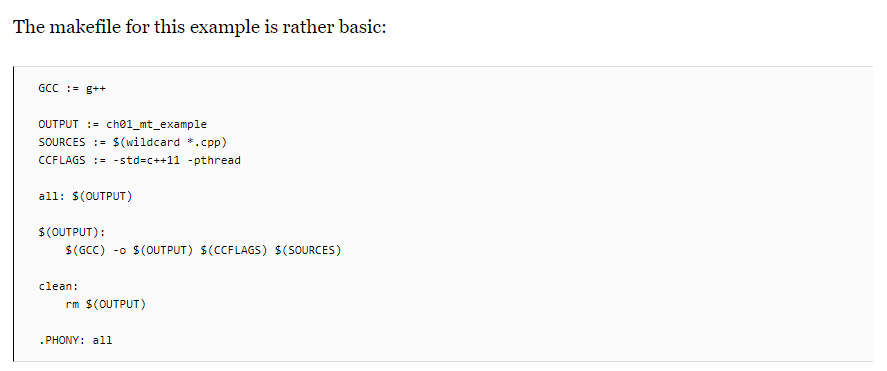
Makefile

In order to compile the code described earlier, one could use an IDE, or type the command on the command line. As mentioned in the beginning of this chapter, we'll be using makefiles for the examples in this book. The big advantages of this are that one does not have to repeatedly type in the same extensive command, and it is portable to any system which supports make.

Further advantages include being able to have previous generated artifacts removed automatically and to only compile those source files which have changed, along with a detailed control over build steps.



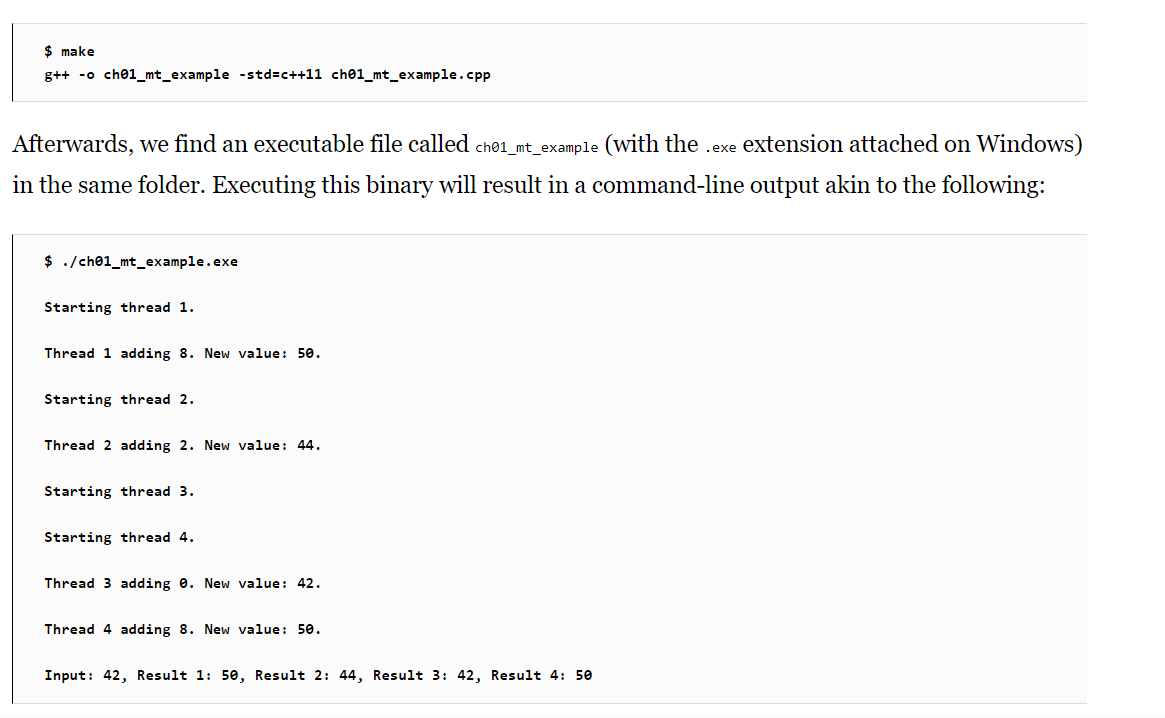
From the top down, we first define the compiler that we'll use (g++), set the name of the output binary (the .exe extension on Windows will be post-fixed automatically), followed by the gathering of the sources and any important compiler flags.

The wildcard feature allows one to collect the names of all files matching the string following it in one go without having to define the name of each source file in the folder individually.

For the compiler flags, we're only really interested in enabling the c++11 features, for which GCC still requires one to supply this compiler flag.

For the all method, we just tell make to run g++ with the supplied information. Next we define a simple clean method which just removes the produced binary, and finally, we tell make to not interpret any folder or file named all in the folder, but to use the internal method with the .PHONY section.

When we run this makefile, we see the following command-line output:



What one can see here already is the somewhat asynchronous nature of threads and their output. While threads 1 and 2 appear to run synchronously, starting and quitting seemingly in order, threads 3 and 4clearly run asynchronously as both start simultaneously before logging their action. For this reason, and especially in longer-running threads, it's virtually impossible to say in which order the log output and results will be returned.

While we use a simple vector to collect the results of the threads, there is no saying whether Result 1truly originates from the thread which we assigned ID 1 in the beginning. If we need this information, we need to extend the data we return by using an information structure with details on the processing thread or similar.

One could, for example, use struct like this:

