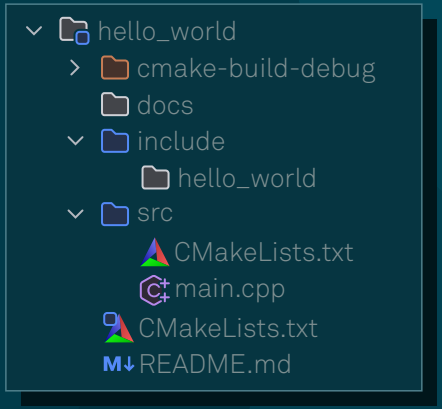


Project Creation and Structure

Basics – C++ Fundamentals

Agenda

- Compiling a C++ Executable
  - Basic Hello-world Executable
  - Compiling Manually
  - Makefiles
- Project Organization

Compiling a C++ Executable

Let's Start with Some Code

hello_world

Let's Start with Some Code

Let's Start with Some Code

main.cpp

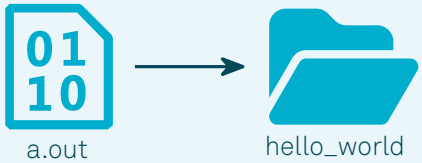
```cpp
#include <cstdlib>
#include <iostream>

int main() {
   std::cout << "Hello, world!" << std::endl;
   return EXIT_SUCCESS;
}
```

Compiling and Running

```
hello_world % clang++ -std=c++23 main.cpp
hello_world %
```

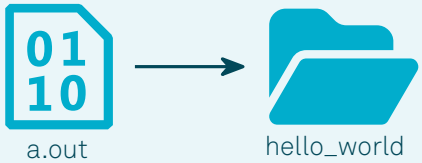
Compiling and Running

```
hello_world % clang++ -std=c++23 main.cpp
hello_world %
```



Compiling and Running

```
hello_world % clang++ -std=c++23 main.cpp
hello_world %
```




```
hello_world % ./a.out
Hello, world!
hello_world %
```

What Happens with Larger Projects?

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

- More files means more work when compiling

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

- More files means more work when compiling
- Adding in outside code makes this more difficult

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

- More files means more work when compiling
- Adding in outside code makes this more difficult
- The order files are compiled matters, so this isn't a trivial task

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

- More files means more work when compiling
- Adding in outside code makes this more difficult
- The order files are compiled matters, so this isn't a trivial task
- Not impossible to do manually, but it's a lot of work

What Happens with Larger Projects?

- For larger projects, code is split into multiple files

```
hello_world % clang++ -std=c++23 main.cpp file1.cpp file2.cpp
hello_world %
```

- More files means more work when compiling
- Adding in outside code makes this more difficult
- The order files are compiled matters, so this isn't a trivial task
- Not impossible to do manually, but it's a lot of work
- This is where makefiles come in!

Makefiles

What is a makefile?

A *makefile* is a file run by a program called “make” which automates shell commands

Makefiles

What is a makefile?

A *makefile* is a file run by a program called “make” which automates shell commands

- Our makefile might need to change throughout the development of a project

Makefiles

What is a makefile?

A *makefile* is a file run by a program called “make” which automates shell commands

- Our makefile might need to change throughout the development of a project
- CMake automates the writing of makefiles from a high level

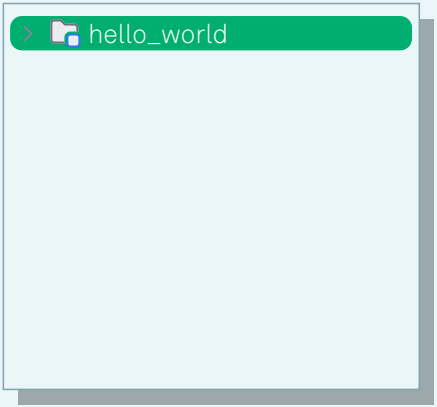
Project Organization

Project Organization

- Keeping our files organized will make compiling a bit simpler, and it will standardize our projects

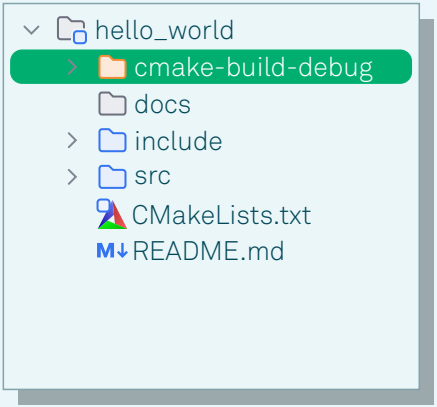
Project Organization

- The project directory where everything for a project is stored



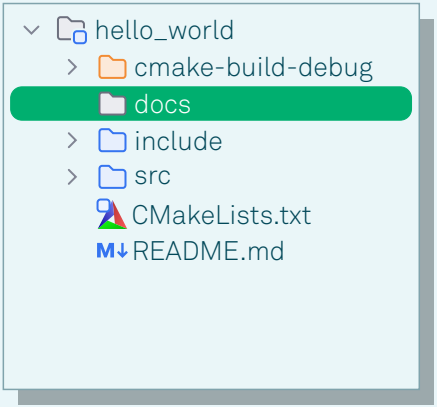
Project Organization

- The build directory where the build files are stored



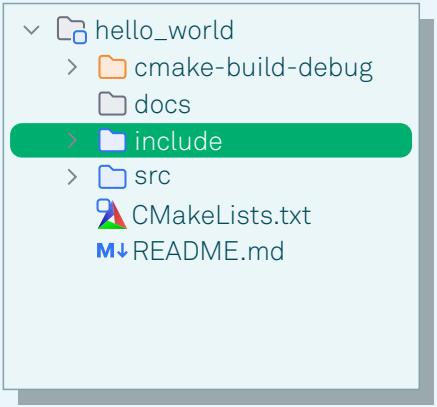
Project Organization

- The documentation directory where all files related to documentation are stored



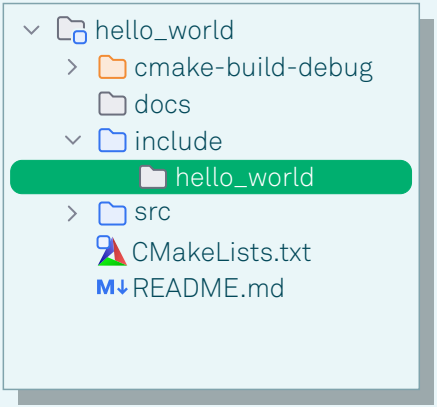
Project Organization

- The include directory where all files we want to include will go



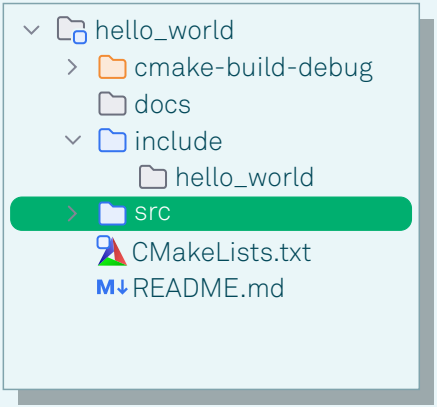
Project Organization

- The include project directory where the files we need to include for our project will go



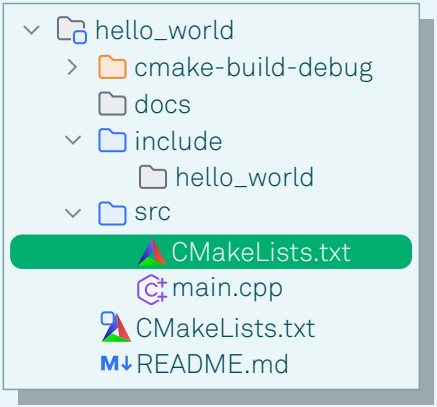
Project Organization

- The source directory where all source files go



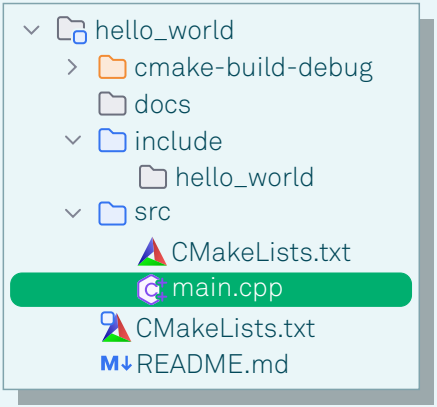
Project Organization

- The source CMake file which dictates how to compile the source files and the include files



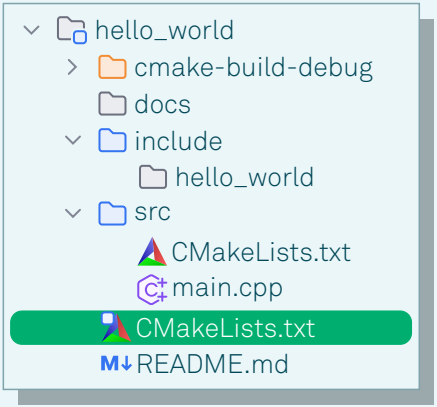
Project Organization

- The main file where our executable will start



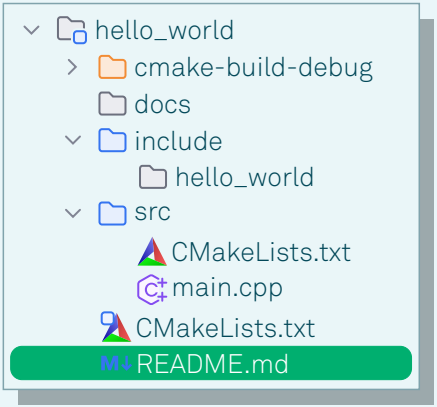
Project Organization

- The global CMake file which dictates project information, compiler settings, and external dependencies



Project Organization

- The read-me file which holds instructions for how to use our project



Creating Projects

- By default, our IDE won’t create new projects this way

Creating Projects

- By default, our IDE won't create new projects this way
- Different ways to set up a C++ project depending on use

Creating Projects

- By default, our IDE won't create new projects this way
- Different ways to set up a C++ project depending on use
- We'll use a makefile to create new projects with the proper structure and the proper CMake to configure them

Any Questions?