Contents

<Sun 12.24.23> Note: so far this file only details what was covered in meetings. I will add much more content to it later.

1 The C compilation process

If you are writing your program entirely in one file, compiling your program with this shell command will do:

```
cc file.c -o executable_name
```

If you split your code across *multiple* **source files** (.c files) and **header files** (.h files), however, you may want to understand the compilation process at a deeper level. That is what this document will detail.

1.1 Translation units and object files

A **translation unit** or **compilation unit** is a collection of source code that the compiler reads to output an object file. A translation unit is made up of a source file and all of the header files it may include.

An **object file** (.o file) contains the machine code of a translation unit. One or more object files are fed to the **linker**, a program which will link data, such as function calls between object files, to output the final executable of a program.

1.2 Producing direct object files

When you run this command:

```
cc file.c -o executable_name
```

The compiler will produce object files and link them automatically for you. To only produce object files and skip linking, use the `-c` flag:

```
cc -c file.c
```

This will produce the object file `file.o`. If you want to explicitly choose the object file name, use the `-o` flag:

```
cc -c file.c -o file.o
```

1.3 Linking object files to produce an executable

To produce an executable using all of your object files, run this command:

```
cc file.o -o program
```

This command will produce the executable file `program`. You may insert as many object file names before `-o` as you like.