**CS5900 Special Topics - Programming in C**

Summer 2015

**Separate Compilation Information**

All of the programs that we have looked at to this point have been contained within a single file. This is the exception rather than the rule. As programs become larger, it is important to spread them across files of a reasonable size. There are some parts of the program that are generic and could be separated out for possible reused.

**Separate compilation**

Separate compilation is an integral part of the standard for the C programming language. When a C source code file is compiled there are two tasks performed by the compiler. First, the file is compiled into a format called an *object file*. Then the object file is linked with a library of standard C functions to produce an executable file. An object file contains coding of the source code file into language that the machine understands. It is incomplete in that it may contain calls to functions that are in other files, and it need not contain a main function.

The process of **dividing** a large program into many smaller source files is called *separate compilation*. It offers several benefits:

* Some project management systems (e.g., RCS) do not allow concurrent editing of files; therefore, if we have multiple people working on a project, it is a good idea to break it into pieces so that everyone can work on their files.
* The speed of recompilation increases. When we make changes to the program, we frequently do not need to recompile the whole program, but only files that have been changed.
* The code is easier to maintain (modify, or debug), because each file is smaller, and hence it is easier to find the part of the code that we need to update.
* The program is more modular. That is, data and functions that are related are put together, but are separated from the rest. This makes it easier to reuse the code in future

**Header files and #includes**

If the main program calls a function that is defined in another file, then the compiler needs to see a declaration for the function before the call in order to check correctness of argument and returned value types. In addition, the main program may need to declare variables whose types are declared elsewhere. This is done with *header files*, where shared types and functions are declared. Conventionally, the name of a header file ends in a .h suffix.

<https://www.cs.utah.edu/~zachary/isp/tutorials/separate/separate.html>

<https://www.cs.bu.edu/teaching/c/separate-compilation/>

<http://www.d.umn.edu/~gshute/make/home.html>

<http://www.ugrad.cs.ubc.ca/~cs219/CourseNotes/Make/sepCom.html>

Checkout the links above...