NWEN 241
Systems Programming

Week 1 Tutorial

Tutorial-Style Lecture Plan

- Compilation Process
- I/O Using Standard C Library
- Introduction to Functions

What You Need to Program in C/C++

- **Text editor** to type in code
  - Any text editor will do (even notepad)
  - Suggested editors: Sublime Text, Kate (Linux only)

- **C/C++ toolchain** (pre-processor, compiler, assembler, linker, debugger)
  - Already installed in ECS lab computers (CO246) and servers

- **Terminal** to run compilation commands and execute program

Compilation Process At A Glance

1) Preprocessing Phase

2) Compilation Phase

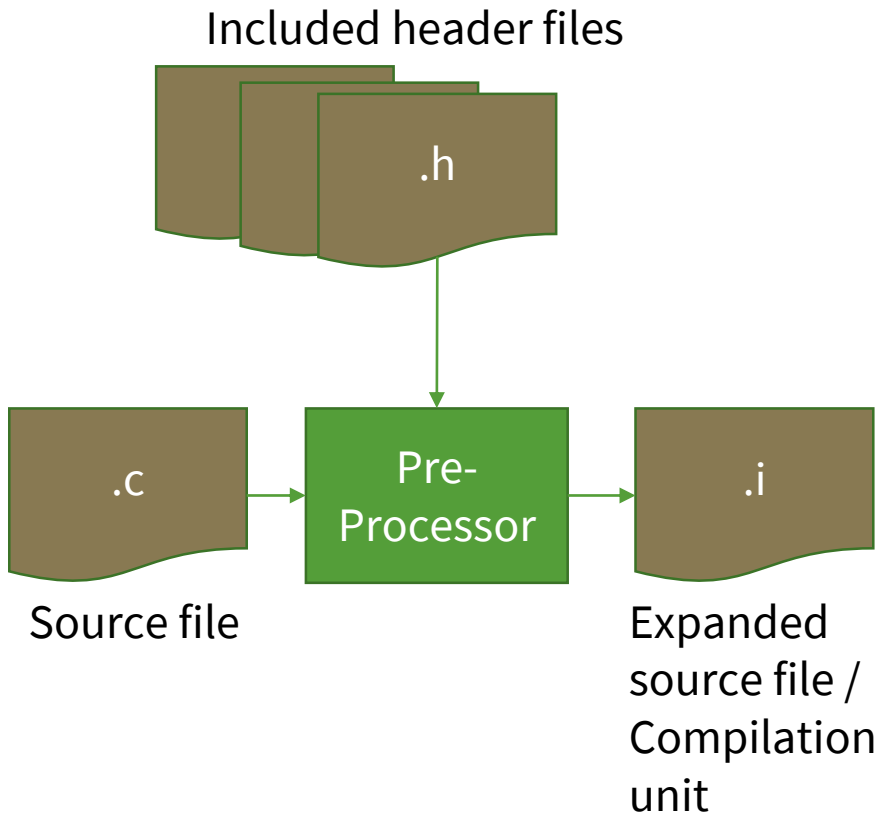
3) Assembly Phase

4) Linking Phase

Preprocessing Phase

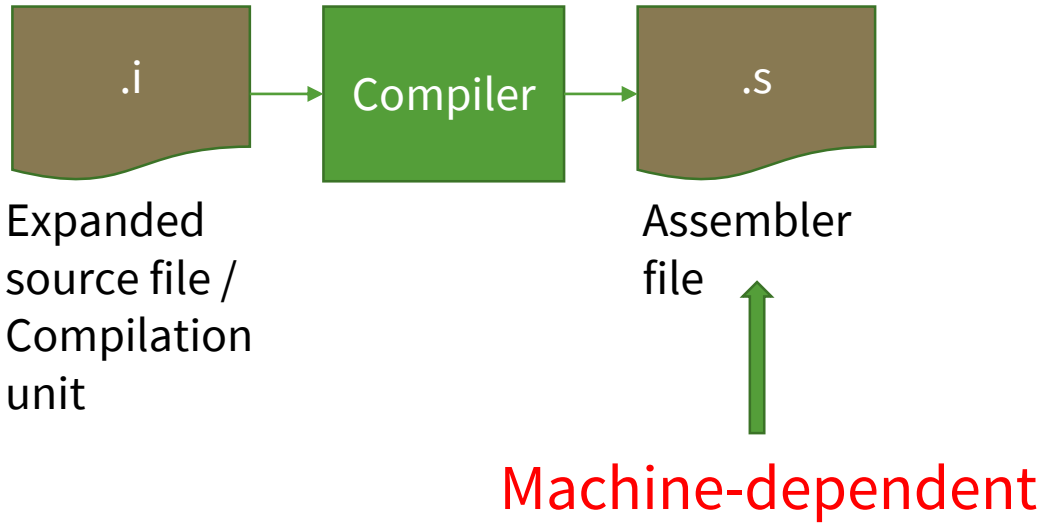
- The **preprocessor** modifies the original C program according to directives that begin with the '#' character
  - Example: #include <stdio.h> command tells the preprocessor to read the contents of the system header file **stdio.h** and insert it directly into the program text.

- The result is another C program, typically with the **.i** suffix.



Compilation Phase

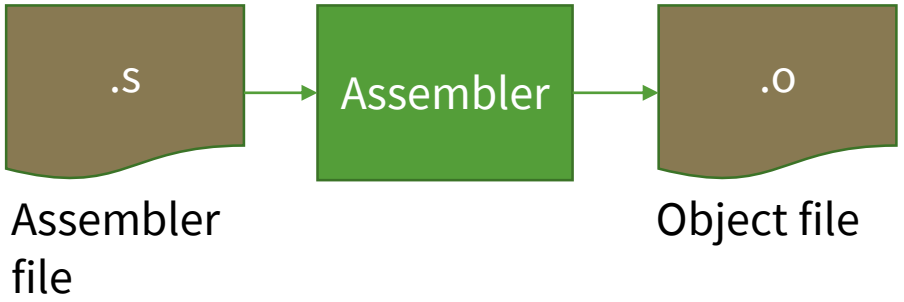
- The **compiler** translates the text file (**.i**) into the text file (**.s**), which contains an assembly-language program.



Assembly Phase

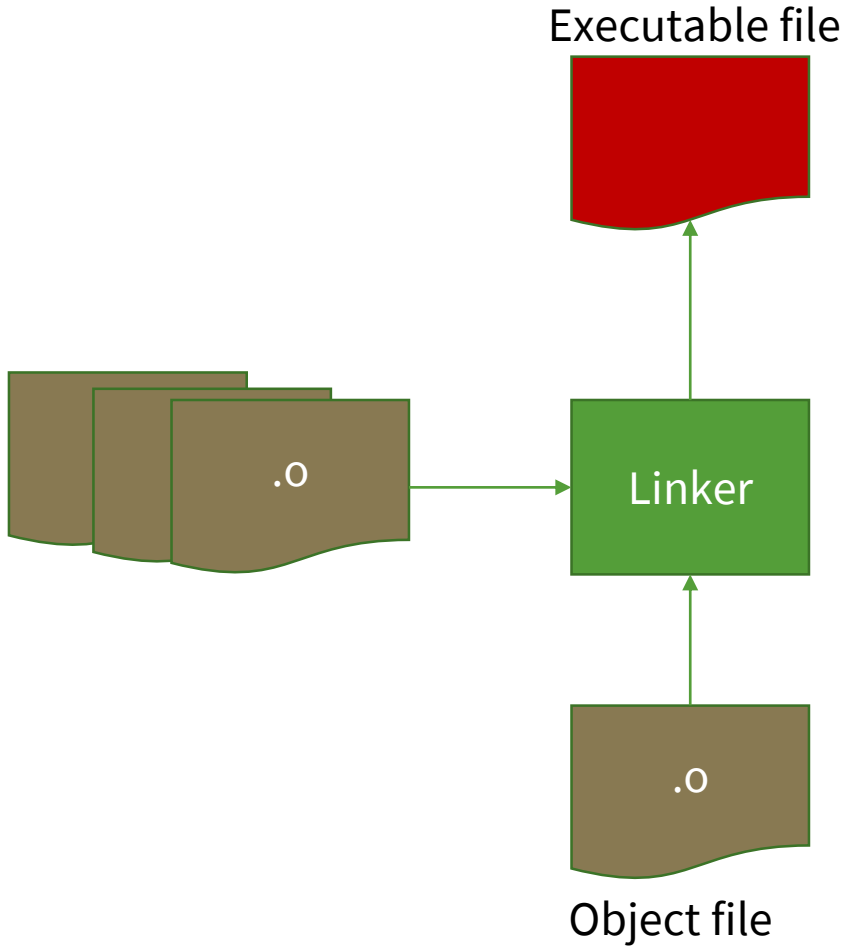
- The **assembler** translates assembler file (**.s**) into machine-language instructions, packages them in a form known as a *relocatable object program*, and stores the result in the *object file* (**.o**).

- Object files are binary - if you try to open one with a text editor, it would appear to be gibberish.



Linking Phase

- The **linker** looks for external object files needed by the program and merges these with the object file generated in the assembly phase, creating an executable object file (or simply *executable*) that is ready to be loaded into memory and executed by the system.



In Practice

- All the phases can be done in one step using the GNU C Compiler (gcc)

`hello.c`

```
#include <stdio.h>
int main(void)
{
   printf("Hello world\n");

   return 0;
}
```

```
gcc hello.c
```

Generates executable file **`a.out`**

```
gcc hello.c -o hello
```

Generates executable file **`hello`**

`gcc` Options

Phase	gcc Option	Result	Output File
Preprocessing	`-E`	Compilation unit	`.i` `.ii`
Compilation	`-S`	Assembler file	`.s`
Assembly	`-C`	Object file	`.o` `.obj`
Linking		Executable	Binary executable (`.exe` in Windows)

I/O Using Standard C Library

Recall: C provides a set of header files (standard C library) that you can use to write your code

C provides a standard library* which consists of the following headers:

`assert.h`	`float.h`	`math.h`	`stdarg.h`	`stdlib.h`
`ctype.h`	`limits.h`	`setjmp.h`	`stddef.h`	`string.h`
`errno.h`	`locale.h`	`signal.h`	**`stdio.h`**	`time.h`

- You don’t have to start from scratch!

I/O Streams

- C provides functions with input and output capability
- From the program's point of view, data input and data output are made possible through file streams
- Every C program has access to 3 such file streams: `stdin`, `stdout`, `stderr`

File	Description	Remarks
`stdin`	Standard input file	Connected to the keyboard
`stdout`	Standard output file	Connected to the screen
`stderr`	Standard error file	Connected to the screen

I/O Functions

- C input/output functions can be classified into 2 types:
  - Non-formatted input/output
    - `getchar`
    - `putchar`
    - `gets`
    - `puts`
  - Formatted input/output
    - `printf` and its variants
    - `scanf` and its variants

How To Use a Function

- Find its manual or documentation
  - In Linux terminal, use the **man** command
  - You can also search online
    - This website provides a pretty good documentation for the standard C library: https://www.tutorialspoint.com/c_standard_library/index.htm
- What to look for in the function manual?
  - What the function does
  - What header file(s) to include
  - What are the arguments to the function
  - What is the return type
  - What happens in case of errors

`printf()` and `scanf()`

- `printf()` writes a string to the standard output stream (stdout).
- The string is formatted using additional arguments that follow the initial string.

- `scanf()` accepts input from the standard input stream (stdin).
- The format of the expected items are specified and it returns the number of items successfully scanned