Engineering & Computer Science
Syracuse University

CIS 657 – Principles of Operating Systems

Topic: Background on C

Endadul Hoque

Acknowledgement

- http://pages.cs.wisc.edu/~powerjg/cs354-fall15/Notes/C.basics.html

Why learn to program in C?

- **Practicality**
  - Many, many companies/research projects do all their programming in C.
  - Many, many existing programs and systems are written in C, and you might be the one chosen to modify a C program.
  - Many courses in CS and security research expect you to know C, or be able to pick it up on your own.

Why learn to program in C?

- It is an industry standard.
- It looks good on your resume.

Why learn to program in C?

- **Concepts.**
  - C is occasionally called a “low-level,” high level language.
  - The **semantics** of C often mimic what actually occurs in the machine code (or assembly language, if you prefer to think of it in that way)
  - Due to the low level nature of C, the programming constructs that it provides to the programmer (such as **pointers**) are essential to know.
  - Understanding these concepts and constructs may allow you to be a better programmer; one who can write better code, and code that runs faster. Please note that this is true, even if you only write Java code for the rest of your life!

Assumptions on Your Background

- **You have written high level language programs.**
  - E.g., Java, Python
- **You understand some basic program constructs**
  - variables and types and what a declaration is
  - control structures such as for loops, while loops, and if statements
  - methods/functions, as well as parameters. What they are, and how to use them in your favored language.

Difference with OOP

- Object oriented (OO) programming (e.g., Java, C++) allow typing and structure that far surpasses non-OO languages like C.
- This has the further implication that a compiler can check for correctness of many more items within the source code.
- Therefore, more bugs are caught at compile time.
  - The more bugs caught, the better.
- Did you know that most completed, production code has lots of bugs?
  - They just have not yet been tested for, or caught.

Basics of C

- C is a programming language
  - invented (derived from B, actually) to be a low-level language that would facilitate more easily describing/writing operating system code.
- It is general purpose.
- The code itself is rather compact.
- C is a **procedural** language. This distinguishes it from (later invented) object-oriented languages.

No Objects. No member methods.

- So, how does anything get accomplished without objects and methods?
  - Computers are really just fancy calculators.
  - Combined with the stored program concept, computers are fancy (and fast) calculators that can re-do their calculations over and over.
- What does a computer/calculator do?
  - Arithmetic. On variables.
  - Variables are numerical values that may change over time.
- The C language manipulates variables. (Just like any other programming language.)

No Objects. No member methods.

- In a procedural language, **procedures** (also called **functions** or **subroutines**) are the “equivalent” of an object-oriented language's methods.
- In C, we call them **functions**. They operate on parameters (which are often variables).
- The control structures (of Java) that you already know *were derived from C*!
  - The designers of Java (C++, too!) knew that the vast majority of programmers already knew C.
  - Since no one was complaining about the syntax used in C, and so many already knew the syntax, the designers of the Java language used the same syntax!
  - This implies that learning C should be quite easy. . .

A Sample program

```
#include <stdio.h>

#define MAX 100

main()
{
   int x;

   x = 1;
   while (x <= MAX) {
      printf("%d\n", x);
                  x++;
   }
}
```

```
#include <stdio.h>

main()
{
   int x;

   for (x = 1; x <= 100; x++) {
      printf("%d\n", x);
   }
}
```

A Sample program

```
#include <stdio.h>

#define MAX 100

main()
{
   int x;

   x = 1;
   while (x <= MAX) {
      printf("%d\n", x);
               x++;
   }
}
```

Some things you need to know about C:
(you probably can figure out most of it without an explanation!)

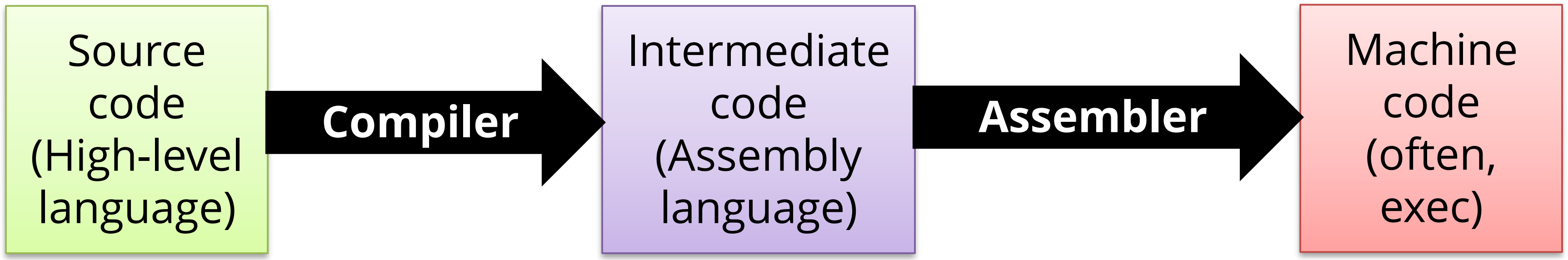
- **main()** is the name of a function. It is very much the same as the main method in a Java class.
- **printf()** is also the name of a function, similar to Java's **system.out.print()**
- This program can use **printf()** function, because of the line **#include <stdio.h>**, similar to Java's **import java.io.*;**
- **#define MAX 100** essentially defines a constant (but needs more explanation)

Some other things

- Comments are enclosed by the strings **/*** and ***/**
- **//** is used for single line comment (up to a newline)
- There is no equivalent to Java/C++ exceptions.
- The code must be written to detect and handle error conditions, which are often given in a **return value** from a function.

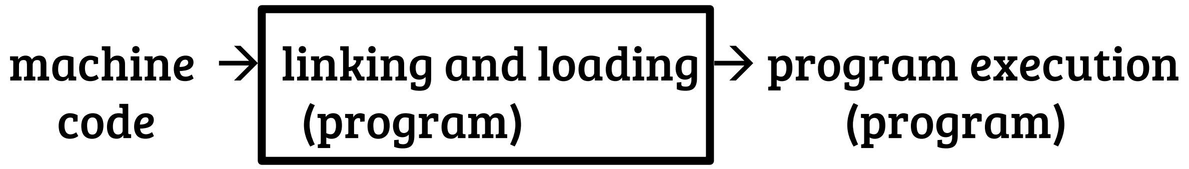
C Program: Source to execution

- The standard way to generate and eventually execute a program is much the same for all high level languages


Source code (High-level language)
Compiler
Intermediate code (Assembly language)
Assembler
Machine code (often, exec)

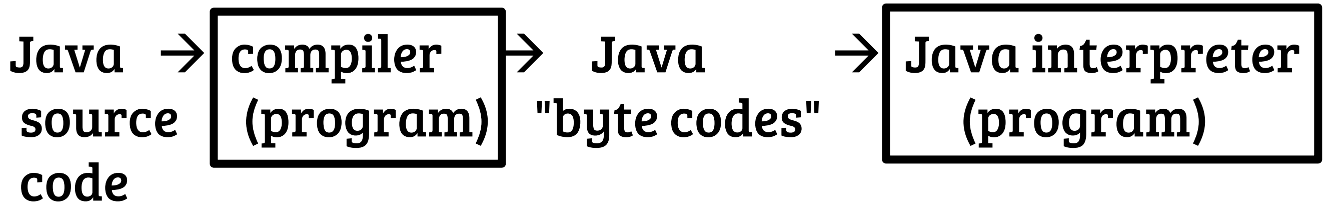
C Program: Source to execution

- Once we have a machine code



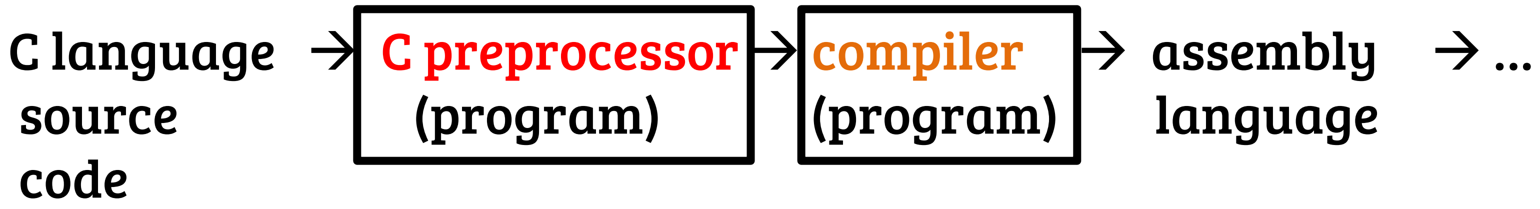
C Program: Source to execution

- Without going into too much detail, Java programs do not exactly go through this set of steps. Instead Java programs are “executed” by an **interpreter**



C Program: Source to execution

- For C programs, compilation has two conceptually separate steps



Intro to C (Review)

- See the other slide (intro-c.pdf)

C Preprocessor

- Does 3 important things. Within source code, each item that matters to the preprocessor is identified by the **#** (**pound sign**) at the beginning of a line.
- **1. File inclusion**, also called header files.
  - Examples: **#include <stdio.h>**
  - These files are included in the source code of the program. They **contain required declarations** (of symbols and/or functions).
  - File names enclosed within **<** and **>** characters cause the search for the file (like this standard I/O library file) to begin in a system-defined location.
  - File names enclosed within **double quote marks** cause the search for the file to begin in the same directory as the source file being pre-processed.

C Preprocessor

- **2. Macro substitution.**
  - Macro substitution takes a **fixed character sequence** and **substitutes** it with **another character sequence**
  - Define a **constant** value in a C program. Here is a simple example: **#define MAXITERATIONS 10000**
    It substitutes the integer value 10000 for every instance of **MAXITERATIONS** within the source code. It is efficient (for code execution *and* for memory allocation).
  - Another example: **#define max(A, B) ((A) > (B) ? (A) : (B))**
    We write source code **x = max(y, z-4);**
    and the preprocessor substitutes all the right values in all the right places. This is ***more* efficient than implementing a function** do the same thing. For *every* function call, it takes execution time to **set up** and **return** from the call.

C Preprocessor

- **3. Conditional compilation**.
  - It essentially provides a way of defining a variable, such that based on the variable's value, a specific portion of (source) code **is or is not compiled** into the resulting object code.
  - Code must often be different (do different things) to be correct when executing under different operating systems.
  - When compiled for a target system (meaning that the operating system is defined) only the code intended specifically for that system is compiled in.
  - This makes the code **more general**, and able to be compiled for more operating systems.

Questions?