PURDUE
UNIVERSITY®

CS 240: Programming in C

Lecture 2: Compiling, Object Files, Linking, and Executing

Prof. Jeff Turkstra

Announcements

- Homework 1 will be released on Monday
- The answer to Quiz 0 was True. Discussing implementation details surrounding an assignment is cheating, even if code is not directly viewed or exchanged!

Supplemental Instruction

- Monday/Wednesday 5:30pm – 6:20pm, WALC 3138
  - Keshav Shylesh
    - kshylesh@purdue.edu
- Monday/Wednesday 7:30pm – 8:20pm, WALC 2121
  - Manya Gupta
    - gupta969@purdue.edu
- Data show students that regularly attend SI sessions obtain higher grades in the course, on average

Gradescope

- Takehome Quizzes must be hand written
  - No credit otherwise
- We will use Gradescope in this course
  - https://www.gradescope.com/
- There are two Gradescope “classes” for CS 240
  - One is for the code standard
  - One is for the quizzes
- We’d really like you to scan your quiz, if at all possible
  - But if you can’t, a picture is okay

Tips for Success

- Write your programs out on paper first
- Start on homework projects as early as possible
- Practice
- Experiment
- Create a cool personal project
- Look at source code available on the Internet
- If you can learn to enjoy programming you are guaranteed to do well

How to Fail This Course*

- Assume that since your prerequisites were easy, this class will be too!
- Try to do the homework at the last minute
- Don’t do the homework at all
- Don’t come to lecture
- Get “help” from somebody else in the class
- Don’t practice for the exams

* most students do quite well

Why C?

- It is still widely used and growing
  - TIOBE January 2025 index lists it as #4
  - Python is #1, C++ is #2
- Programming Language of the Year 2019
- Used in a huge number of embedded and IoT devices
- Ubiquitous for systems programming
- Small
- Fast

C vs. Java

- You already know how to write some C code
  - Java was designed using C/C++-style syntax
- But, there are many differences

Slides

- Some slides based on Prof. Rodgriguez-Rivera’s, Prof. Fred Mowle’s, and Dr. Richard Kennell’s material

Java vs. C

Attribute	C	Java
language type	function oriented	object oriented
basic programming unit	function	class (ADT)
source portability	yes, when done right	yes
compiled binary portability	recompile for each architecture	"write once, run anywhere" (bytecode)
security	yes, when done right	more built-in safety
compiling	gcc hello.c	javac Hello.java
linking a library (e.g., math)	gcc -lm mycalc.c	javac MyCalc.java (no flags needed)
execution	a.out (or name of binary)	java MainClass
(runtime) array declaration	int *a = malloc(N * sizeof(*a))	int[] a = new int[N];
array size	often unknown	a.length
strings	'\0'-terminated char array	built-in immutable String type
using a library	#include <stdio.h>	import java.io.File;

Java vs. C

Attribute	C	Java

accessing a library function	#include <math.h>
	x = sqrt(2.2);
		import java.util.Math;
		x = Math.sqrt(2.2);

standard output	printf("sum = %d\n", x);	System.out.printf("sum = %d\n", x);
reading from stdin	scanf("%d", &x);	int x = StdIn.readInt()
memory address	pointer	reference
manipulating pointers	*, &, +	not permitted
functions	int max(int a, int b)	public int max(int a, int b)
data structures	struct	class
methods	function pointers	yes
pass-by-value	yes	yes
allocating memory	malloc()	new
de-allocating memory	free()	garbage collection
constants	const and #define	final

Java vs. C

Attribute	C	Java
variable auto-initialization	not guaranteed	yes, compile-time checking
data type for generic item	void *	Object
variable naming convention	sum_of_squares	sumOfSquares
file naming convention	stack.c, stack.h	Stack.java
assertions	assert	assert

* Not a complete list :-)

C

- Created by Dennis Ritchie 1969-1973 at Bell Labs
- Early operating systems typically implemented entirely in assembly
  - Not portable
- Desire to make UNIX portable
- With C, only about 5% in assembly
  - Much easier to port to different architectures

More C

- The Linux kernel is written in C
  - So is most of Windoze's and Mac's kernel
- Many libraries and programs are also in C
  - Especially if they need to be fast
- The Java JVM is in C(++)
  - So are some of its native libraries
- Embedded systems
- Firmware
- Drivers
- ...and a lot more

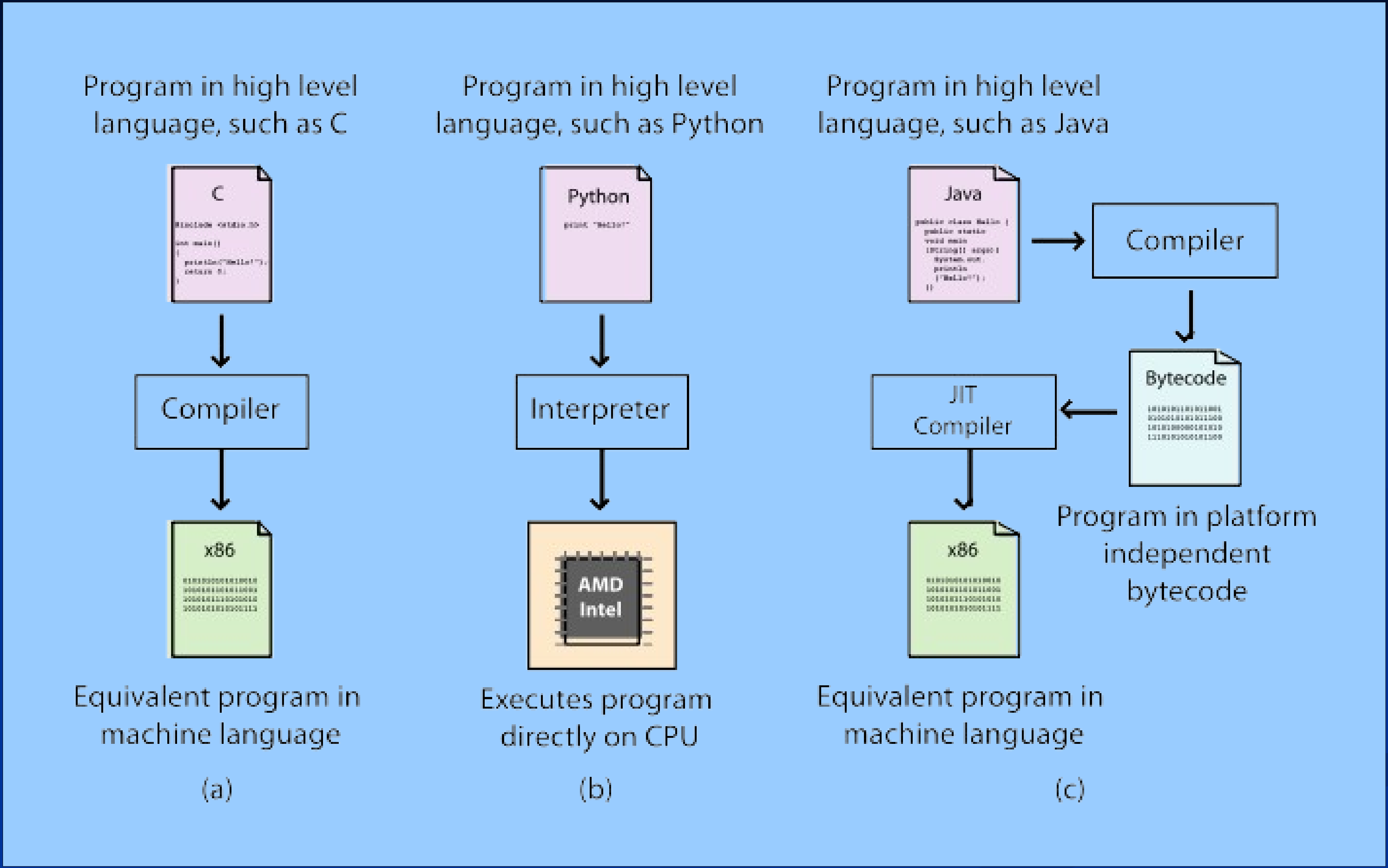
Why C?

- It is fast
- It is powerful
- It is simple
  - Easy to do low-level things
  - No abstractions to worry about

The C Standard

- C is alive and well
  - ...and continues to evolve

- ANSI C or ISO C
  - Refers to C89 or C90
- C94 or C95
- C99
- C11 (C11X)
- C18
- C23



Program in high level language, such as C
C
Compiler
x86
Equivalent program in machine language
(a)
Program in high level language, such as Python
Python
Interpreter
AMD Intel
Executes program directly on CPU
(b)
Program in high level language, such as Java
Java
Compiler
Bytecode
JIT Compiler
x86
Program in platform independent bytecode
Equivalent program in machine language
(c)

Purdue trivia

- The Purdue Exponent is one of a handful of independent college newspapers
  - Published by the Purdue Student Publishing Foundation
- Founded on December 15, 1889
- Became independent in 1969
- 80+ students and seven full time professionals
- Many notable alums
  - Six Pulitzer Prizes, four Emmys, one Oscar
- Only one of two college newspapers that own and operate their own printing press.

Takehome Quiz 0

....on Ed Discussion!

Compiling

- For this course, we will be using the `gcc` compiler exclusively
- Survey the flags to use
- Look at error messages

gcc

- GNU Compiler Collection
- Standard compiler for most UNIX-like operating systems
- First released March 22, 1987
- Many different front ends: C, C++, Objective-C, Objective-C++, Fortran, Java, Ada, Go

Common gcc flags

-c	Compile file into object code
-g	Include debug symbols
-Wall	Enable ALL warnings
-Werror	Turn warnings into errors
-O	Optimize the output file
-o file	Output to 'file'
-ansi	Adhere to the ANSI standard
-std=X	Adhere to some standard X

Examples of flags

- Compile file.c into file.o and make it debuggable. Enable all warnings…

```
gcc -g -Wall -c file.c
```

- Compile X.c into Y.o. No debug. C99.

```
gcc -c X.c -std=c99 -o Y.o
```

- Compile and optimize…

```
gcc -O -c file.c
```

What is an ‘object’ file?

- An object file is like an incomplete executable
  - It is the compiled form of a C module
  - It contains binary code
  - It contains a symbol table
  - Usually has a .o or .obj filename extension
- To create an executable from multiple object files, we need to link them together
- One object must contain main()
- gcc knows how to link objects too!

Examples of linking…

- Compile two C files and link them together:

```
gcc -Wall -Werror -g -c file1.c
gcc -Wall -Werror -g -c file2.c
gcc -o my_progr file1.o file2.o
```

- Could do the same thing in one step, without generating object files:

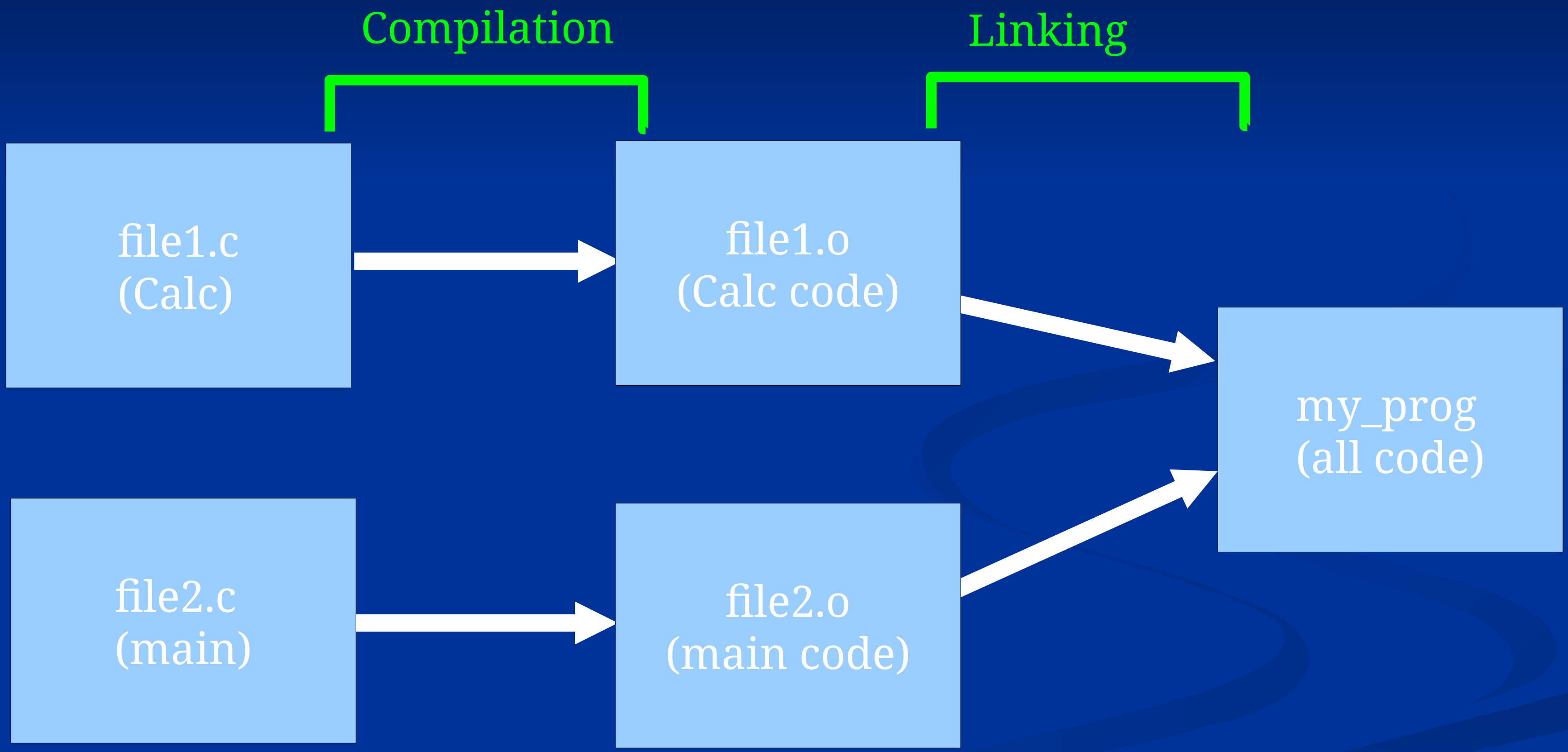
```
gcc -Wall -Werror -g -o my_prog file1.c
file2.c
```

  - But then it doesn’t produce any .o files
- I know what you’re thinking…
  ”Why do we want all of these object files?”

Why object files?

- It takes a long time to compile "big" applications if they consist of lots of C files.
  - It's better to do incremental compilation of the application
- You can give parts of programs to people without letting them see the source code
  - That's the way your homework will be

Illustration of compile/link



Executing

- If there were no errors compiling or linking your program, you can invoke it by typing its name:

```
$ gcc -Wall -Werror -c hello.c
$ gcc -o hello hello.o
$ ./hello
Hello, world!
```

Common errors

- When putting functions into separate modules, they need to have prototypes (forward declarations for functions)
  - Prevents type mismatches
  - A little extra bookkeeping for the programmer to make sure of types

file1.c

```
float calc(float first_val, float second_val)
{
  float temp = 0.0;

  temp = first_val * second_val;

  return temp;
}
```

file2.c

```
#include <stdio.h>

int main() {
   float result;

   result = calc(11.10, 3);

   printf(“My salary is $%f\n”, result);

   return 0;
}
```

file2.c with prototype

```
#include <stdio.h>

float calc(float first, float sec);

int main() {
  float result;

  result = calc(11.10, 3);

  printf(“My salary is $%f\n”, result);

  return 0;
}
```

For Next Lecture

- Keep practicing
- Read Chapter 2 of K&R
  - Skip section 2.7
- Read Beej’s up through Chapter 3.3
  - Optional, but recommended
- Homework 1 will be released Monday!

Boiler Up!