Chapter 11
Introduction to Programming in C

There are 10 kinds of people in the world…
…those that know binary,
and those that don't.

Aside: What is Unix?

The most influential operating system

First developed in 1969 at AT&T Bell Labs
- By Ken Thompson and Dennis Ritchie
- Designed for "smaller" computers of the day
- Reject some of the complexity of MIT's Multics

They found writing in assembly tedious
- Result: Dennis Ritchie invented the C programming language

Introduced to UC-Berkeley (Cal) in 1974
- Bill Joy was an early Unix hacker as a PhD student at Cal
- Much of the early internet consisted of Unix systems Mid-80s
- Good, solid TCP/IP for BSD in 1984

Linux
- Free implementation of Unix (libre and gratuit)
- Announced by Linus Torvalds in 1991

Much more in CSE380!

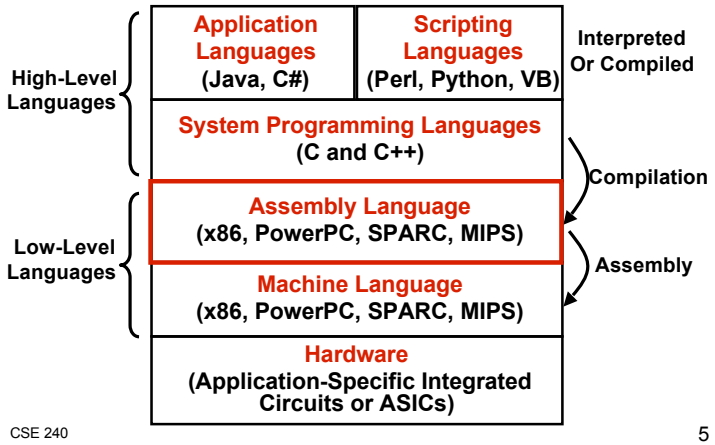
Aside: The Unix Command Line

Text-based approach to give commands
- Commonly used before graphical displays
- Many advantages even today

Examples
- `mkdir cse240hw8` make a directory
- `cd cse240hw8` change to the directory
- `ls` list contents of directory
- `cp /mnt/eniac/home1/c/cse240/project/hw/hw8/* .`
  - Copy files from one location to current dir (".")
- `emacs foo.c &` run the command "emacs" with input "foo.c"
- `gcc -o foo foo.c` compile foo.c (create program called "foo")

Unix eventually developed graphical UIs (GUIs)
- X-windows (long before Microsoft Windows)

Programming Levels



The Course Thus Far…

We did digital logic

- Bits are bits
- Ultimately, to understand a simple processor

We did assembly language programming

- Programming the "raw metal" of the computer
- Ultimately, to understand C programming

Starting today: we're doing C programming

- C is still common for systems programming
- You'll need it for the operating systems class (CSE380)
- Ultimately, for a deeper understanding of any language (Java)

Why High-Level Languages?

Easier than assembly. Why?

- Less primitive constructs
- Variables
- Type checking

Portability

- Write program once, run it on the LC-3 or Intel's x86

Disadvantages

- Slower and larger programs (in most cases)
- Can't manipulate low-level hardware
  - ➢ All operating systems have some assembly in them

Verdict: assembly coding is rare today

Our Challenge

99% of you already know either Java or C

- We're going to try to cover the basics quickly
- We'll spend more time on pointers & other C-specific nastiness

Created two decades apart

- C: 1970s - AT&T Bell Labs
- C++: 1980s - AT&T Bell Labs
- Java: 1990s - Sun Microsystems

Java and C/C++

- Syntactically similar (Java uses C syntax)
- C lacks many of Java's features
- Subtly different semantics

C is Similar To Java Without:

Objects
- No classes, objects, methods, or inheritance

Exceptions
- Check all error codes explicitly

Standard class library
- C has only a small standard library

Garbage collection
- C requires explicit memory allocate and free

Safety
- Java has strong type checking, checks array bounds
- In C, anything goes

Portability
- Source: C code is less portable (but better than assembly)
- Binary: C compiles to specific machine code

More C vs Java differences

C has a "preprocessor"
- A separate pre-pass over the code
- Performs replacements

Include vs Import
- Java has `import java.io.*;`
- C has: `#include <stdio.h>`
- #include is part of the preprocessor

Boolean type
- Java has an explicit boolean type
- C just uses an "int" as zero or non-zero
- C's lack of boolean causes all sorts of trouble

More differences as we go along…

What is C++?

C++ is an extension of C
- Backward compatible (good and bad)
- That is, all C programs are legal C++ programs

C++ adds many features to C
- Classes, objects, inheritance
- Templates for polymorphism
- A large, cumbersome class library (using templates)
- Exceptions (not actually implemented for a long time)
- More safety (though still unsafe)
- Operator and function overloading

Thus, many people uses it (to some extent)
- However, we're focusing on only C, not C++

Quotes on C/C++ vs Java

"C is to assembly language as Java is to C"
- Unknown

"With all due respect, saying Java is just a C++ subset is rather like saying that `Pride and Prejudice' is just a subset of the Encyclopedia Britanica. While it is true that one is shorter than the other, and that both have the same syntax, there are rather overwhelming differences."
- Sam Weber, on the ACM SIGSCE mailing list

"Java is C++ done right."
- Unknown

More quotes on C/C++

"The C programming language combines the power of assembly language with the ease-of-use of assembly language."

- Unknown

"It is my impression that it's possible to write good programs in C++, but nobody does."

- John Levine, moderator of comp.compilers

"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it, it blows your whole leg off."

- Bjarne Stroustrup, creator of C++

Compilation vs. Interpretation

Different ways of translating high-level languages

Interpretation

- Interpreter: program that executes program statements
  - ➢ Directly interprets program (portable but slow)
  - ➢ Limited optimization
- Easy to debug, make changes, view intermediate results
- Languages: BASIC, LISP, Perl, Python, Matlab

Compilation

- Compiler: translates statements into machine language
  - ➢ Creates executable program (non-portable, but fast)
  - ➢ Performs optimization over multiple statements
- Harder to debug, change requires recompilation
- Languages: C, C++, Fortran, Pascal

Hybrid

- Java, has features of both interpreted and compiled languages

Compilation vs. Interpretation

Consider the following algorithm:

```
• Get W from the keyboard.
• X = W + W
• Y = X + X
• Z = Y + Y
• Print Z to screen.
```

If interpreting, how many arithmetic operations occur?

If compiling, we can analyze the entire program and possibly reduce the number of operations.

- Can we simplify the above algorithm to use a single arithmetic operation?

Compiling a C Program

Entire mechanism is usually called the "compiler"

Preprocessor

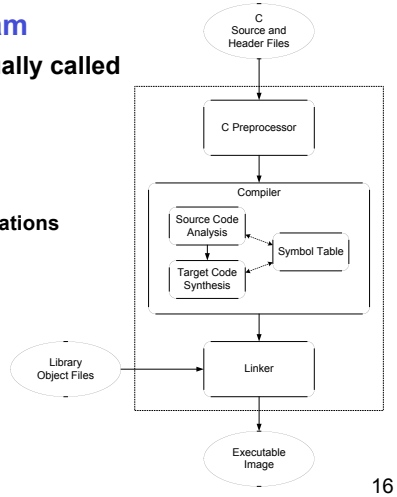
- Macro substitution
- Conditional compilation
- "Source-level" transformations
  - ➢ Output is still C

Compiler

- Generates object file
  - ➢ Machine instructions

Linker

- Combine object files (including libraries) into executable image



Compiler

Source Code Analysis

- "Front end"
- Parses programs to identify its pieces
  - Variables, expressions, statements, functions, etc.
- Depends on language (not on target machine)

Code Generation

- "Back end"
- Generates machine code from analyzed source
- May optimize machine code to make it run more efficiently
- Very dependent on target machine

Example Compiler: GCC

- The Free-Software Foundation's compiler
- Many front ends: C, C++, Fortran, Java
- Many back ends: Intel x86, PowerPC, SPARC, MIPS, Itanium

A Simple C Program

```
#include <stdio.h>
#define STOP 0

main()
{
  /* variable declarations */
  int counter;  /* an integer to hold count values */
  int startPoint; /* starting point for countdown */

  /* prompt user for input */
  printf("Enter a positive number: ");
  scanf("%d", &startPoint); /* read into startPoint */

  /* count down and print count */
  for (counter=startPoint; counter >= STOP; counter--) {
    printf("%d\n", counter);
  }
}
```


Preprocessor Directives

`#include <stdio.h>`

- Before compiling, copy contents of header file (stdio.h) into source code.
- Header files typically contain descriptions of functions and variables needed by the program.
  - no restrictions -- could be any C source code

`#define STOP 0`

- Before compiling, replace all instances of the string "STOP" with the string "0"
- Called a *macro*
- Used for values that won't change during execution, but might change if the program is reused. (Must recompile.)

Comments

Begins with /* and ends with */

- Can span multiple lines
- Comments are not recognized within a string
  - example: "my/*don't print this*/string"
    would be printed as: my/*don't print this*/string

Begins with // and ends with "end of line"

- Single-line comment
- Much like ";" in LC-3 assembly
- Introduced in C++, later back-ported to C

As before, use comments to help reader, not to confuse or to restate the obvious

`main` Function

Every C program must have a function called `main()`

- Starting point for every program
- Similar to Java's main method
  - `public static void main(String[] args)`

The code for the function lives within brackets:

```
void main()
{
  /* code goes here */
}
```


Variable Declarations

Variables are used as names for data items

Each variable has a *type*, tells the compiler:

- How the data is to be interpreted
- How much space it needs, etc.

```
int counter;
int startPoint;
```

C has similar primitive types as Java

- int, char, long, float, double
- More later

Input and Output

Variety of I/O functions in *C Standard Library*

- Must include `<stdio.h>` to use them

```
printf("%d\n", counter);
```

- String contains characters to print and formatting directions for variables
- This call says to print the variable `counter` as a decimal integer, followed by a linefeed (`\n`)

```
scanf("%d", &startPoint);
```

- String contains formatting directions for looking at input
- This call says to read a decimal integer and assign it to the variable `startPoint` (Don't worry about the `&` yet)

More About Output

Can print arbitrary expressions, not just variables

```
printf("%d\n", startPoint - counter);
```

Print multiple expressions with a single statement

```
printf("%d %d\n", counter,
       startPoint - counter);
```

Different formatting options:

- `%d` decimal integer
- `%x` hexadecimal integer
- `%c` ASCII character
- `%f` floating-point number

Examples

This code:

```
printf("%d is a prime number.\n", 43);
printf("43 plus 59 in decimal is %d.\n", 43+59);
printf("43 plus 59 in hex is %x.\n", 43+59);
printf("43 plus 59 as a character is %c.\n", 43+59);
```

produces this output:

```
43 is a prime number.
43 plus 59 in decimal is 102.
43 plus 59 in hex is 66.
43 plus 59 as a character is f.
```


Examples of Input

Many of the same formatting characters are available for user input

`scanf("%c", &nextChar);`
- reads a single character and stores it in nextChar

`scanf("%f", &radius);`
- reads a floating point number and stores it in radius

`scanf("%d %d", &length, &width);`
- reads two decimal integers (separated by whitespace), stores the first one in length and the second in width

Must use ampersand (&) for variables being modified
(Explained in Chapter 16.)

Compiling and Linking

Various compilers available
- cc, gcc
- includes preprocessor, compiler, and linker

Lots and lots of options!
- level of optimization, debugging
- preprocessor, linker options
- intermediate files --
  object (.o), assembler (.s), preprocessor (.i), etc.

Remaining Chapters

A more detailed look at many C features
- Variables and declarations
- Operators
- Control Structures
- Functions
- Pointers and Data Structures
- I/O

Emphasis on how C is converted to assembly language

Also see "C Reference" in Appendix D