CS 61C: Great Ideas in Computer Architecture
Introduction to C, Part I

Instructors:
Krste Asanovic
Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/F12

Agenda
- Review
- Compile vs. Interpret
- Python vs. Java vs. C
- Administrivia
- Quick Start Introduction to C
- Technology Break
- More C
- Summary

New-School Machine Structures
(It’s a bit more complicated!)

Review
- Request-Level Parallelism
  - High request volume, each largely independent of other
  - Use replication for better request throughput, availability
- MapReduce Data Parallelism
  - Map: Divide large data set into pieces for independent parallel processing
  - Reduce: Combine and process intermediate results to obtain final result

Big Idea #1: Levels of Representation/Interpretation

Introduction to C
“The Universal Assembly Language”

Smart Phone
Warehouse
Scale
Computer
Hardware
Harness
Parallelism & Achieve High Performance

Software

Hardware

Logic Gates

Input/Output

Computer

Memory

(Cache)

Functional Unit(s)

Core

Core

A

B

0

+1

=1

=2

=3

=0

0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111

- Class pre-req included classes teaching Java
- Java used in two labs and one project
- C used for everything else
Flash Card Language Poll!
Please raise card for first one of following you can say yes to

☐ I have programmed in C,C++,C#, or Objective-C
☐ I have programmed in Java
☐ I have programmed in FORTRAN, Cobol, Algol-68, Ada, Pascal, or Basic
☐ None of the above

Disclaimer
• You will not learn how to fully code in C in these lectures! Can only learn a language by using it!
• You’ll need your C reference for this course
  – K&R is a must-have
    • Check online for more sources
    • “Java in a Nutshell,” O’Reilly
  – Brian Harvey’s helpful transition notes
    • On CS61C class website: pages 3-19
    • http://inst.eecs.berkeley.edu/~cs61c/resources/HardyNotes.pdf
• Key C concepts: Pointers, Arrays, Implications for Memory management

Intro to C
• C is not a “very high level” language, nor a “big” one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages.
  — Kernighan and Ritchie
• Enabled first operating system not written in assembly language: UNIX - A portable OS!
• C and derivatives (C+/Obj-C/C#) still one of the most popular application programming languages after >40 years!

TIOBE Index of Language Popularity

<table>
<thead>
<tr>
<th>Position</th>
<th>Language</th>
<th>Percentage</th>
<th>2012</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>18.95%</td>
<td>+1.50%</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>Java</td>
<td>16.30%</td>
<td>-3.06%</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>Objective-C</td>
<td>9.44%</td>
<td>+0.69%</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>C++</td>
<td>9.39%</td>
<td>+0.96%</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>C#</td>
<td>6.99%</td>
<td>+0.56%</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>PHP</td>
<td>5.24%</td>
<td>-0.81%</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>(Visual) Basic</td>
<td>5.33%</td>
<td>+0.32%</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>Python</td>
<td>3.87%</td>
<td>+0.46%</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>Perl</td>
<td>2.72%</td>
<td>-0.04%</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>Ruby</td>
<td>1.69%</td>
<td>+0.38%</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>JavaScript</td>
<td>1.36%</td>
<td>-0.19%</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>Delphi/Object Pascal</td>
<td>1.01%</td>
<td>-0.06%</td>
<td>A</td>
</tr>
</tbody>
</table>

Basic C Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler</td>
<td>Creates useable programs from C source</td>
</tr>
<tr>
<td>Typed variables</td>
<td>Kind of data that a variable contains</td>
</tr>
<tr>
<td>Typed functions</td>
<td>The kind of data returned from a function</td>
</tr>
<tr>
<td>Header files (.h)</td>
<td>Declare functions and variables in a separate file</td>
</tr>
<tr>
<td>Structs</td>
<td>Groups of related values</td>
</tr>
<tr>
<td>Enums</td>
<td>Lists of predefined values</td>
</tr>
<tr>
<td>Pointers</td>
<td>Aliases to other variables</td>
</tr>
</tbody>
</table>

Integers: Python vs. Java vs. C

<table>
<thead>
<tr>
<th>Language</th>
<th>sizeof(int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>&gt;=32 bits (plain ints), infinite (long ints)</td>
</tr>
<tr>
<td>Java</td>
<td>32 bits</td>
</tr>
<tr>
<td>C</td>
<td>Depends on computer; 16 or 32 or 64</td>
</tr>
</tbody>
</table>

• C: int should be integer type that target processor is most efficient working with
• Only guarantee: sizeof(long) ≥ sizeof(int) ≥ sizeof(short)
  – All could be 64 bits
### Compilation: Overview

- **C compilers** map C programs into architecture-specific machine code (string of 1s and 0s)
  - Unlike Java, which converts to architecture-independent bytecode
  - Unlike Python environments, which interpret the code
  - These differ mainly in exactly when your program is converted to low-level machine instructions (“levels of interpretation”)
  - For C, generally a two part process of compiling .c files to .o files, then linking the .o files into executables;
  - Assembling is also done (but is hidden, i.e., done automatically, by default); we’ll talk about that later

### Compilation: Advantages

- Excellent run-time performance: generally much faster than Scheme or Java for comparable code (because it optimizes for a given architecture)
- Fair compilation time: enhancements in compilation procedure (Makefiles) allow only modified files to be recompiled
- Why C?: we can write programs that allow us to exploit underlying features of the architecture — memory management, special instructions, parallelism

### Compilation: Disadvantages

- Compiled files, including the executable, are architecture-specific, depending on processor type and the operating system
- Executable must be rebuilt on each new system
  - i.e., “porting your code” to a new architecture
- “Change → Compile → Run [repeat]” iteration cycle can be slow, during the development cycle

### Typed Variables in C

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>int, unsigned int</td>
<td>integer numbers, including negatives</td>
<td>0, 46, 900</td>
</tr>
<tr>
<td>float</td>
<td>floating-point numbers</td>
<td>0.0, 1.618, -1.4</td>
</tr>
<tr>
<td>char</td>
<td>single text character or symbol</td>
<td>‘a’, ‘b’, ‘f’</td>
</tr>
<tr>
<td>double, long</td>
<td>greater precision/big FP number/larger signed integer</td>
<td>10E100, 6,000,000,000</td>
</tr>
</tbody>
</table>

### C vs. Java

<table>
<thead>
<tr>
<th>Type of Language</th>
<th>C</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Unit</td>
<td>Function Oriented</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>Compilation</td>
<td>gcc hello.c creates machine language code</td>
<td>javac Hello.java creates Java virtual machine language bytecode</td>
</tr>
<tr>
<td>Execution</td>
<td>a.out loads and executes program</td>
<td>java Hello interprets bytecode</td>
</tr>
</tbody>
</table>

| Storage | Manual (malloc, free) | Automatic (garbage collection) |

### C vs. Java

<table>
<thead>
<tr>
<th>Comments</th>
<th>/* */</th>
<th>/* <em>/ or /</em> ... */ end of line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>const, #define</td>
<td>final</td>
</tr>
<tr>
<td>Preprocessor</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Variable declaration</td>
<td>At beginning of a block</td>
<td>Before you use it</td>
</tr>
<tr>
<td>Variable naming conventions</td>
<td>sum_of_squares, sumosquares</td>
<td>sumOfSquares</td>
</tr>
<tr>
<td>Accessing a library</td>
<td>#include &lt;stdio.h&gt;</td>
<td>import java.io.File;</td>
</tr>
</tbody>
</table>

### Examples

- int variable1 = 2;
- float variable2 = 1.618;
- char variable3 = ‘A’;

### Examples

- Must declare the type of data a variable will hold
  - Types can’t change

### Examples

- C vs. Java

<table>
<thead>
<tr>
<th>C</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments</td>
<td>/* */</td>
</tr>
<tr>
<td>Constants</td>
<td>const, #define</td>
</tr>
<tr>
<td>Preprocessor</td>
<td>Yes</td>
</tr>
<tr>
<td>Variable declaration</td>
<td>At beginning of a block</td>
</tr>
<tr>
<td>Variable naming conventions</td>
<td>sum_of_squares, sumosquares</td>
</tr>
<tr>
<td>Accessing a library</td>
<td>#include &lt;stdio.h&gt;</td>
</tr>
</tbody>
</table>

### Examples

- int variable1 = 2;
- float variable2 = 1.618;
- char variable3 = ‘A’;
Typed Functions in C

```c
int number_of_people() {
    return 3;
}
float dollars_and_cents() {
    return 10.33;
}
char first_letter() {
    return 'A';
}
```

- You have to declare the type of data you plan to return from a function.
- Return type can be any C variable type, and is placed to the left of the function name.
- You can also specify the return type as `void` – just think of this as saying that no value will be returned.
- Also necessary to declare types for values passed into a function.
- Variables and functions MUST be declared before they are used.

Structs in C

```c
typedef struct {
    int length_in_seconds;
    int year_recorded;
} Song;
```

- Structs are structured groups of variables, e.g.,
- You can have a constant version of any of the standard C variable types
- Enums: a group of related integer constants used to parameterize libraries:
  ```c
  enum card_suit {Clubs, Diamonds, Hearts, Spades};
  ```

Consts and Enums in C

- Constant is assigned a value once in the declaration; value can’t change during entire execution of program.
- You can have a constant version of any of the standard C variable types.
- Enums: a group of related integer constants used to parameterize libraries:
  ```c
  enum card_suit {Clubs, Diamonds, Hearts, Spades};
  ```

Question: Which statement is TRUE regarding C and Java?

- short, int, and long are in both languages and they have the same meaning
- As Java was derived from C, it has the same names of data types
- C programs use compilers to produce executable code but Java does not
- C has a preprocessor that allows conditional compilation, but Java does not

Agenda

- Review
- Compile vs. Interpret
- Scheme vs. Java vs. C
- Administrivia
- Quick Start Introduction to C
- Technology Break
- More C
- Summary

Administrivia

- CS61c is relentless!
  - Next week: Lab #2, HW #2
  - Lab #2, Amazon EC2
  - HW #2 will soon be posted
- Monday is Labor Day Holiday – no lecture!
- Wonderful to see the valuable discussion and help going on in Piazza!
A First C Program: Hello World

Original C:

```c
main()
{
  printf("Hello World\n");
  int main(void)
  { 
    printf("Hello World\n");
    return (0);
  }
}
```

ANSI Standard C:

```c
#include <stdio.h>

main()
{
  printf("Hello World\n");
  int main(void)
  { 
    printf("Hello World\n");
    return (0);
  }
}
```

C Syntax: main

- When a program starts,
- 1st runs initialization code to set up process for your program
- Then calls your procedure named main()

- To get arguments to the main function, use:
  - `int main (int argc, char *argv[])`

- What does this mean?
  - `argc` contains the number of strings on the command line (the executable counts as one, plus one for each argument).
  Here, `argc` is 2:
    - `argc` is an array containing the arguments
    - Arrays are pointers to strings (more on pointers later)

Example

- `foo hello 87`
- `argc = 3 /* number arguments */`
- `argv[0] = "foo",
  argv[1] = "hello",
  argv[2] = "87"

---

A Second C Program: Compute Table of Sines

```c
#include <stdio.h>

int main(void)
{
  int angle_degree;
  double angle_rad, pi, value;
  /* Print a header */
  printf("Compute Table of Sines\n");
  /* increment the loop index */
  value = pi = angle_degree / 180.0;
  /* scan over angle
   * for all the entries in the table */
  printf("Value of PI = \%lf\n", pi);
}
```

```c
/* or just use pi = M_PI, where */
/* Print a header */
int main(void)
{
  int angle_degree;
  double angle_rad, pi, value;
  /* Print a header */
  printf("Compute Table of Sines\n");
  /* scan over angle
   * for all the entries in the table */
  printf("Value of PI = \%lf\n", pi);
}
```

C Syntax: Variable Declarations

- Similar to Java, but with a few minor but important differences
- All variable declarations must appear before they are used (e.g., at the beginning of the block)
- A variable may be initialized in its declaration; if not, it holds garbage!
- Examples of declarations:
  - Correct:
    ```c
    int a = 0, b = 10;
    ```
  - Incorrect:
    ```c
    for (int i = 0; i < 10; i++)
    ```
C Syntax : Control Flow (1/2)

• Within a function, remarkably close to Java constructs (shows Java's legacy) in terms of control flow
  – if-else
    – if (expression) statement
    – if (expression) statement1
      else statement2
  – while
    – while (expression)
      statement
      do
      statement
      while (expression);

C Syntax : Control Flow (2/2)

– for
  – for (initialize; check; update)
    statement
– switch
  – switch (expression){
    case const1:    statements
    case const2:    statements
    default:        statements
  }
  – break

C Syntax: True or False

• What evaluates to FALSE in C?
  – 0 (integer)
  – NULL (a special kind of pointer: more on this later)
  – No explicit Boolean type
• What evaluates to TRUE in C?
  – Anything that isn’t false is true
  – Same idea as in Python: only 0s or empty sequences are false, anything else is true!

C and Java operators nearly identical

• arithmetic: +, -, *, /, %
• assignment: =
• augmented assignment: +=, -=, *=, %, &, | =, ^=, <<=, >>=
• bitwise logic: ~, &, |, ^
• bitwise shifts: <<, >>
• boolean logic: !, &&, ||
• equality testing: ==, !=
• subexpression grouping: ()
• order relations: <, <=, >, >=
• increment and decrement: ++ and --
• member selection: ., ->
• conditional evaluation: ?: 

And In Conclusion, ...

• All data C is an efficient compiled language, widely used for systems programming (operating systems) and application development
• C successors (C++, Objective-C, C#) are also popular
• Java syntax based on C, due to C’s popularity
• BUT C can be more difficult to use than more modern programming languages