COS 217: Introduction to Programming Systems

Welcome!
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Introductions

Instructor-of-Record
• Aarti Gupta, Ph.D.
  • aartig@cs.princeton.edu

Lead Preceptors
• Robert Dondero, Ph.D.
  • rdondero@cs.princeton.edu
• Iasonas Petras, Ph.D.
  • ipetras@cs.princeton.edu

Graduate Student Preceptors
• Hao (Frank) Wu
  • haow@princeton.edu
• Haoyu (Harris) Zhang
  • haoyuz@princeton.edu
Introductions

Graders
(in alphabetical order)

• Dorothy Chen
  • dschen@princeton.edu

• Annie Chu
  • anyuanc@princeton.edu

• Matthew Colen
  • mcolen@princeton.edu

• Naphat Sanguansin
  • naphats@princeton.edu
Agenda

Course overview
- Introductions
- **Course goals**
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Goal 1: “Programming in the large”

- Help you learn how to compose large computer programs

Topics

- Modularity/abstraction, information hiding, resource management, error handling
- Testing and debugging your code
- Performance improvement
- Tool support
Goal 2: “Under the Hood”

Goal 2: “Look under the hood”
• Help you learn what happens “under the hood” of computer systems

Downward tours

- C Language
- Assembly Language
- Machine Language
- Application Program
- Operating System
- Hardware

Language levels tour
Service levels tour
Goals: Summary

Help you to become a...

*Power Programmer!!!*
Question: Why C instead of Java?

Answer: C supports Goal 2 better
- Languages-level tour
  - Closely related to assembly language
- Services-level tour
  - Linux (our OS) is written in C

Answer: C supports Goal 1 better
- C is a lower-level language
  - Provides more opportunities to create abstractions
- C has some flaws
  - Motivate discussions of software engineering principles
Goals: Why Linux?

**Question:** Why Linux instead of Microsoft Windows?

**Answer 1:** Linux is good for education and research
- Open source, well-specified

**Answer 2:** Linux (with GNU) is good for programming
- Variant of Unix, GNU provides rich open-source programming environment
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Lectures

• Describe material at conceptual (high) level
• Slides available via course website
• Suggestion: Bring hard copy of slides

Lecture etiquette

• Please don’t use electronic devices during lectures
Precepts

• Describe material at concrete (low) level
• Support your work on assignments
• Hard copy handouts distributed during precepts
• Handouts available via course website

Precept etiquette

• Attend your precept
• Use SCORE to move to another precept
  • Trouble => See Colleen Kenny-McGinley (CS Bldg 210)
  • But Colleen can’t move you into a full precept
• Must miss your precept => inform preceptors & attend another

Precepts begin today!
Website

Website

• Access from http://www.cs.princeton.edu
  • Academics → Course Schedule → COS 217
  • Home page, schedule page, assignment page, policies page
Piazza

• https://piazza.com/class#spr2015/cos217
• Instructions provided in first precept

Piazza etiquette

• Study provided material before posting question
  • Lecture slides, precept handouts, required readings
  • Read all (recent) Piazza threads before posting question
• Don’t show your code!!!
  • See course policies
Books

**The Practice of Programming** *(recommended)*
- Kernighan & Pike
- “Programming in the large”

**Computer Systems: A Programmer’s Perspective (Second Edition)** *(recommended)*
- Bryant & O'Hallaron
- “Under the hood”

**C Programming: A Modern Approach (Second Edition)** *(required)*
- King
- C programming language and standard libraries
Manuals (for reference only, available online)

- *IA32 Intel Architecture Software Developer’s Manual, Volumes 1-3*
- *Tool Interface Standard & Executable and Linking Format*
- *Intel 64 and IA-32 Architectures Optimization Reference Manual*
- *Using as, the GNU Assembler*

See also

- Linux `man` command
  - `man` is short for “manual”
  - For more help, type `man man`
Programming Environment

Server

- nobel.princeton.edu
- Linux
- GNU
- Your Pgm
- compton
- davisson

Client

Local Computer

- SSH

Your computer or cluster computer; on-campus or off-campus
Agenda

Course overview
• Introductions
• Course goals
• Resources
• **Grading**
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Grading

<table>
<thead>
<tr>
<th>Course Component</th>
<th>Percentage of Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments *</td>
<td>50</td>
</tr>
<tr>
<td>Midterm Exam **</td>
<td>15</td>
</tr>
<tr>
<td>Final Exam **</td>
<td>25</td>
</tr>
<tr>
<td>Subjective ***</td>
<td>10</td>
</tr>
</tbody>
</table>

* Final assignment counts double; penalties for lateness

** Closed book, closed notes, no electronic devices

*** Did your involvement benefit the course as a whole?
  • Lecture and precept attendance and participation counts
Programming Assignments

Programming assignments
• A “de-comment” program
• A string module
• A symbol table module
• IA-32 assembly language programs
• A buffer overrun attack (partner from your precept)
• A heap manager module (partner from your precept)
• A Unix shell

First assignment is available now
• Due on Sunday, February 15 (at 9:00 PM)

Start early!!!
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Study the course “Policies” web page!

Especially the assignment collaboration policies

- Violations often involve trial by Committee on Discipline
- Typical course-level penalty is F for course
- Typical University-level penalty is suspension from University for 1 academic year
Assignment Related Policies

Some highlights:

• You may not reveal any of your assignment solutions (products, descriptions of products, design decisions) on Piazza.

• **Getting help**
  • use only authorized sources of information
  • may consult with other people only via the course's Piazza account or via interactions that might legitimately appear on the course's Piazza account
  • must declare your sources in your readme file for the assignment

• **Giving help**
  • only via the course's Piazza account or interactions that might legitimately appear on the course's Piazza account
  • may not share your assignment solutions with anyone, ever, in any form

Ask the instructor-of-record for clarifications

• Only the instructor-of-record can waive any policies (not verbally)
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- **Schedule**

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
## Course Schedule

<table>
<thead>
<tr>
<th>Weeks</th>
<th>Lectures</th>
<th>Precepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>Number Systems C (conceptual)</td>
<td>Linux/GNU C (pragmatic)</td>
</tr>
<tr>
<td>3-6</td>
<td>“Programming in the Large”</td>
<td>Advanced C</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Midterm Exam</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Recess</td>
</tr>
<tr>
<td>8-13</td>
<td>“Under the Hood” (conceptual)</td>
<td>“Under the Hood” (programming asgts)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reading Period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Exam</td>
</tr>
</tbody>
</table>
Any questions so far?
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- **History of C**
- Building and running C programs
- Characteristics of C
- C details (if time)
“C is quirky, flawed, and an enormous success. While accidents of history surely helped, it evidently satisfied a need for a system implementation language efficient enough to displace assembly language, yet sufficiently abstract and fluent to describe algorithms and interactions in a wide variety of environments.”
Java vs. C: History

<table>
<thead>
<tr>
<th>Year</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>BCPL</td>
</tr>
<tr>
<td>1970</td>
<td>B</td>
</tr>
<tr>
<td>1972</td>
<td>C</td>
</tr>
<tr>
<td>1978</td>
<td>K&amp;R C</td>
</tr>
<tr>
<td>1989</td>
<td>ANSI C89</td>
</tr>
<tr>
<td></td>
<td>ISO C90</td>
</tr>
<tr>
<td>1999</td>
<td>ISO C99</td>
</tr>
<tr>
<td></td>
<td>ANSI C99</td>
</tr>
<tr>
<td>2011</td>
<td>ISO C11</td>
</tr>
</tbody>
</table>

We will use

Not (yet?) popular; our compiler supports only partially
Java vs. C: Design Goals

<table>
<thead>
<tr>
<th>Java Design Goals</th>
<th>C Design Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language of the Internet</td>
<td>Support development of Unix</td>
</tr>
<tr>
<td>High-level; insulated from hardware and OS</td>
<td>Low-level; close to HW and OS</td>
</tr>
<tr>
<td>Good for application-level programming</td>
<td>Good for system-level programming</td>
</tr>
<tr>
<td>Support object-oriented programming</td>
<td>Support structured programming</td>
</tr>
<tr>
<td>Look like C!</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
$ javac MyPgm.java

Java compiler (machine lang code)

MyPgm.java (Java code) → javac → MyPgm.class (bytecode)
Running Java Programs

$ java MyPgm

HW (nobel) → OS (Linux) → Java interpreter (Java virtual machine) (machine lang code)

MyPgm.class (bytecode) → java → data

data → java → data
Building C Programs

$ gcc217 mypgm.c –o mypgm

C “compiler driver” (machine lang code)
$ mypgm

mypgm
(machine lang code)

HW (nobel)

OS (Linux)

data

mypgm

data
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
# Java vs. C: Portability

<table>
<thead>
<tr>
<th>Program</th>
<th>Code Type</th>
<th>Portable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPgm.java</td>
<td>Java source code</td>
<td>Yes</td>
</tr>
<tr>
<td>mypgm.c</td>
<td>C source code</td>
<td>Mostly</td>
</tr>
<tr>
<td>MyPgm.class</td>
<td>Bytecode</td>
<td>Yes</td>
</tr>
<tr>
<td>mypgm</td>
<td>Machine lang code</td>
<td>No</td>
</tr>
<tr>
<td>javac (Java compiler)</td>
<td>Machine lang code</td>
<td>No</td>
</tr>
<tr>
<td>java (Java interpreter)</td>
<td>Machine lang code</td>
<td>No</td>
</tr>
<tr>
<td>gcc217 (C compiler driver)</td>
<td>Machine lang code</td>
<td>No</td>
</tr>
</tbody>
</table>

Java programs are more portable.
Java vs. C: Efficiency

Java programs run on “virtual” machine which runs on “real” machine

C programs run on “real” machine

C programs are faster
Java vs. C: Safety

Java programs run on “virtual” machine defined by interpreter; can provide safe environment (e.g. array bounds checks)

C programs run directly on “real” machine

Java programs are safer
Java vs. C: Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Efficiency</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Safety</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
If this is Java...
Java vs. C: Characteristics

Then this is C
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Java vs. C: Details

Remaining slides provide some details

Use for future reference
# Java vs. C: Details

<table>
<thead>
<tr>
<th>Overall Program Structure</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello.java: public class Hello { public static void main (String[] args) { System.out.println(&quot;hello, world&quot;); } }</td>
<td>hello.c: #include &lt;stdio.h&gt; int main(void) { printf(&quot;hello, world\n&quot;); return 0; }</td>
<td></td>
</tr>
</tbody>
</table>

| Building | $ javac Hello.java | $ gcc217 hello.c –o hello |
| Running  | $ java Hello hello, world $ | $ hello hello, world $ |
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Character type</strong></td>
<td>char // 16-bit Unicode</td>
<td>char /* 8 bits */</td>
</tr>
<tr>
<td></td>
<td><strong>Integral types</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>byte // 8 bits</td>
<td>(unsigned) char</td>
</tr>
<tr>
<td></td>
<td>short // 16 bits</td>
<td>(unsigned) short</td>
</tr>
<tr>
<td></td>
<td>int // 32 bits</td>
<td>(unsigned) int</td>
</tr>
<tr>
<td></td>
<td>long // 64 bits</td>
<td>(unsigned) long</td>
</tr>
<tr>
<td></td>
<td><strong>Floating point types</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>float // 32 bits</td>
<td>float</td>
</tr>
<tr>
<td></td>
<td>double // 64 bits</td>
<td>double</td>
</tr>
<tr>
<td></td>
<td><strong>Logical type</strong></td>
<td>long double</td>
</tr>
<tr>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>/* no equivalent */</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/* use integral type */</td>
</tr>
<tr>
<td></td>
<td><strong>Generic pointer type</strong></td>
<td>void*</td>
</tr>
<tr>
<td></td>
<td>// no equivalent</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Constants</strong></td>
<td>#define MAX 1000</td>
</tr>
<tr>
<td></td>
<td>final int MAX = 1000;</td>
<td>const int MAX = 1000;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enum {MAX = 1000};</td>
</tr>
</tbody>
</table>
# Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arrays</strong></td>
<td><code>int [] a = new int [10];</code></td>
<td><code>int a[10];</code></td>
</tr>
<tr>
<td></td>
<td><code>float [][] b = new float [5][20];</code></td>
<td><code>float b[5][20];</code></td>
</tr>
<tr>
<td><strong>Array bound checking</strong></td>
<td>// run-time check</td>
<td>/* no run-time check */</td>
</tr>
<tr>
<td><strong>Pointer type</strong></td>
<td><code>// Object reference is an implicit pointer</code></td>
<td><code>int *p;</code></td>
</tr>
<tr>
<td><strong>Record type</strong></td>
<td><code>class Mine</code></td>
<td><code>struct Mine</code></td>
</tr>
<tr>
<td></td>
<td><code>{ int x; float y; }</code></td>
<td><code>{ int x; float y; }</code></td>
</tr>
</tbody>
</table>

```java
class Mine
{ int x;
 float y;
}
```
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strings</strong></td>
<td>String s1 = &quot;Hello&quot;;</td>
<td>char *s1 = &quot;Hello&quot;;</td>
</tr>
<tr>
<td></td>
<td>String s2 = new</td>
<td>char s2[6];</td>
</tr>
<tr>
<td></td>
<td>String(&quot;hello&quot;);</td>
<td>strcpy(s2, &quot;hello&quot;);</td>
</tr>
<tr>
<td>String</td>
<td>s1 + s2</td>
<td>#include &lt;string.h&gt;</td>
</tr>
<tr>
<td>concatenation</td>
<td>s1 += s2</td>
<td>strcat(s1, s2);</td>
</tr>
<tr>
<td><strong>Logical ops</strong></td>
<td>&amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td><strong>Relational ops</strong></td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td><strong>Arithmetic ops</strong></td>
<td>+, -, *, /, %, unary -</td>
<td>+, -, *, /, %, unary -</td>
</tr>
<tr>
<td><strong>Bitwise ops</strong></td>
<td>&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;, &amp;,</td>
<td>, ^</td>
</tr>
<tr>
<td><strong>Assignment ops</strong></td>
<td>=, *=, /=, +=, -=, &lt;&lt;=, &gt;&gt;=, &gt;&gt;&gt;=, &gt;&gt;&gt;&gt;&gt;, &amp;=, ^=,</td>
<td>=, %=</td>
</tr>
</tbody>
</table>

**Essentially the same in the two languages**
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th><strong>Java</strong></th>
<th><strong>C</strong></th>
</tr>
</thead>
</table>
| if stmt ** | `if (i < 0)  
statement1;  
else  
statement2;` | `if (i < 0)  
statement1;  
else  
statement2;` |
| switch stmt ** | `switch (i)  
{  case 1:  
    ...  
    break;  
    case 2:  
    ...  
    break;  
    default:  
    ...  
}` | `switch (i)  
{  case 1:  
    ...  
    break;  
    case 2:  
    ...  
    break;  
    default:  
    ...  
}` |
| goto stmt | `// no equivalent` | `goto someLabel;` |

** Essentially the same in the two languages**
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for stmt</strong></td>
<td>for (int i=0; i&lt;10; i++) statement;</td>
<td>int i; for (i=0; i&lt;10; i++) statement;</td>
</tr>
<tr>
<td><strong>while stmt</strong></td>
<td>while (i &lt; 0) statement;</td>
<td>while (i &lt; 0) statement;</td>
</tr>
<tr>
<td><strong>do-while stmt</strong></td>
<td>do statement; while (i &lt; 0)</td>
<td>do statement; while (i &lt; 0);</td>
</tr>
<tr>
<td><strong>continue stmt</strong></td>
<td>continue;</td>
<td>continue;</td>
</tr>
<tr>
<td><strong>labeled continue stmt</strong></td>
<td>continue someLabel;</td>
<td>/* no equivalent */</td>
</tr>
<tr>
<td><strong>break stmt</strong></td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td><strong>labeled break stmt</strong></td>
<td>break someLabel;</td>
<td>/* no equivalent */</td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages*
## Java vs. C: Details

<table>
<thead>
<tr>
<th><strong>Java</strong></th>
<th><strong>C</strong></th>
</tr>
</thead>
</table>
| **return stmt** **return 5;**
| **return;** | **return 5;**
| **return;** | **return;** |
| **Compound stmt** **{**
| **alias block)** **{**
| **statement1;** | **statement1;**
| **statement2;** | **statement2;**
| **}** | **}** |
| **Exceptions** **throw, try-catch-finally** | /* no equivalent */ |
| **Comments** /* comment */
| // another kind | /* comment */ |
| **Method / function call** **f(x, y, z);**
| **someObject.f(x, y, z);** | **f(x, y, z);**
| **SomeClass.f(x, y, z);** |

** Essentially the same in the two languages **
Example C Program

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const double KMETERS_PER_MILE = 1.609;
    int miles;
    double kMeters;

    printf("miles: ");
    if (scanf("%d", &miles) != 1)
    {
        fprintf(stderr, "Error: Expected a number.\n");
        exit(EXIT_FAILURE);
    }

    kMeters = (double)miles * KMETERS_PER_MILE;
    printf("%d miles is %f kilometers.\n",
            miles, kMeters);
    return 0;
}
```
Summary

Course overview

• Introductions
• Course goals
  • Goal 1: Learn “programming in the large”
  • Goal 2: Look “under the hood”
  • Use of C and Linux supports both goals
• Resources
  • Lectures, precepts, programming environment, Piazza, textbooks
  • Course website: access via http://www.cs.princeton.edu
• Grading
• Policies
• Schedule
Summary

Getting started with C

- History of C
- Building and running C programs
- Characteristics of C
- Details of C
  - Java and C are similar
  - Knowing Java gives you a head start at learning C
Getting Started

Check out the course website soon
  • Study “Policies” page
  • First assignment is available

Establish a reasonable computing environment soon
  • Instructions given in first precept