COS 217: Introduction to Programming Systems
Jennifer Rexford

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
- Jen Rexford, Ph.D.
  - jrex@cs.princeton.edu

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

Introductions: Other Preceptors
- Robert MacDavid
- Reid Oda
- Sergiy Popovych
- Huilian (Sophie) Qiu
- Laura Roberts
- Katherine Wolf

Goal 1: “Programming in the Large”
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, debugging, 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

<table>
<thead>
<tr>
<th>C Language</th>
<th>Application Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly Language</td>
<td>Operating System</td>
</tr>
<tr>
<td>Machine Language</td>
<td>Hardware</td>
</tr>
</tbody>
</table>

Goals: Summary

Help you to become a...

Power Programmer!!!

Goals: Why C?

Question: Why C instead of Java?

Answer 1: C supports Goal 2 better
Answer 2: C supports Goal 1 better

Goals: Why Linux?

Question: Why Linux instead of Microsoft Windows?

Answer 1: Linux is good for education and research
Answer 2: Linux (with GNU) is good for programming

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

Lectures
- Describe material at conceptual level
- Slides available via course website
- Suggestion: Bring hard copy of slides

Lecture etiquette
- Please don’t use electronic devices during lectures
Precepts

Precepts
• Describe material at physical (low) level
• Support your work on assignments
• 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 Monday September 21

Website

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

Books

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

• Bryant & O’Hallaron
  • “Under the hood”

• King
  • C programming language and standard libraries

Piazza

Piazza
• http://piazza.com/class#fall2015/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

Programming Environment

Server
FC010 Cluster
Linux
GNU
Your Pgm
0010-labpc-01
0010-labpc-21

Client
Your Computer
On-campus or off-campus

Manuals

Manuals (for reference only, available online)
• Intel 64 and IA-32 Architectures Software Developer’s Manual, Volumes 1-3
• Intel 64 and IA-32 Architectures Optimization Reference Manual
• Using as, the GNU Assembler

See also
• Linux man command
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
• 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
Start early!!!

Policies

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: To help you compose an assignment solution you may 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, and must declare your sources in your readme file for the assignment.
• Giving help: You may help other students with assignments only via the course’s Piazza account or interactions that might legitimately appear on the course’s Piazza account, and you 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 (and 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>&quot;Programming in the Large&quot;</td>
<td>Advanced C</td>
</tr>
<tr>
<td>6</td>
<td>Midterm Exam</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Recess</td>
<td></td>
</tr>
<tr>
<td>8-13</td>
<td>&quot;Under the Hood&quot; (conceptual)</td>
<td>&quot;Under the Hood&quot; (programming asgts)</td>
</tr>
<tr>
<td></td>
<td>Reading Period</td>
<td>Final Exam</td>
</tr>
</tbody>
</table>

Any questions?

The C Programming Language

Who? Dennis Ritchie
When? ~1972
Where? Bell Labs
Why? Compose the Unix OS

Java vs. C: History

Not (yet?) popular; our compiler supports only partially

We will use ISO C11
## 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>Compose 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)

## Building Java Programs

```bash
$ javac MyPgm.java
```

**Diagram:**

```
<table>
<thead>
<tr>
<th>HW (fc010)</th>
<th>OS (Linux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPgm.java</td>
<td>javac</td>
</tr>
</tbody>
</table>
```

**MyPgm.java** (Java code)

**MyPgm.class** (bytecode)

## Running Java Programs

```bash
$ java MyPgm
```

**Diagram:**

```
<table>
<thead>
<tr>
<th>HW (fc010)</th>
<th>OS (Linux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td></td>
</tr>
</tbody>
</table>
```

**MyPgm.class** (bytecode)

## Building C Programs

```bash
$ gcc217 mypgm.c -o mypgm
```

**Diagram:**

```
<table>
<thead>
<tr>
<th>HW (fc010)</th>
<th>OS (Linux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mypgm.c</td>
<td>gcc217</td>
</tr>
</tbody>
</table>
```

**mypgm.c** (C code)

**mypgm** (machine lang code)

## Running C Programs

```bash
$ mypgm
```

**Diagram:**

```
<table>
<thead>
<tr>
<th>HW (fc010)</th>
<th>OS (Linux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mypgm</td>
<td></td>
</tr>
</tbody>
</table>
```

**mypgm** (machine lang code)
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>

Conclusion: 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

Conclusion: 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

Conclusion: 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

Java vs. C: Details

Remaining slides provide some details

Use for future reference

Slides covered now, as time allows...

Java vs. C: Details

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

Java vs. C: Details

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

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strings</strong></td>
<td></td>
</tr>
<tr>
<td><code>String s1 = &quot;Hello&quot;;</code></td>
<td><code>char *s1 = &quot;Hello&quot;;</code></td>
</tr>
<tr>
<td><code>String s2 = new String(&quot;hello&quot;);</code></td>
<td><code>strcpy(s2, &quot;hello&quot;);</code></td>
</tr>
<tr>
<td><strong>Logical ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>&amp;&amp;</code>, `</td>
<td></td>
</tr>
<tr>
<td><strong>Relational ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>=</code>, <code>!=</code>, <code>&gt;</code>, <code>&lt;</code>, <code>&gt;=</code>, <code>&lt;=</code></td>
<td><code>=</code>, <code>!=</code>, <code>&gt;</code>, <code>&lt;</code>, <code>&gt;=</code>, <code>&lt;=</code></td>
</tr>
<tr>
<td><strong>Arithmetic ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>+</code>, <code>-</code>, <code>*</code>, <code>/</code>, <code>%</code>, unary <code>-</code></td>
<td><code>+</code>, <code>-</code>, <code>*</code>, <code>/</code>, <code>%</code>, unary <code>-</code></td>
</tr>
<tr>
<td><strong>Bitwise ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>&gt;&gt;, </code>&lt;&lt;<code>, </code>&gt;&gt;&gt;<code>, </code>&amp;<code>, </code></td>
<td><code>, </code>^`</td>
</tr>
<tr>
<td><strong>Assignment ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>=</code>, <code>*=</code> , <code>/=</code> , <code>+=</code>, <code>-=</code>, <code>&lt;&lt;=</code>, <code>&gt;&gt;=</code>, <code>&gt;&gt;&gt;=</code> , <code>&amp;=</code>, <code>^=</code> , `</td>
<td>=`</td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages

### Java vs. C: Details

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

* Essentially the same in the two languages

### Example C Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.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 course website soon
- Study "Policies" page
- First assignment is available

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