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)
Goals: Why C?

**Question:** Why C instead of Java?

**Semi-answer:** C and Java are both very widely used in software development; they use different approaches to memory management; good to understand both approaches

**Answer:** C is the primary language for low-level systems (operating systems, devices)

Goals: Why Linux?

**Question:** Why Linux instead of MS Windows or MacOs?

**Answer 1:** Linux is the most widely used platform for professional software development

**Answers 2,3:** Linux (with GNU) has excellent open-source tool suites, doesn’t lock you in to a single proprietary vendor; Linux/GNU is elegant and easily scriptable. (These help explain Answer 1)

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

Lecture etiquette

- Let’s start on time, please
- Please don’t use electronic devices during lectures
- If you must phiddle with your phone or laptop, sit in the back row where you won’t distract other students
Precepts

• Describe material at the "practical" 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/courses/schedule
  • Princeton CS → Courses → Course Schedule → COS 217
• Home page, schedule page, assignment page, policies page

Piazza

Piazza
• http://piazza.com/class#spring2017/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"

• Bryant & O’Hallaron
  • "Under the hood"

• King
• C programming language and standard libraries

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
Programming Environment

Server

CourseLab Cluster

Linux
GNU
Your
Pgm
courselab01
courselab02

Client

Your Computer

SSH

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

Course Component | Percentage of Grade
--- | ---
Assignments * | 50
Midterm Exam ** | 15
Final Exam ** | 25
Subjective *** | 10

* Final assignment counts double; penalties for lateness
** Closed book, closed notes, no electronic devices
*** Did your involvement benefit the course as a whole?
  - Precept attendance and participation counts

Programming Assignments

Programming assignments
0. Introductory survey
1. "De-comment" program
2. String module
3. Symbol table module
4. Assembly language programs
5. Buffer overrun attack (partner from your precept)
6. Heap manager module (partner from your precept)
7. Unix shell

Assignments 0 and 1 are available now
Start early!!!

University rules:

Sources of help, citing your sources

2.4.5 Tutoring
An undergraduate is subject to disciplinary action if that student makes use of any tutoring service or facility other than that regularly authorized by the Office of the Dean of the College.

2.4.6 General Requirements for the Acknowledgment of Sources in Academic Work
... An important general rule is this: If you are unsure whether or not to acknowledge a source, always err on the side of caution and completeness by citing rather than not citing. ...
...
In those cases where individual reports are submitted based on work involving collaboration, proper acknowledgment of the extent of the collaboration must appear in the report. ... each student's signature is taken to mean that the student has contributed fairly to the work involved ...
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 professor for clarifications
- Only Prof. Gupta can waive any policies (and only in writing)

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</td>
<td>Linux/GNU</td>
</tr>
<tr>
<td></td>
<td>C (conceptual)</td>
<td>C (pragmatic)</td>
</tr>
<tr>
<td>3-6</td>
<td>“Pgmming in the Large”</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>“Under the Hood” (conceptual)</td>
<td>“Under the Hood” (pgmming asgts)</td>
</tr>
<tr>
<td></td>
<td>Reading Period</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Final Exam</td>
<td></td>
</tr>
</tbody>
</table>

The C Programming Language

Who? Dennis Ritchie
When? ~1972
Where? Bell Labs
Why? Compose the Unix OS
Java vs. C: History

Our compiler supports these only partially

We will use

<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>LISP</td>
</tr>
<tr>
<td>1972</td>
<td>B</td>
</tr>
<tr>
<td>1978</td>
<td>C</td>
</tr>
<tr>
<td>1989</td>
<td>K&amp;R C</td>
</tr>
<tr>
<td>1999</td>
<td>ISO C99</td>
</tr>
<tr>
<td>2011</td>
<td>ISO C11</td>
</tr>
</tbody>
</table>

BCPL -> B -> C -> K&R C -> ISO C99 -> ISO C11

Java vs. C: Design Goals

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Language of the Internet</td>
<td>Compose Unix OS</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>Safe: can't step &quot;outside the sandbox&quot;</td>
<td>Unsafe: don't get in the programmer's way</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

$ javac MyPgm.java

Java compiler (machine lang code)

Running Java Programs

$ java MyPgm

Java interpreter (Java virtual machine) (machine lang code)

Building C Programs

$ gcc217 mypgm.c –o mypgm

C "compiler driver" (machine lang code)
Running C Programs

$ ./mypgm

mypgm (machine lang code)

OS (Linux)

HW (CourseLab)

data
data

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 has automatic array-bounds checking, nullpointer checking, automatic memory management (garbage collection), other safety features

C has manual bounds checking, null checking, memory management

Result: C programs are (often) faster

Result 2: C programs are buggy, exploitable

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>

Java vs. C: Characteristics

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></td>
</tr>
<tr>
<td>char</td>
<td>char /* 8 bits */</td>
</tr>
<tr>
<td>byte</td>
<td>(unsigned) char</td>
</tr>
<tr>
<td>short</td>
<td>(unsigned) short</td>
</tr>
<tr>
<td>int</td>
<td>(unsigned) int</td>
</tr>
<tr>
<td>long</td>
<td>(unsigned) long</td>
</tr>
<tr>
<td>Floating point types</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>float double</td>
</tr>
<tr>
<td>double</td>
<td>long double</td>
</tr>
<tr>
<td>Logical type</td>
<td>/* no equivalent */</td>
</tr>
<tr>
<td>boolean</td>
<td>/* use integral type */</td>
</tr>
<tr>
<td>Generic pointer type</td>
<td>Object</td>
</tr>
<tr>
<td>Constants</td>
<td>final int 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];</td>
</tr>
<tr>
<td>Array bound checking</td>
<td>/* no run-time check */</td>
</tr>
<tr>
<td>Pointer type</td>
<td>int *p;</td>
</tr>
<tr>
<td>Record type</td>
<td>struct Mine</td>
</tr>
</tbody>
</table>

Java vs. C: Details

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

<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></td>
</tr>
<tr>
<td><strong>String concatenation</strong></td>
<td></td>
</tr>
<tr>
<td><code>s1 + s2</code></td>
<td><code>strcpy(s1, s2);</code></td>
</tr>
<tr>
<td><code>s1 += s2</code></td>
<td></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>, <code>unary -</code></td>
<td><code>+</code>, <code>-</code>, <code>*</code>, <code>/</code>, <code>%</code>, <code>unary -</code></td>
</tr>
<tr>
<td><strong>Bitwise ops</strong></td>
<td></td>
</tr>
<tr>
<td><code>&gt;&gt;</code>, <code>&lt;&lt;</code>, <code>&gt;&gt;&gt;</code>, <code>&amp;</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></td>
<td><code>=</code>, <code>*=</code>, <code>/=</code>, <code>+=</code>, <code>-=</code></td>
</tr>
<tr>
<td><code>&amp;=</code>, `</td>
<td>=<code>, </code>^=<code>, </code>%=`</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>do</code></td>
</tr>
<tr>
<td><code>statement; while (i &lt; 0);</code></td>
<td><code>statement; while (i &lt; 0);</code></td>
</tr>
<tr>
<td><strong>continue stmt</strong></td>
<td></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>continue someLabel;</code></td>
<td><code>/* no equivalent */</code></td>
</tr>
<tr>
<td><strong>break stmt</strong></td>
<td></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>break someLabel;</code></td>
<td><code>/* no equivalent */</code></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>if stmt</strong></td>
<td></td>
</tr>
<tr>
<td><code>if (i &lt; 0)</code></td>
<td><code>switch (i)</code></td>
</tr>
<tr>
<td><code>statement1;</code></td>
<td><code>{ case 1:</code></td>
</tr>
<tr>
<td><code>else</code></td>
<td><code>... break;</code></td>
</tr>
<tr>
<td><code>statement2;</code></td>
<td><code>case 2:</code></td>
</tr>
<tr>
<td><strong>switch stmt</strong></td>
<td></td>
</tr>
<tr>
<td><code>switch (i)</code></td>
<td><code>default:</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>... break;</code></td>
</tr>
<tr>
<td><code>case 1:</code></td>
<td><code>... break;</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>default:</code></td>
</tr>
<tr>
<td><code>case 2:</code></td>
<td><code>... break;</code></td>
</tr>
<tr>
<td><code>... break;</code></td>
<td><code>...</code></td>
</tr>
<tr>
<td><code>}</code></td>
<td><code>}</code></td>
</tr>
<tr>
<td><code>goto stmt</code></td>
<td><code>/* no equivalent */</code></td>
</tr>
<tr>
<td><code>// no equivalent</code></td>
<td><code>goto someLabel;</code></td>
</tr>
</tbody>
</table>

* 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" and learn low-level programming
- **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