COS 217: Introduction to Programming Systems
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

Professor
  • Aarti Gupta  aartig@cs.princeton.edu

Lead Preceptors
  • Robert Dondero  rdondero@cs.princeton.edu
  • Ananda Gunawardena (“Guna”)  guna@cs.princeton.edu

Preceptors
  • Huilian (Sophie) Qiu  hqiu@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, debugging, performance improvement, tool support
Goal 2: “Under the Hood”

Learn what happens “under the hood” of computer systems

Learn “how to be a client of an operating system”

Downward tours

- **C Language**
  - Assembly Language
  - Machine Language
  - *language levels tour*

- **Application Program**
  - Operating System
  - Hardware
  - *service levels tour*
Modular systems
Goals: Summary

Help you to become a...

Power Programmer!!!
**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
The Pen Is Mightier Than the Keyboard
Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller\textsuperscript{1}
Daniel M. Oppenheimer\textsuperscript{2}

\textsuperscript{1}Princeton University
\textsuperscript{2}University of California, Los Angeles

Pam A. Mueller, Princeton University, Psychology Department,
Princeton, NJ 08544 E-mail: pamuelle@princeton.edu

Abstract

taking notes on laptops rather than in longhand is increasingly common. Many researchers have suggested that laptop note taking is less effective than longhand note taking for learning. Prior studies have primarily focused on students’ capacity for multitasking and distraction when using laptops. The present research suggests that even when laptops are used solely to take notes, they may still be impairing learning because their use results in shallower processing. In three studies, we found that students who took notes on laptops performed worse on conceptual questions than students who took notes longhand. We show that whereas taking more notes can be
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-McGilley (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

- Princeton CS → Courses → Course Schedule → COS 217
- Home page, schedule page, assignment page, policies page
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 (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

- coursela01
- coursela02

**Client**

Your 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

* 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

<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>

These percentages are approximate
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!!!
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)
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 . . .
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 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>
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)
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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BCPL</td>
<td>B</td>
<td>C</td>
<td>K&amp;R C</td>
<td>ANSI C89</td>
<td>ISO C90</td>
<td>ISO C11</td>
</tr>
<tr>
<td></td>
<td>LISP</td>
<td>Smalltalk</td>
<td>C++</td>
<td>Java</td>
<td>ISO C99</td>
<td>ANSI C99</td>
<td></td>
</tr>
</tbody>
</table>

31
### 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><strong>Safe</strong>: can’t step “outside the sandbox”</td>
<td><strong>Unsafe</strong>: 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
```

```
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 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
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

Slides covered now, as time allows…
## Java vs. C: Details

<table>
<thead>
<tr>
<th>Overall Program Structure</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
</table>
| Hello.java: | public class Hello  
{  public static void main (String[] args)  
{    System.out.println(     "hello, world");  
}  
} | hello.c:  
#include <stdio.h>  
int main(void)  
{    printf("hello, world\n");    return 0;  
} |
| 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><strong>Integral types</strong></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><strong>Floating point types</strong></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></td>
<td>long double</td>
</tr>
<tr>
<td><strong>Logical type</strong></td>
<td>boolean</td>
<td>/* no equivalent */</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/* use integral type */</td>
</tr>
<tr>
<td><strong>Generic pointer type</strong></td>
<td>Object</td>
<td>void*</td>
</tr>
<tr>
<td><strong>Constants</strong></td>
<td>final int MAX = 1000;</td>
<td>#define MAX 1000</td>
</tr>
<tr>
<td></td>
<td></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>Arrays</td>
<td>int [] a = new int [10]; float [][] b =</td>
<td>int a[10]; float b[5][20];</td>
</tr>
<tr>
<td></td>
<td>new float [5][20];</td>
<td></td>
</tr>
<tr>
<td>Array bound checking</td>
<td>// run-time check</td>
<td>/* no run-time check */</td>
</tr>
<tr>
<td>Pointer type</td>
<td>// Object reference is an implicit pointer</td>
<td>int *p;</td>
</tr>
<tr>
<td>Record type</td>
<td>class Mine { int x; float y; }</td>
<td>struct Mine { int x; float y; }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></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>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 String(&quot;hello&quot;);</td>
<td>char s2[6];</td>
</tr>
<tr>
<td></td>
<td></td>
<td>strcpy(s2, &quot;hello&quot;);</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td>s1 + s2</td>
<td>#include &lt;string.h&gt;</td>
</tr>
<tr>
<td><strong>concatenation</strong></td>
<td>s1 += s2</td>
<td>strcat(s1, s2);</td>
</tr>
<tr>
<td><strong>Logical</strong></td>
<td>&amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td><strong>ops</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Relational</strong></td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td><strong>ops</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Arithmetic</strong></td>
<td>+, -, *, /, %, unary -</td>
<td>+, -, *, /, %, unary -</td>
</tr>
<tr>
<td><strong>ops</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bitwise</strong></td>
<td>&gt;&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;&gt;, &amp;,,</td>
<td>, ^</td>
</tr>
<tr>
<td><strong>ops</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Assignment</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>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>if stmt *</td>
<td><code>if (i &lt; 0) statement1;</code></td>
<td><code>if (i &lt; 0) statement1;</code></td>
</tr>
<tr>
<td></td>
<td><code>else statement2;</code></td>
<td><code>else statement2;</code></td>
</tr>
<tr>
<td>switch stmt *</td>
<td><code>switch (i)</code></td>
<td><code>switch (i)</code></td>
</tr>
<tr>
<td></td>
<td><code>{ case 1: ... break;</code></td>
<td><code>{ case 1: ... break;</code></td>
</tr>
<tr>
<td></td>
<td><code>case 2: ... break;</code></td>
<td><code>case 2: ... break;</code></td>
</tr>
<tr>
<td></td>
<td><code>default: ...</code></td>
<td><code>default: ...</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code></td>
<td><code>}</code></td>
</tr>
<tr>
<td>goto stmt</td>
<td><code>// no equivalent</code></td>
<td><code>goto someLabel;</code></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
</table>
| for stmt       | `for (int i=0; i<10; i++)`\n`statement;` | `int i;`
`for (i=0; i<10; i++)`\n`statement;` |
| while stmt     | `while (i < 0)`\n`statement;`             | `while (i < 0)`\n`statement;`         |
| do-while stmt  | `do`\n`statement;`
`while (i < 0)` | `do`\n`statement;`
`while (i < 0);` |
| continue stmt  | `continue;`                             | `continue;`                           |
| labeled continue stmt | `continue someLabel;`            | /* no equivalent */                   |
| break stmt     | `break;`                                 | `break;`                               |
| labeled break stmt | `break someLabel;`                  | /* no equivalent */                   |

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