Princeton University
Computer Science 217: Introduction to Programming Systems

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

Professor
- Andrew W. Appel appel@cs.princeton.edu

Lead Preceptors
- Iasonas Petras ipetras@cs.princeton.edu
- Xiaoyan Li xiaoyan@cs.princeton.edu

Faculty Preceptors
- Donna Gabai dgabai@princeton.edu

Preceptors
- Oluwatosin Adewale oadewale@princeton.edu
- Gregory W. Gundersen ggundersen@princeton.edu
- Seo Young Kyung skyung@princeton.edu
- Austin Le austinle@princeton.edu

Goal 1: Programming in the Large

Goal 1: “Programming in the large”
- Help you learn how to write 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
Application Program

Assembly Language
Operating System

Machine Language
Hardware
Modular systems

Goals: Summary

Help you to become a...

Power Programmer!!!

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

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

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 \Rightarrow See Colleen Kenny-McGinley (CS Bldg 210)
  \- But Colleen can’t move you into a full precept
- Must miss your precept? \Rightarrow inform preceptors & attend another

Precepts begin Monday

Website

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

Piazza

Piazza
- http://piazza.com/class#fall2017/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

<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?
  - 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!!!

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)

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. Appel 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>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”</td>
<td>“Under the Hood”</td>
</tr>
<tr>
<td></td>
<td>(conceptual)</td>
<td>(pgmming asgts)</td>
</tr>
<tr>
<td></td>
<td>Reading Period</td>
<td>Final Exam</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

<table>
<thead>
<tr>
<th>Year</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>Algol</td>
</tr>
<tr>
<td>1970</td>
<td>BCPL</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 C99</td>
</tr>
<tr>
<td>1999</td>
<td>ISO C99</td>
</tr>
<tr>
<td>2011</td>
<td>ISO C11</td>
</tr>
</tbody>
</table>

### C vs. Java: Design Goals

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the Unix OS</td>
<td>Language of the Internet</td>
</tr>
<tr>
<td>Low-level; close to HW and OS</td>
<td>High-level; insulated from hardware and OS</td>
</tr>
<tr>
<td>Good for system-level programming</td>
<td>Good for application-level programming</td>
</tr>
<tr>
<td>Support structured programming</td>
<td>Support object-oriented programming</td>
</tr>
<tr>
<td>Unsafe: don’t get in the programmer’s way</td>
<td>Safe: can’t step “outside the sandbox”</td>
</tr>
<tr>
<td></td>
<td>Look like C</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
```

### Running Java Programs

```bash
java MyPgm
```

### Building C Programs

```bash
gcc217 mypgm.c –o mypgm
```
Running C Programs

$ ./mypgm

OS (Linux)

HW (CourseLab)

data

mypgm (machine lang code)

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>
</tbody>
</table>

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>Feature</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>char // 16-bit Unicode char /* 8 bits */</td>
</tr>
<tr>
<td>Integral types</td>
<td>byte // 8 bits (unsigned) char short // 16 bits (unsigned) short int // 32 bits (unsigned) int long // 64 bits (unsigned) long</td>
</tr>
<tr>
<td>Floating point types</td>
<td>float // 32 bits float double // 64 bits double long double</td>
</tr>
</tbody>
</table>
| Logical type | boolean /* no equivalent */
| Generic pointer type | Object void* |
| Constants | final int MAX = 1000; |

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>float [][] b = new float [5][20];</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>Strings</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>String s1 = &quot;Hello&quot;;</td>
<td>char *s1 = &quot;Hello&quot;;</td>
<td></td>
</tr>
<tr>
<td>String s2 = new String(&quot;hello&quot;);</td>
<td>char s2[6];</td>
<td></td>
</tr>
<tr>
<td>strcpy(s2, &quot;hello&quot;);</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String concatenation</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1 + s2</td>
<td>$\text{include &lt;string.h&gt;}$</td>
<td>strcat(s1, s2);</td>
</tr>
<tr>
<td>s1 += s2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logical ops *</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;,</td>
<td></td>
<td>, !</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relational ops *</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>=, !=, &gt;, &lt;=, &gt;=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arithmetic ops</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, *, /, %, unary -</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitwise ops</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;, &amp;,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assignment ops</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>=, *=, /=, +=, -=, &lt;&lt;=, &gt;&gt;=, &gt;&gt;&gt;=, =, &amp;=, ^=,</td>
<td>=, %=</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>if stmt *</td>
<td>if (i &lt; 0)</td>
</tr>
<tr>
<td></td>
<td>statement1;</td>
</tr>
<tr>
<td></td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>statement2;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>switch stmt *</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>switch (i)</td>
<td>{ case 1:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>case 2:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>default:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>goto stmt</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto someLabel;</td>
<td>// no equivalent</td>
<td></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