$ cat welcome.c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("COS 217\n");
    printf("Introduction to Programming Systems\n\n");
    printf("Spring, 2018\n");
    return 0;
}

$ gcc217 welcome.c -o welcome
$ ./welcome
COS 217
Introduction to Programming Systems
Spring, 2018
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

Lead Instructor
• Prof. Szymon Rusinkiewicz  smr@princeton.edu

Lead Preceptor
• Robert Dondero, Ph.D.  rdondero@cs.princeton.edu

Faculty Preceptor
• Donna Gabai  dgabai@cs.princeton.edu

Preceptors
• Seo Young Kyung  skyung@princeton.edu
• Austin Le  austinle@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

- Application Program
- Operating System
- Hardware

language levels tour

service levels tour
Modularity!
Goals: Summary

Help you to become a...

Power Programmer!!!
Goals: Why C?

Question: Why C instead of Java?

Answer 1: Primary language for “under the hood” programming

Answer 2: Knowing a variety of approaches helps you “program in the large”
Goals: Why Linux?

**Question:** Why use the Linux operating system?

**Answer 1:** Linux is good for education and research

**Answer 2:** Linux (with GNU tools) 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

• Describe material at conceptual (high) level
• Slides available via course website

Lecture etiquette

• Use electronic devices *only* for taking notes or annotating slides
• No FaceNewsChatBookSnapMail, please

iClicker

• Please obtain one and register in Blackboard (not with iClicker – they’ll charge you)
• Occasional questions in class, graded on participation (with a generous allowance for not being able to attend)
iClicker Question

Q: Do you have an iClicker with you today?

• A. Yes

• B. No, but I’ve been practicing my mental electrotelekinesis and the response is being registered anyway

• C. I’m not here, but someone is iClicking for me (don’t do this!)
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 – attendance will be taken
• 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 and tomorrow!
Website

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

- http://piazza.com/class#spr2018/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

- courselab01
- courselab02

Client

Your Computer

On-campus or off-campus

SSH
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>Participation ***</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/precept attendance and participation counts

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!!!
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)
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 for clarifications

• Permission to deviate from policies must be obtained 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>Spring break!</td>
</tr>
<tr>
<td>8-13</td>
<td>“Under the Hood” (conceptual)</td>
<td>“Under the Hood” (assignment how-to)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reading Period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Exam</td>
</tr>
</tbody>
</table>
Questions?
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? Build the Unix OS
Java vs. C: History

- 1960: Algol
- 1970: BCPL
- 1972: B
- 1978: C
- 1978: K&R C
- 1989: ANSI C89
- 1989: ISO C90
- 1999: ISO C99
- 1999: ANSI C99
- 2011: ISO C11

- 1960: LISP
- 1970: Smalltalk
- 1978: C++
- 1978: Java
## 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><strong>Unsafe: don’t get in the programmer’s way</strong></td>
<td><strong>Safe: can’t step “outside the sandbox”</strong></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

$ javac MyPgm.java

Java compiler (machine lang code)

HW (CourseLab)

OS (Linux)

MyPgm.java (Java code)

javac

MyPgm.class (bytecode)
$ 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

```bash
$ ./mypgm
```

Diagram:
- `data` connected to `mypgm` (yellow)
- `mypgm` connected to `data`
- `mypgm` is inside `OS (Linux)`
- `OS (Linux)` inside `HW (CourseLab)`

Notes:
- `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>
</tbody>
</table>

**Conclusion:** Java programs are more portable
Java vs. C: Safety & Efficiency

Java

- Automatic array-bounds checking,
- NULL pointer checking,
- Automatic memory management (garbage collection)
- Other safety features

C

- Manual bounds checking
- NULL pointer checking,
- Manual memory management

Conclusion 1: Java is often safer than C
Conclusion 2: Java is often slower than C
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>
Q: Which corresponds to the C programming language?

• A.

• B.

• 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>
<tbody>
<tr>
<td></td>
<td>Hello.java:</td>
<td>hello.c:</td>
</tr>
<tr>
<td></td>
<td>public class Hello</td>
<td>#include &lt;stdio.h&gt;</td>
</tr>
<tr>
<td></td>
<td>{ public static void main (String[] args)</td>
<td>int main(void)</td>
</tr>
<tr>
<td></td>
<td>{ System.out.println(</td>
<td>{ printf(&quot;hello, world\n&quot;);</td>
</tr>
<tr>
<td></td>
<td>&quot;hello, world&quot;);</td>
<td>return 0;</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>Building</td>
<td>$ javac Hello.java</td>
<td>$ gcc217 hello.c -o hello</td>
</tr>
<tr>
<td>Running</td>
<td>$ java Hello</td>
<td>$ ./hello</td>
</tr>
<tr>
<td></td>
<td>hello, world</td>
<td>hello, world</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>$</td>
</tr>
</tbody>
</table>
# Java vs. C: Details

<table>
<thead>
<tr>
<th>Character Type</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>char \ 16-bit Unicode</td>
<td>char /* 8 bits */</td>
</tr>
</tbody>
</table>

## Integral Types

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

## Floating Point Types

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>float double</td>
</tr>
<tr>
<td>double</td>
<td>long double</td>
</tr>
</tbody>
</table>

## Logical Type

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>/* no equivalent */</td>
</tr>
</tbody>
</table>

## Generic Pointer Type

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>void*</td>
</tr>
</tbody>
</table>

## Constants

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>final int MAX = 1000;</td>
<td>#define MAX 1000</td>
</tr>
<tr>
<td></td>
<td>const int MAX = 1000;</td>
</tr>
<tr>
<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>int [] a = new int [10];</td>
<td>int a[10];</td>
</tr>
<tr>
<td></td>
<td>float [][] b = new float [5][20];</td>
<td>float b[5][20];</td>
</tr>
<tr>
<td><strong>Array bound</strong></td>
<td>// run-time check</td>
<td>/* no run-time check */</td>
</tr>
<tr>
<td><strong>checking</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pointer type</strong></td>
<td>// Object reference is an implicit pointer</td>
<td>int *p;</td>
</tr>
<tr>
<td><strong>Record type</strong></td>
<td>class Mine {</td>
<td>struct Mine {</td>
</tr>
<tr>
<td></td>
<td>int x;</td>
<td>int x;</td>
</tr>
<tr>
<td></td>
<td>float y;</td>
<td>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>Strings</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>String concatenation</td>
<td>s1 + s2</td>
<td>#include &lt;string.h&gt;</td>
</tr>
<tr>
<td></td>
<td>s1 += s2</td>
<td>strcat(s1, s2);</td>
</tr>
<tr>
<td>Logical ops *</td>
<td>&amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td>Relational ops *</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td>Arithmetic ops *</td>
<td>+, -, *, /, %, unary -</td>
<td>+, -, *, /, %, unary -</td>
</tr>
<tr>
<td>Bitwise ops</td>
<td>&gt;&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;&gt;, &amp;,</td>
<td>, ^</td>
</tr>
<tr>
<td>Assignment ops</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>if (i &lt; 0)</td>
<td>if (i &lt; 0)</td>
</tr>
<tr>
<td></td>
<td>statement1;</td>
<td>statement1;</td>
</tr>
<tr>
<td></td>
<td>else</td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>statement2;</td>
<td>statement2;</td>
</tr>
<tr>
<td>switch stmt *</td>
<td>switch (i)</td>
<td>switch (i)</td>
</tr>
<tr>
<td></td>
<td>{ case 1:</td>
<td>{ case 1:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td></td>
<td>case 2:</td>
<td>case 2:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td></td>
<td>default:</td>
<td>default:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>goto stmt</td>
<td>// no equivalent</td>
<td>goto someLabel;</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>
<tbody>
<tr>
<td><strong>for stmt</strong></td>
<td>for (int i = 0; i &lt; 10; i++) &lt;br&gt;<code>statement;</code></td>
<td>int i;&lt;br&gt;for (i = 0; i &lt; 10; i++)&lt;br&gt;<code>statement;</code></td>
</tr>
<tr>
<td><strong>while stmt</strong></td>
<td>while (i &lt; 0) &lt;br&gt;<code>statement;</code></td>
<td>while (i &lt; 0) &lt;br&gt;<code>statement;</code></td>
</tr>
<tr>
<td><strong>do-while stmt</strong></td>
<td>do&lt;br&gt;<code>statement;</code>&lt;br&gt;while (i &lt; 0)</td>
<td>do&lt;br&gt;<code>statement;</code>&lt;br&gt;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 <code>someLabel;</code></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 <code>someLabel;</code></td>
<td>/* no equivalent */</td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages*
<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>return stmt</strong></td>
<td><strong>return 5;</strong>&lt;br&gt;<strong>return;</strong></td>
</tr>
<tr>
<td><strong>Compound stmt (alias block)</strong></td>
<td><code>{&lt;br&gt;**statement1;**&lt;br&gt;**statement2;**}&lt;br&gt;}</code></td>
</tr>
<tr>
<td><strong>Exceptions</strong></td>
<td><strong>throw, try-catch-finally</strong></td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>/* comment */&lt;br}// another kind</td>
</tr>
<tr>
<td><strong>Method / function call</strong></td>
<td><code>f(x, y, z);</code>&lt;br&gt;<code>someObject.f(x, y, z);</code>&lt;br&gt;<code>SomeClass.f(x, y, z);</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
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