



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

Instructor-of-Record

- Robert Dondero, Ph.D.
 - rdondero@cs.princeton.edu



Lead Preceptor

- Iasonas Petras, Ph.D.
 - ipetras@cs.princeton.edu



Introductions

Preceptors and Graders (in alphabetical order)

- Po-Hsuan (Cameron) Chen
 - pohsuan@princeton.edu
- Alexander Iriza
 - airiza@princeton.edu
- Alan Kaplan, Ph.D.
 - ak18@cs.princeton.edu
- Raghav Sethi
 - raghavs@princeton.edu
- Yannan (Terry) Wang
 - yannanw@princeton.edu
- Yuxuan Wang
 - ywthree@princeton.edu
- Tongbin (Tom) Wu
 - tongbinw@princeton.edu
- Haoyu (Harris) Zhang
 - haoyuz@princeton.edu



Agenda

Course overview

- Introductions
- **Course goals**
- Resources
- Grading
- Policies
- Schedule

Getting started with C

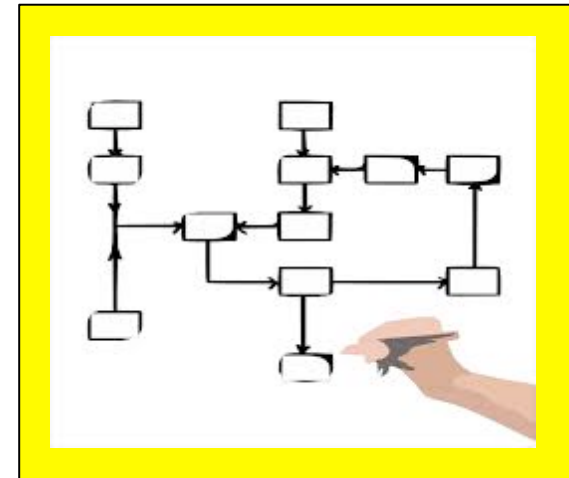
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)

Goal 1: “Pgmming 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

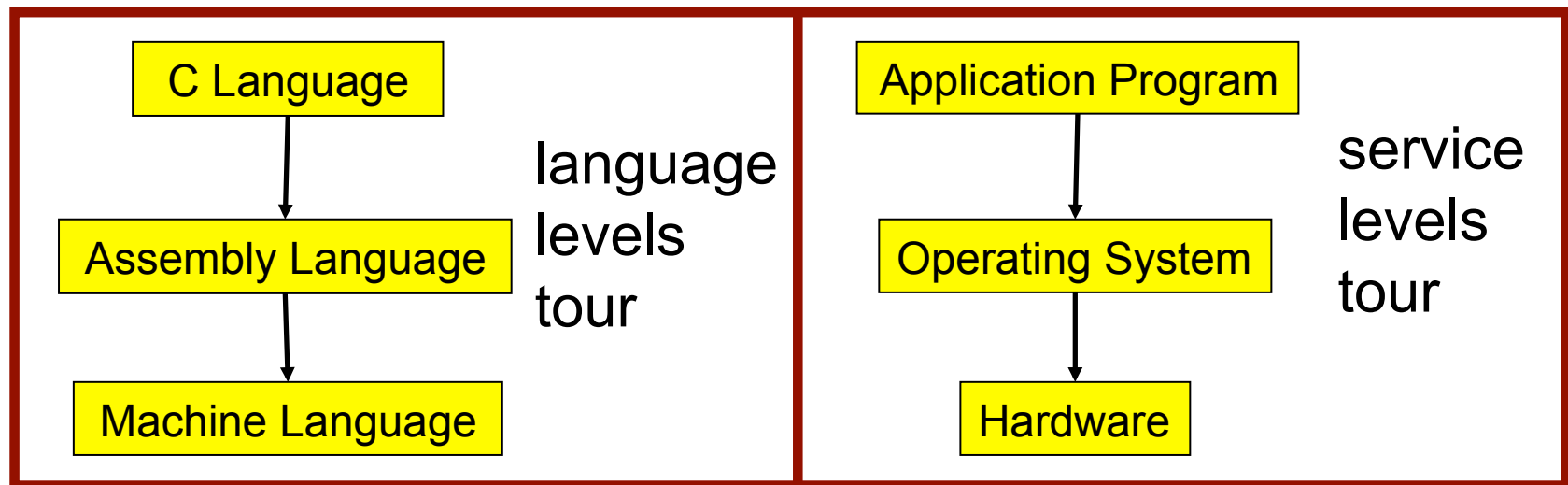


Goal 2: “Under the Hood”

Goal 2: “Look under the hood”

- Help you learn what happens “under the hood” of computer systems

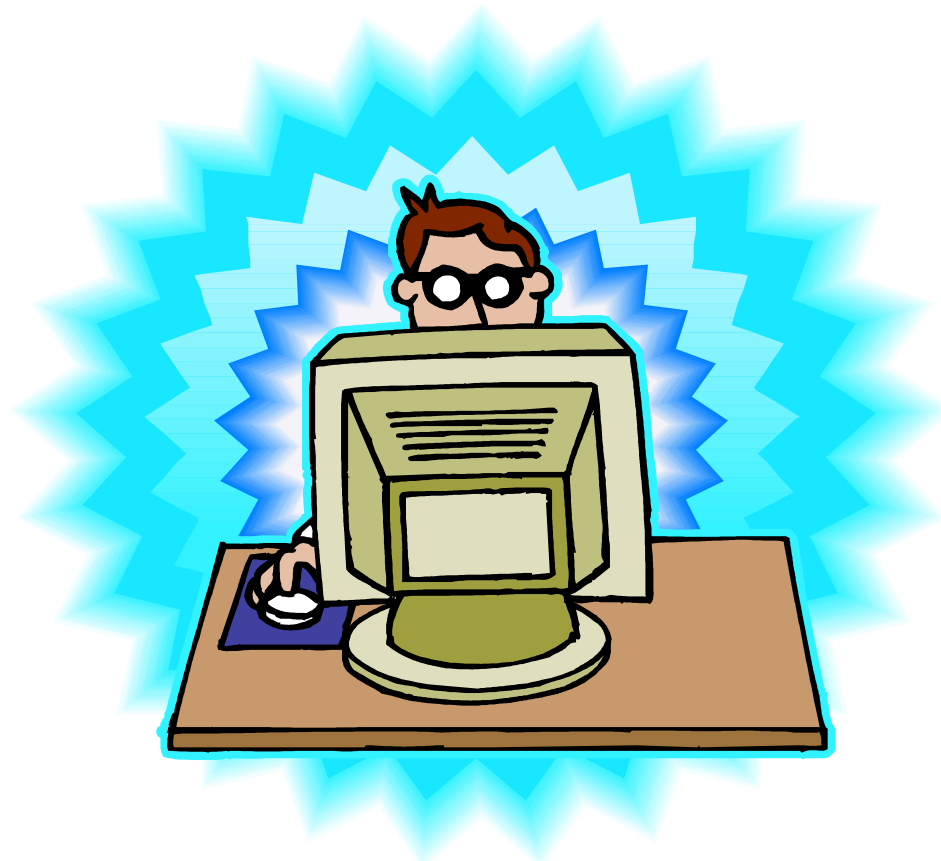
Downward tours





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

THE
C
PROGRAMMING
LANGUAGE



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 (high) 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
- 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 Monday September 15



Website

Website

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





Piazza

Piazza

- <http://piazza.com/class#fall2014/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

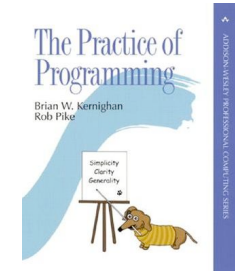


Books



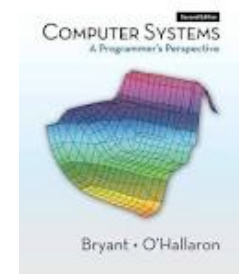
The Practice of Programming (recommended)

- Kernighan & Pike
- “Programming in the large”



Computer Systems: A Programmer's Perspective (Second Edition) (recommended)

- Bryant & O'Hallaron
- “Under the hood”



C Programming: A Modern Approach (Second Edition) (required)

- King
- C programming language and standard libraries





Manuals

Manuals (for reference only, available online)

- *IA32 Intel Architecture Software Developer's Manual, Volumes 1-3*
- *Tool Interface Standard & Executable and Linking Format*
- *Intel 64 and IA-32 Architectures Optimization Reference Manual*
- *Using as, the GNU Assembler*



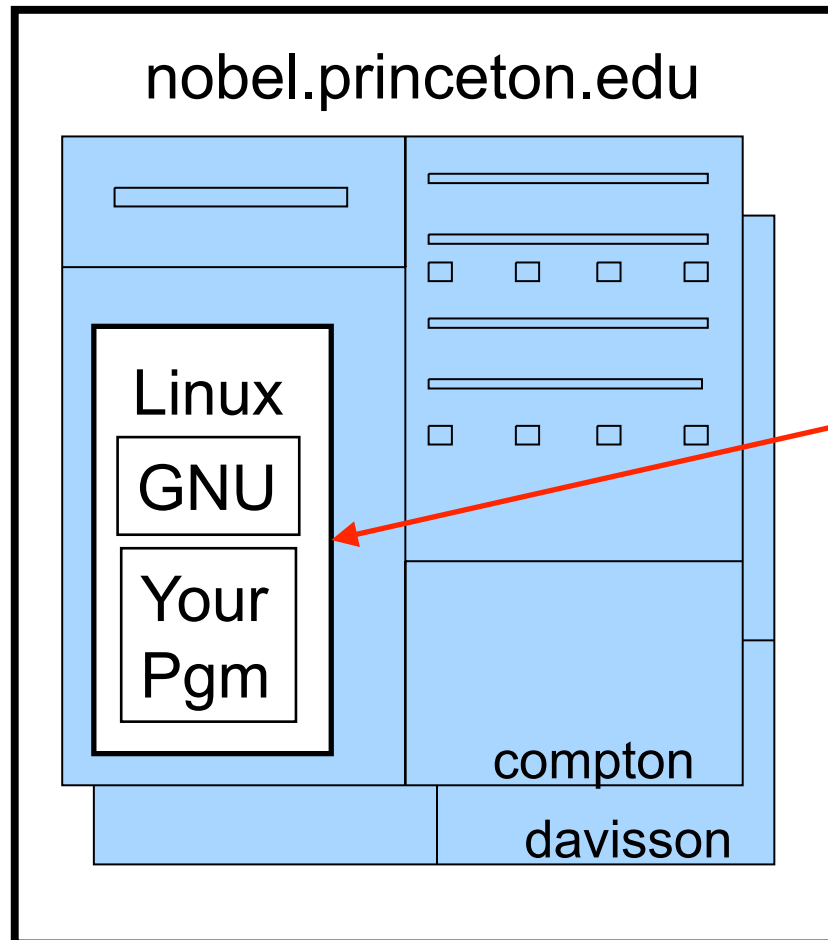
See also

- Linux **man** command

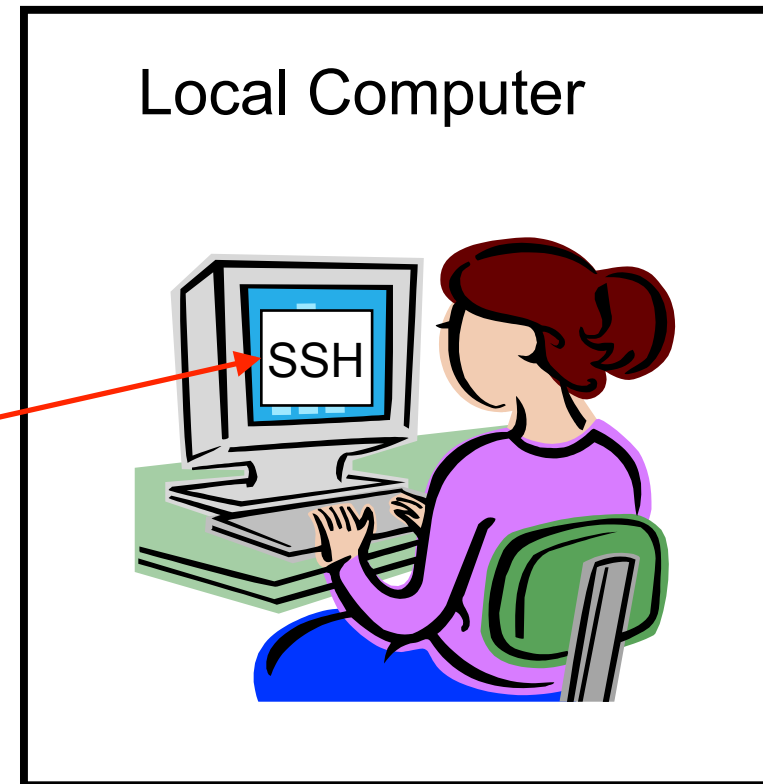


Programming Environment

Server



Client



Your computer or cluster 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

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?
 - Lecture and precept attendance and participation counts



Programming Assignments

Programming assignments

- A “de-comment” program
- A string module
- A symbol table module
- IA-32 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!!!



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-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



Weeks	Lectures	Precepts
1-2	Number Systems C (conceptual)	Linux/GNU C (pragmatic)
3-6	“Pgmning in the Large”	Advanced C
6	Midterm Exam	
7	Recess	
8-13	“Under the Hood” (conceptual)	“Under the Hood” (pgmning asgts)
	Reading Period	
	Final Exam	



Any 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? Compose the Unix OS

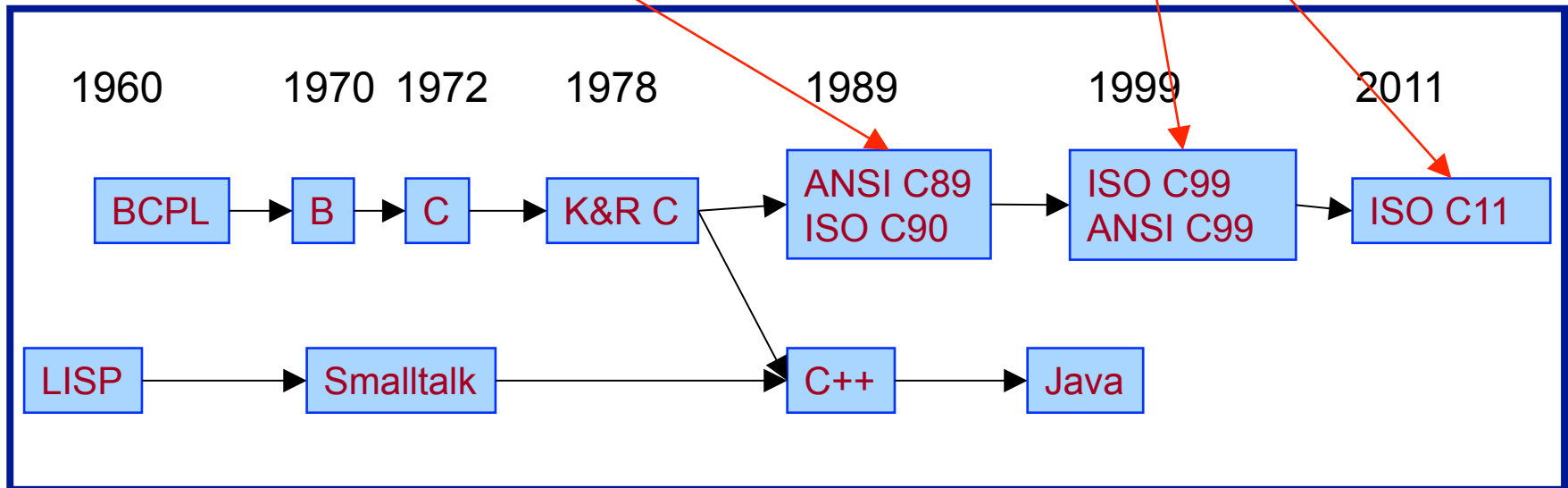


Java vs. C: History



We will use

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





Java vs. C: Design Goals

Java Design Goals	C Design Goals
Language of the Internet	Compose Unix
High-level; insulated from hardware and OS	Low-level; close to HW and OS
Good for application-level programming	Good for system-level programming
Support object-oriented programming	Support structured programming
Look like C!	



Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C

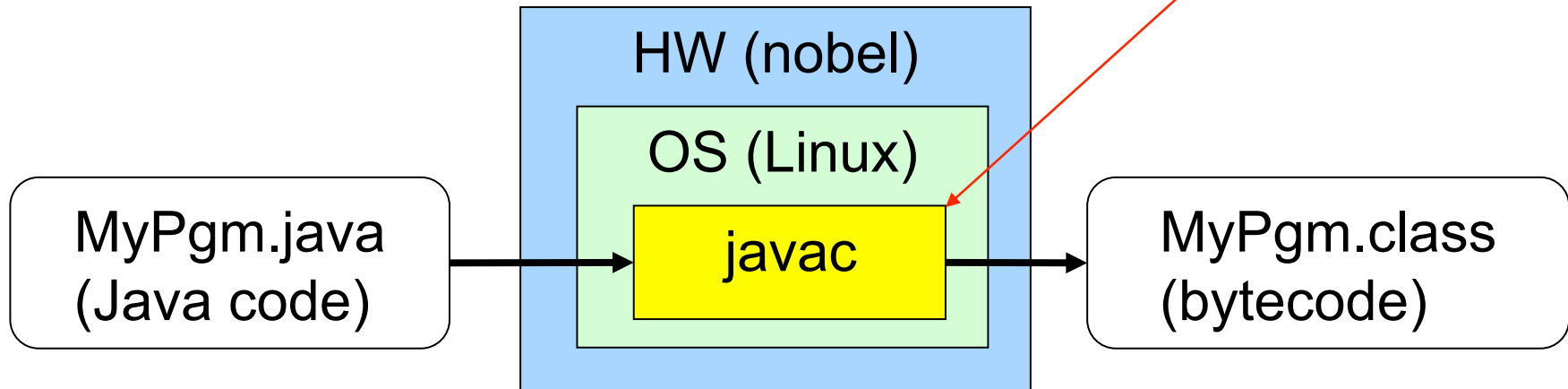
- History of C
- **Building and running C programs**
- Characteristics of C
- C details (if time)



Building Java Programs

\$ javac MyPgm.java

Java compiler
(machine lang code)

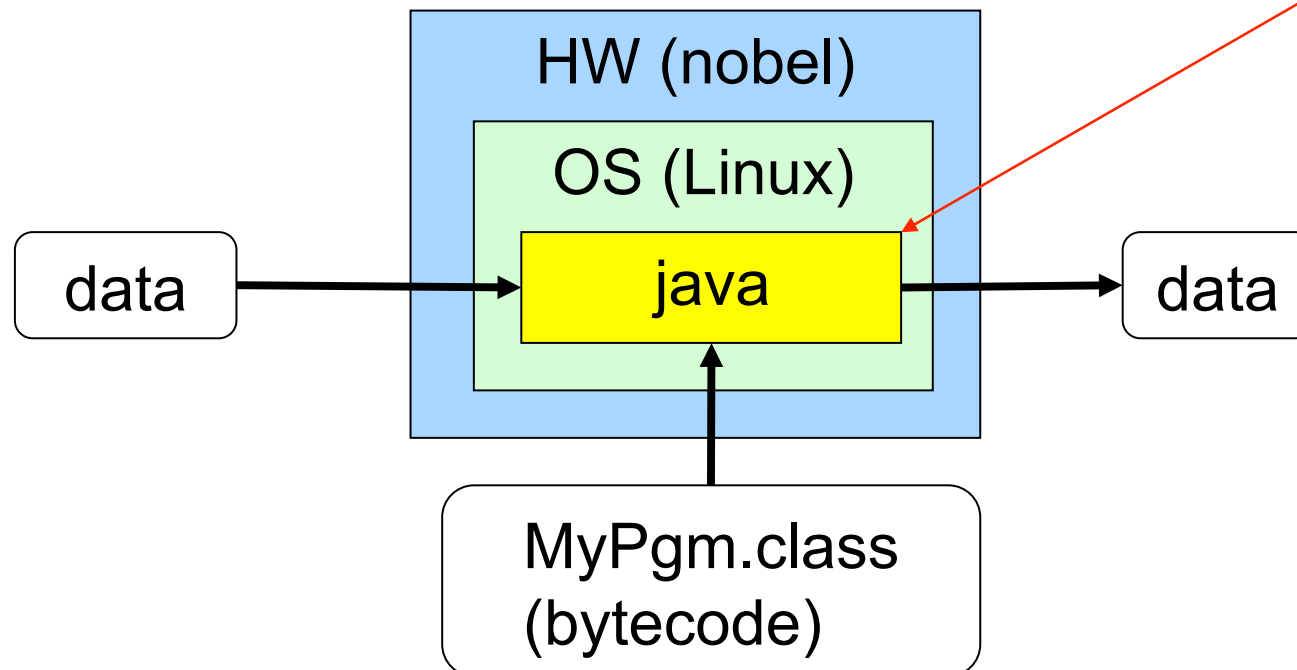




Running Java Programs

`$ java MyPgm`

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

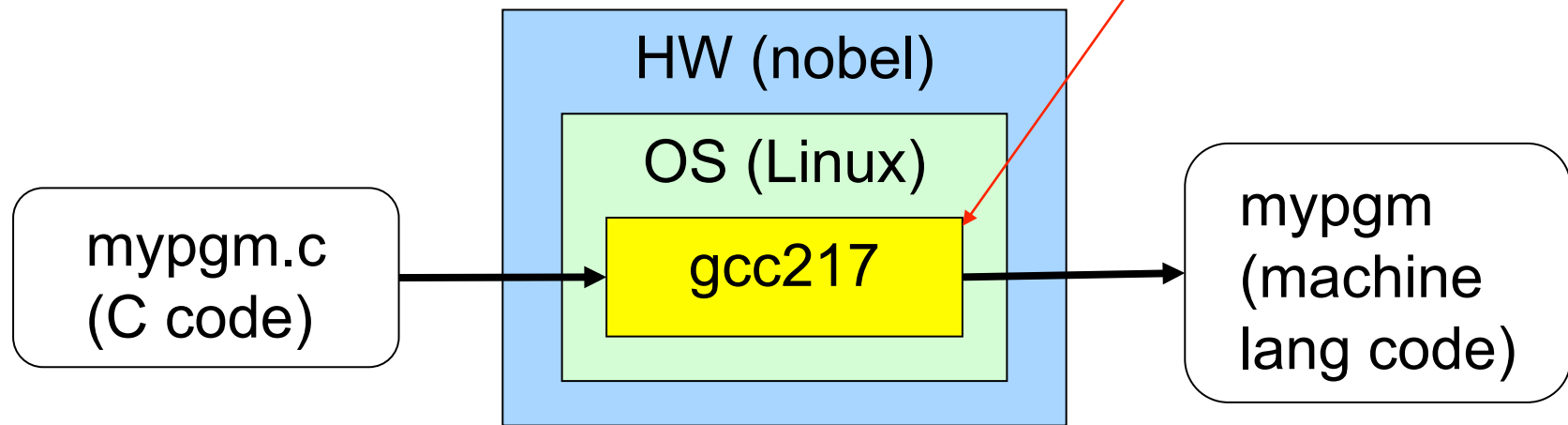




Building C Programs

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

C “compiler driver”
(machine lang code)

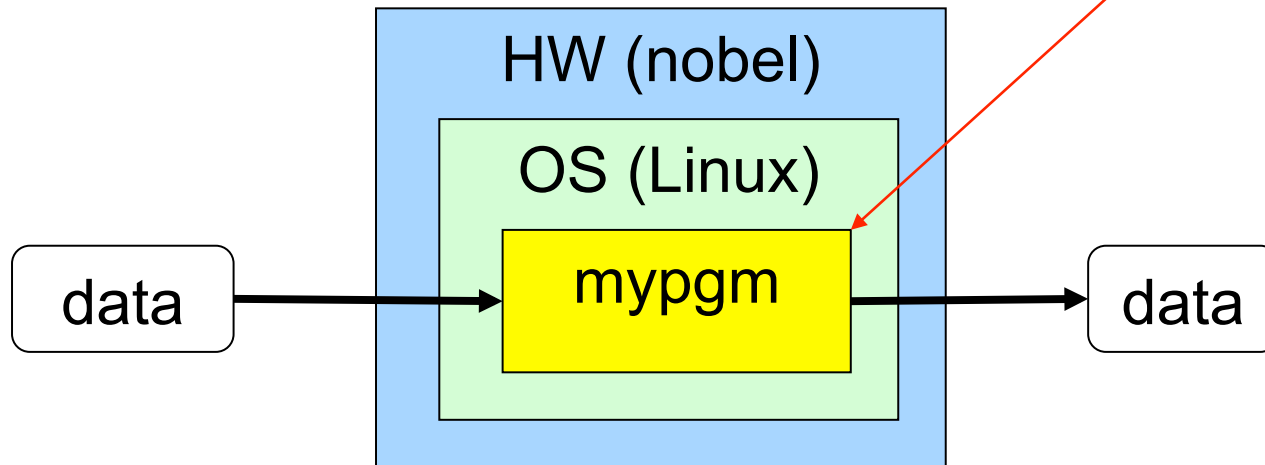




Running C Programs

`$ mypgm`

mypgm
(machine lang code)





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)



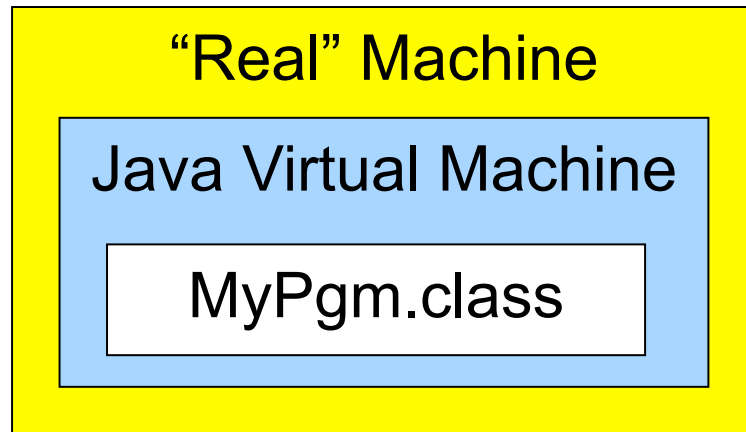
Java vs. C: Portability

Program	Code Type	Portable?
MyPgm.java	Java source code	Yes
mypgm.c	C source code	Mostly
MyPgm.class	Bytecode	Yes
mypgm	Machine lang code	No
javac (Java compiler)	Machine lang code	No
java (Java interpreter)	Machine lang code	No
gcc217 (C compiler driver)	Machine lang code	No

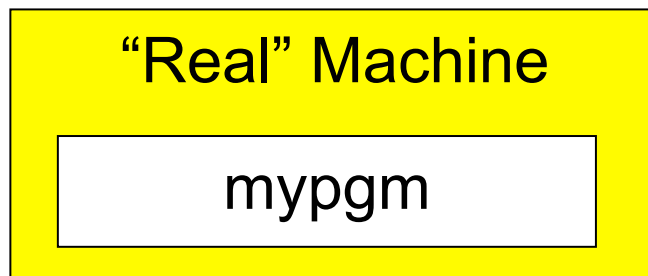
Conclusion: Java programs are more portable



Java vs. C: Efficiency



Java programs run on “virtual” machine which runs on “real” machine

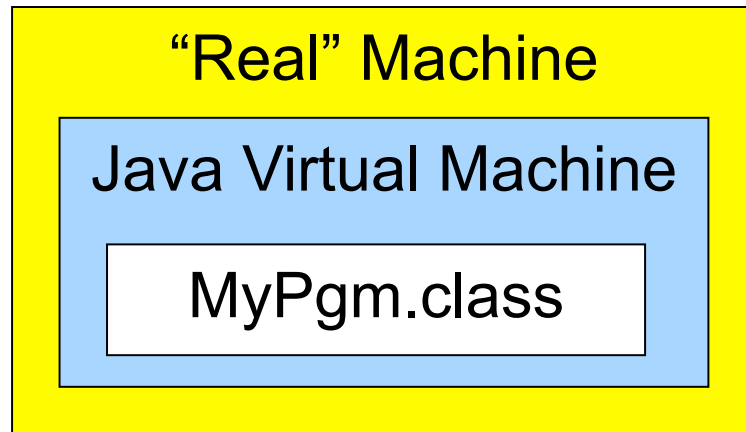


C programs run on “real” machine

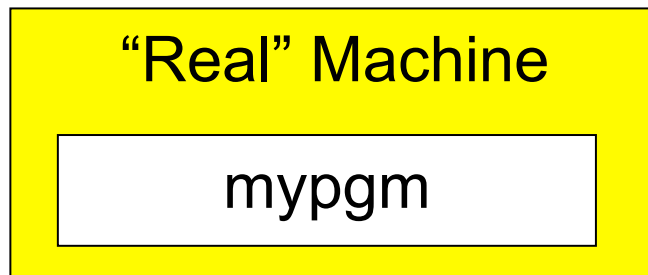
Conclusion: C programs are faster



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

	Java	C
Portability	+	-
Efficiency	-	+
Safety	+	-



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

	Java	C
Overall Program Structure	<pre>Hello.java: public class Hello { public static void main (String[] args) { System.out.println("hello, world"); } }</pre>	<pre>hello.c: #include <stdio.h> int main(void) { printf("hello, world\n"); return 0; }</pre>
Building	<pre>\$ javac Hello.java</pre>	<pre>\$ gcc217 hello.c -o hello</pre>
Running	<pre>\$ java Hello hello, world \$</pre>	<pre>\$ hello hello, world \$</pre>



Java vs. C: Details

	Java	C
Character type	<code>char // 16-bit Unicode</code>	<code>char /* 8 bits */</code>
Integral types	<code>byte // 8 bits</code> <code>short // 16 bits</code> <code>int // 32 bits</code> <code>long // 64 bits</code>	<code>(unsigned) char</code> <code>(unsigned) short</code> <code>(unsigned) int</code> <code>(unsigned) long</code>
Floating point types	<code>float // 32 bits</code> <code>double // 64 bits</code>	<code>float</code> <code>double</code> <code>long double</code>
Logical type	<code>boolean</code>	<code>/* no equivalent */</code> <code>/* use integral type */</code>
Generic pointer type	<code>// no equivalent</code>	<code>void*</code>
Constants	<code>final int MAX = 1000;</code>	<code>#define MAX 1000</code> <code>const int MAX = 1000;</code> <code>enum {MAX = 1000};</code>



Java vs. C: Details

	Java	C
Arrays	<pre>int [] a = new int [10]; float [][] b = new float [5][20];</pre>	<pre>int a[10]; float b[5][20];</pre>
Array bound checking	<pre>// run-time check</pre>	<pre>/* no run-time check */</pre>
Pointer type	<pre>// Object reference is an // implicit pointer</pre>	<pre>int *p;</pre>
Record type	<pre>class Mine { int x; float y; }</pre>	<pre>struct Mine { int x; float y; };</pre>



Java vs. C: Details

	Java	C
Strings	<pre>String s1 = "Hello"; String s2 = new String("hello");</pre>	<pre>char *s1 = "Hello"; char s2[6]; strcpy(s2, "hello");</pre>
String concatenation	<pre>s1 + s2 s1 += s2</pre>	<pre>#include <string.h> strcat(s1, s2);</pre>
Logical ops *	<pre>&&, , !</pre>	<pre>&&, , !</pre>
Relational ops *	<pre>=, !=, >, <, >=, <=</pre>	<pre>=, !=, >, <, >=, <=</pre>
Arithmetic ops *	<pre>+, -, *, /, %, unary -</pre>	<pre>+, -, *, /, %, unary -</pre>
Bitwise ops	<pre>>>, <<, >>>, &, , ^</pre>	<pre>>>, <<, &, , ^</pre>
Assignment ops	<pre>=, *=, /=, +=, -=, <<=, >>=, >>>=, =, &=, ^=, =, %=</pre>	<pre>=, *=, /=, +=, -=, <<=, >>=, =, &=, ^=, =, %=</pre>

* Essentially the same in the two languages



Java vs. C: Details

	Java	C
if stmt *	<pre>if (i < 0) statement1; else statement2;</pre>	<pre>if (i < 0) statement1; else statement2;</pre>
switch stmt *	<pre>switch (i) { case 1: ... break; case 2: ... break; default: ... }</pre>	<pre>switch (i) { case 1: ... break; case 2: ... break; default: ... }</pre>
goto stmt	// no equivalent	goto <i>someLabel</i> ;

* Essentially the same in the two languages



Java vs. C: Details

	Java	C
for stmt	<pre>for (int i=0; i<10; i++) statement;</pre>	<pre>int i; for (i=0; i<10; i++) statement;</pre>
while stmt *	<pre>while (i < 0) statement;</pre>	<pre>while (i < 0) statement;</pre>
do-while stmt *	<pre>do statement; while (i < 0)</pre>	<pre>do statement; while (i < 0);</pre>
continue stmt *	<pre>continue;</pre>	<pre>continue;</pre>
labeled continue stmt	<pre>continue someLabel;</pre>	<pre>/* no equivalent */</pre>
break stmt *	<pre>break;</pre>	<pre>break;</pre>
labeled break stmt	<pre>break someLabel;</pre>	<pre>/* no equivalent */</pre>

* Essentially the same in the two languages



Java vs. C: Details

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

* Essentially the same in the two languages



Example C Program

```
#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”
 - 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