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

- 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

Agenda

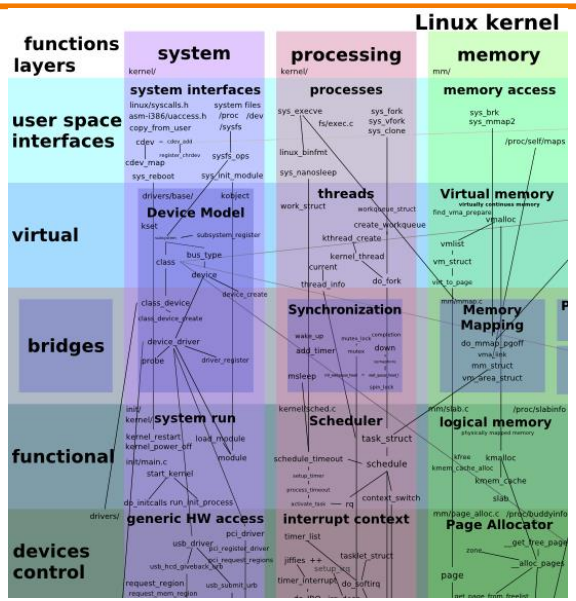
Course overview

- 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



Goal 1: "Programming in the large"

- Help you learn how to write large computer programs

Topics

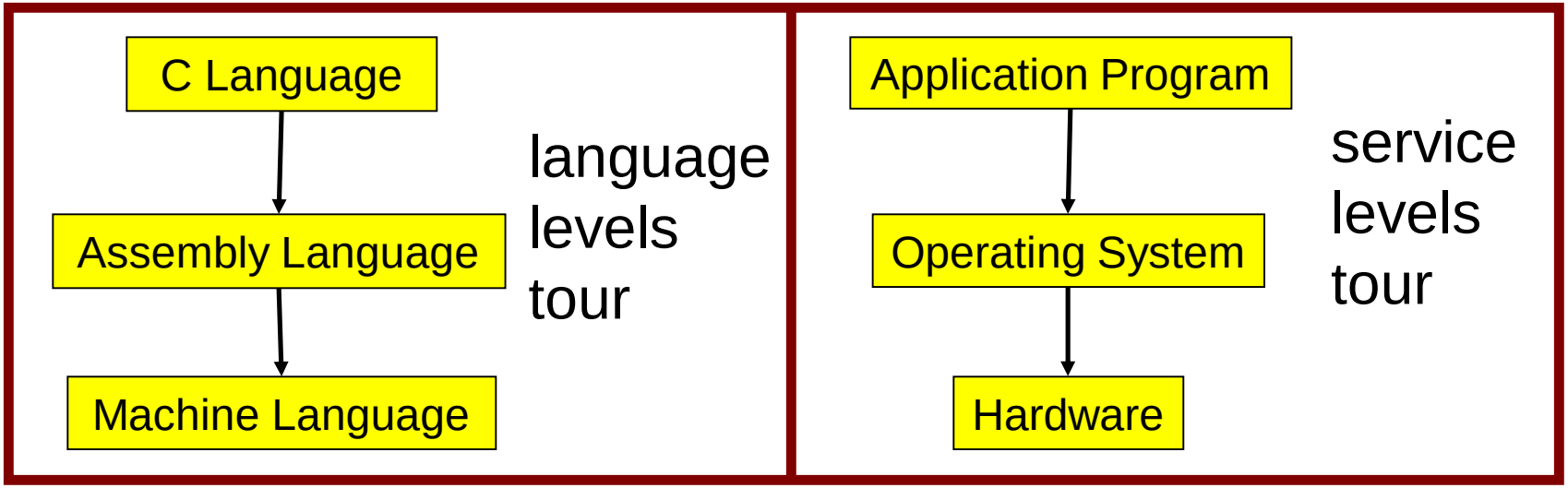
- Modularity/abstraction, information hiding, resource management, error handling, testing, debugging, performance improvement, tool support

Goal 2: Under the Hood

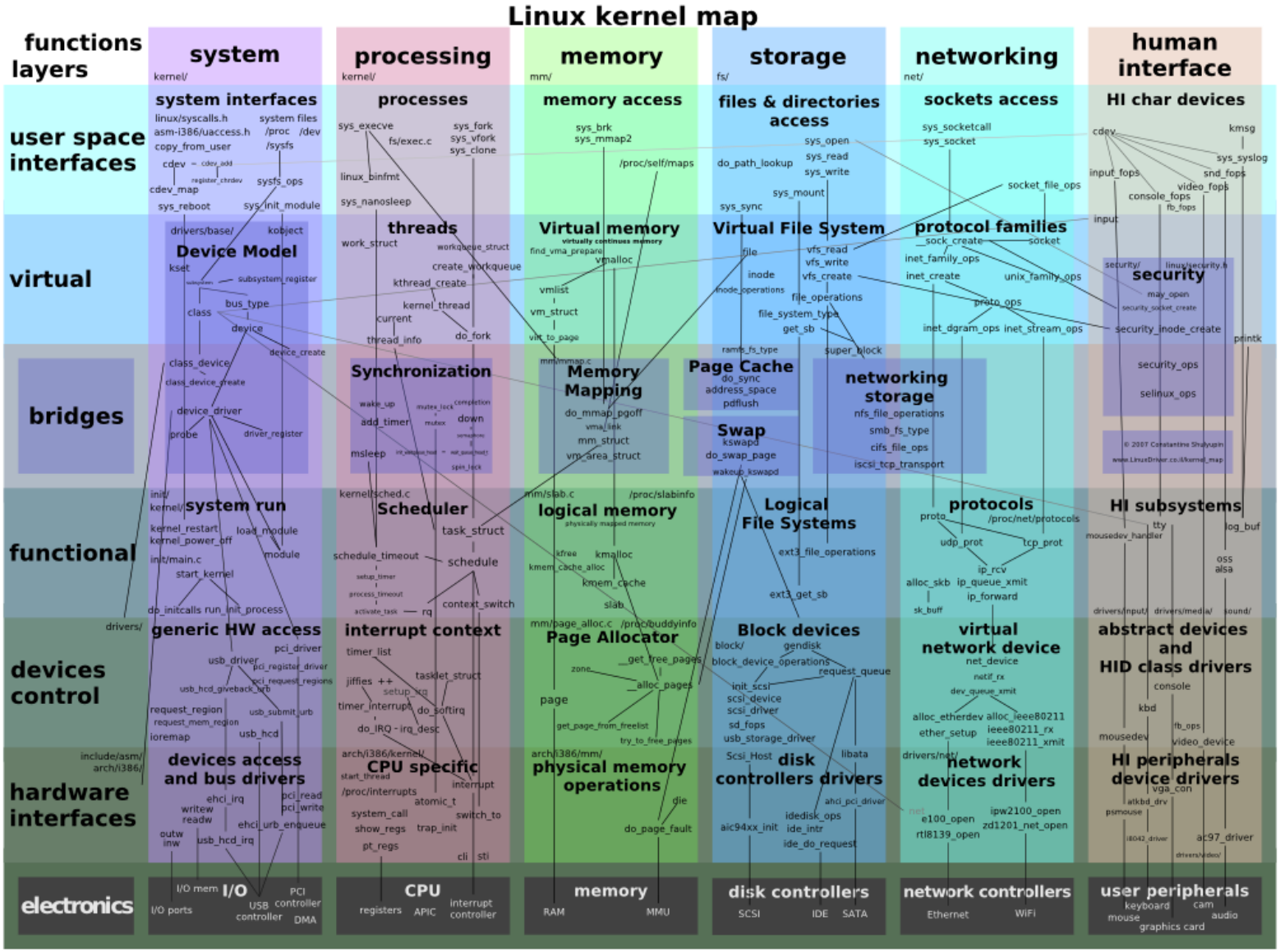
Learn what happens "under the hood" of computer systems

Learn "how to be a client of an operating system"

Downward tours



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

Psychological SCIENCE
A Journal of the Association for Psychological Science

The Pen Is Mightier Than the Keyboard

Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller[1]

Daniel M. Oppenheimer[2]

[1]Princeton University
[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

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 Monday

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#fall2017/cos217/
- Instructions provided in first precept

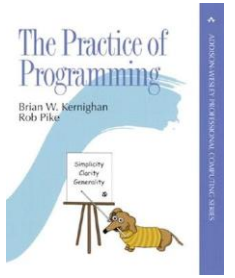
Piazza etiquette

- 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

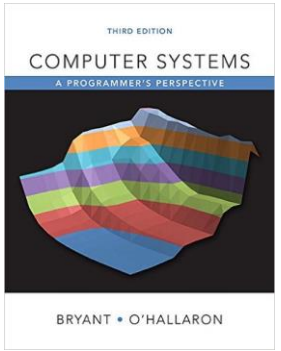
The Practice of Programming (recommended)

- Kernighan & Pike
- "Programming in the large"



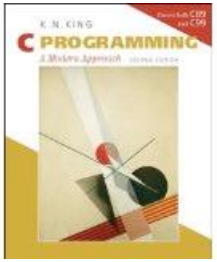
Computer Systems: A Programmer's Perspective (Third Edition) (recommended)

- Bryant & O'Hallaron
- "Under the hood"



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

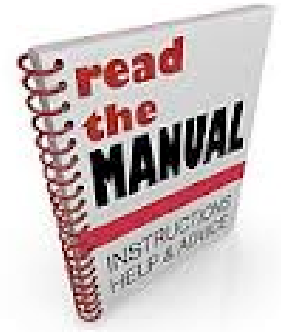
- King
- C programming language and standard libraries



Manuals

Manuals (for reference only, available online)

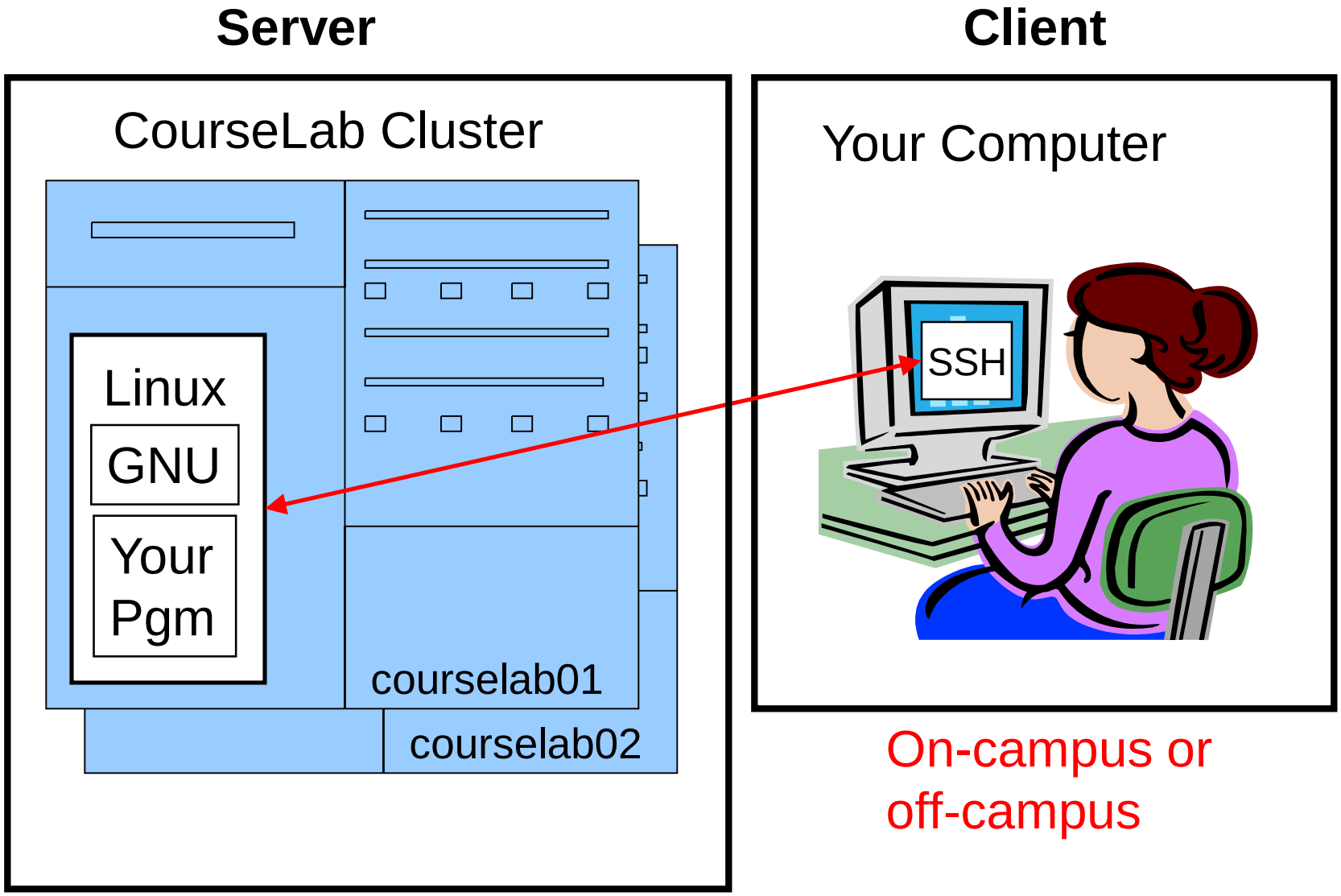
- ***Intel 64 and IA-32 Architectures Software Developer's Manual, Volumes 1-3***
- ***Intel 64 and IA-32 Architectures Optimization Reference Manual***
- ***Using `as`, the GNU Assembler***



See also

- Linux **`man`** command

Programming Environment



Agenda

Course overview

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

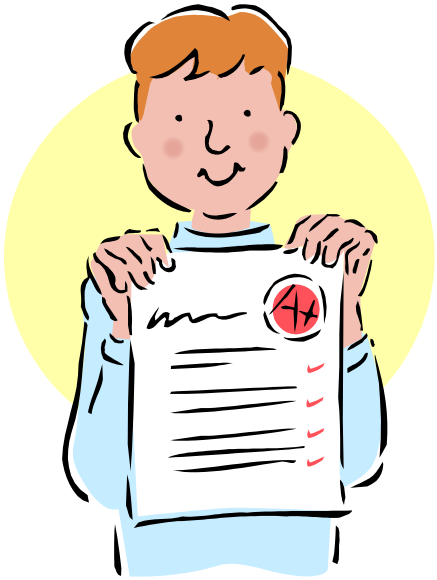
Getting started with C

- 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

These percentages are approximate



* 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

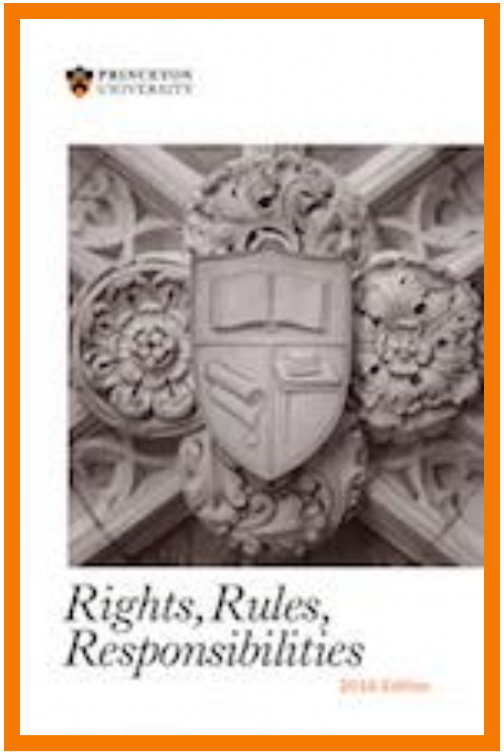
- Introductions
- Course goals
- Resources
- Grading
- **Policies**
- Schedule

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

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

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

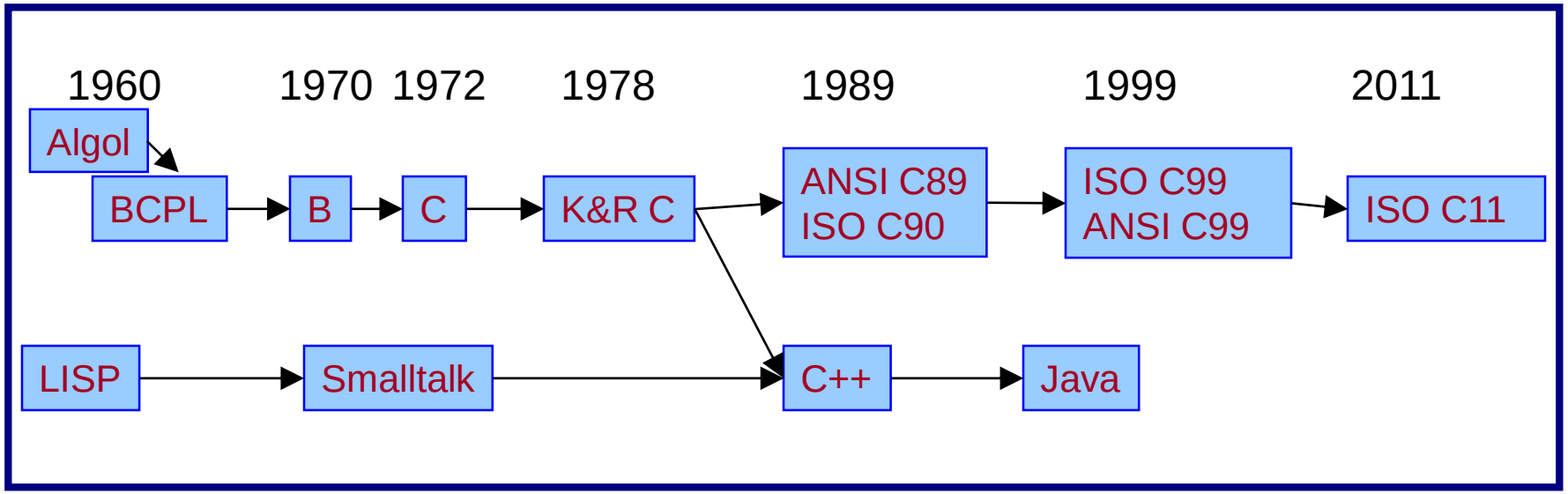
Who? Dennis Ritchie

When? ~1972

Where? Bell Labs

Why? Compose the Unix OS

Java vs. C: History



C vs. Java: Design Goals

C Design Goals (1975)	Java Design Goals (1995)
Build the Unix OS	Language of the Internet
Low-level; close to HW and OS	High-level; insulated from hardware and OS
Good for system-level programming	Good for application-level programming
Support structured programming	Support object-oriented programming
Unsafe: don't get in the programmer's way	Safe: can't step "outside the sandbox"
	Look like C!

Agenda

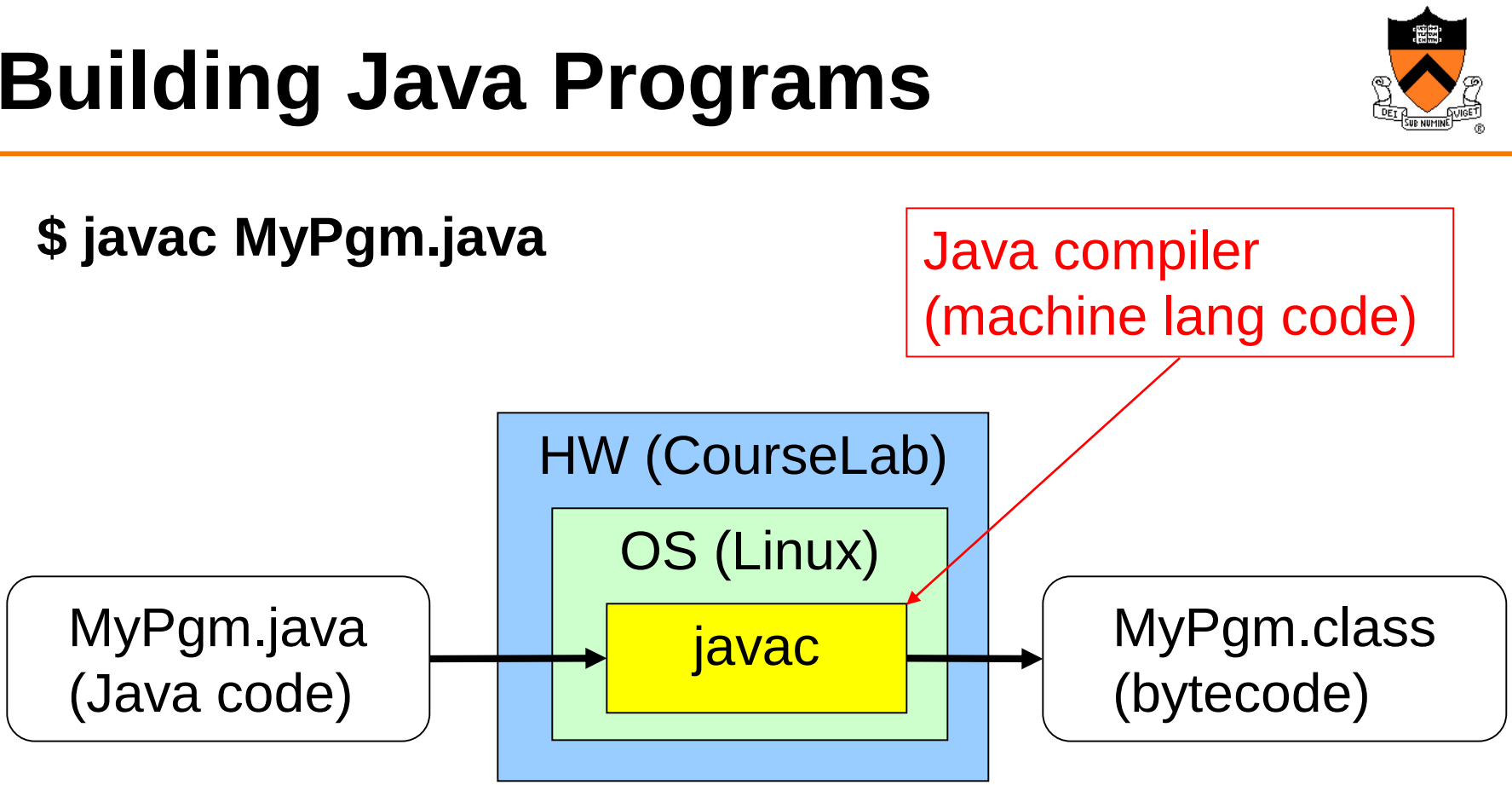
Course overview

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

Getting started with C

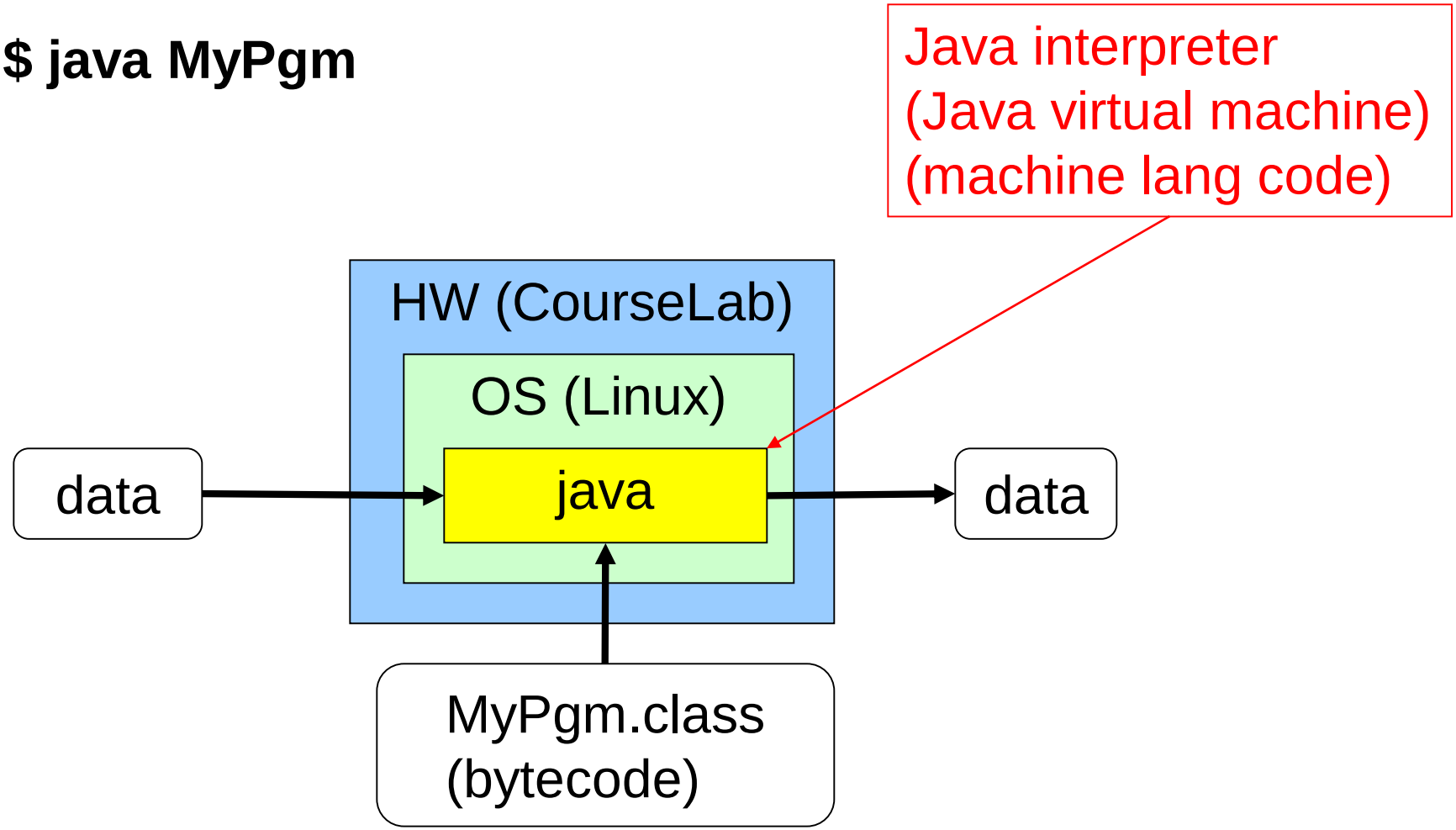
- History of C
- **Building and running C programs**
- Characteristics of C
- C details (if time)

Building Java Programs



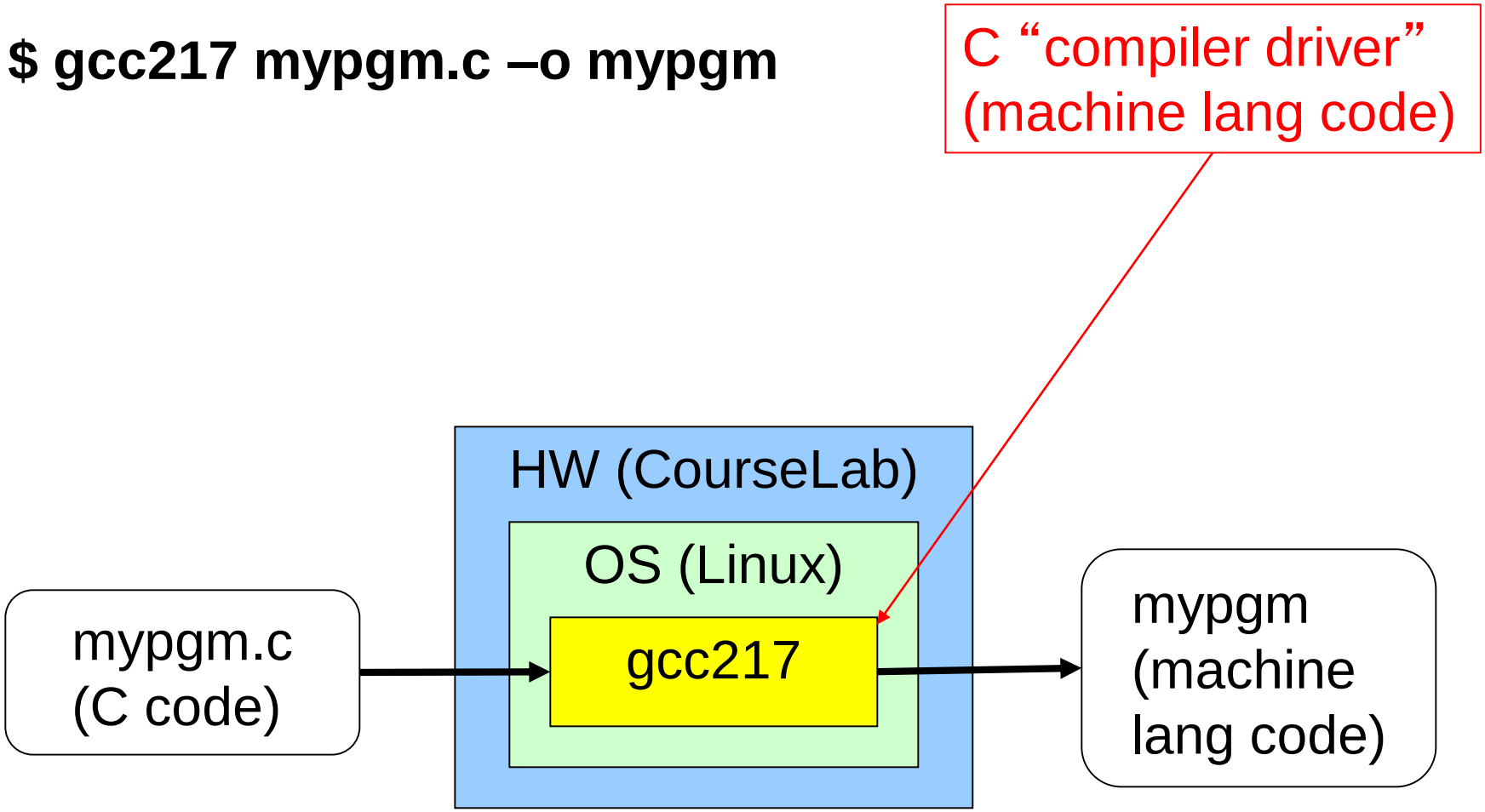
Running Java Programs

$ java MyPgm



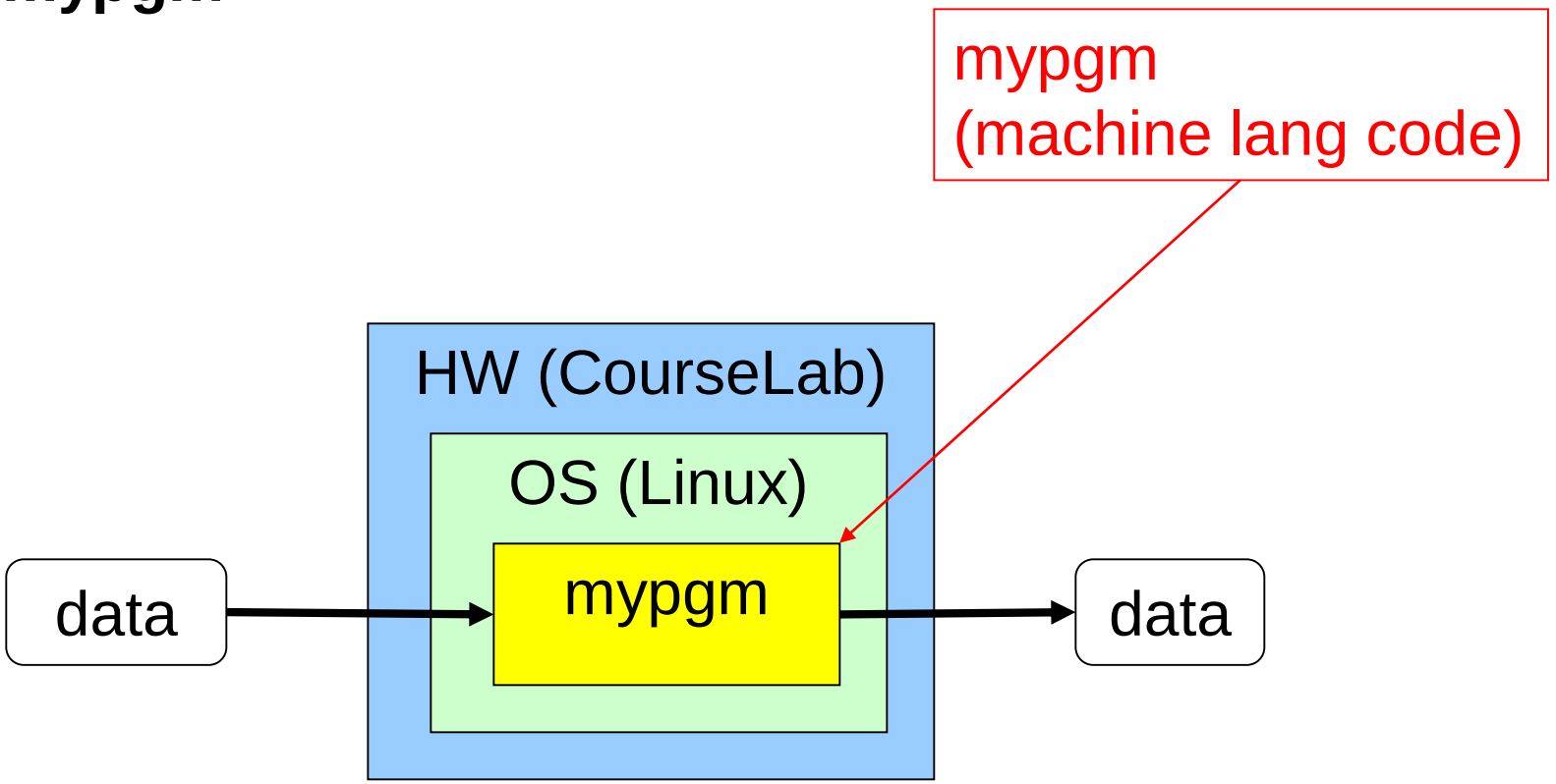
Building C Programs

$ gcc217 mypgm.c –o mypgm



Running C Programs

$ ./mypgm



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

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

	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	`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

	Java	C
Character type	`char // 16-bit Unicode`	`char /* 8 bits */`

Integral types	`byte // 8 bits`
	`short // 16 bits`
	`int // 32 bits`
	`long // 64 bits`
		`(unsigned) char`
		`(unsigned) short`
		`(unsigned) int`
		`(unsigned) long`

Floating point types	`float // 32 bits`
	`double // 64 bits`
		`float`
		`double`
		`long double`

Logical type	`boolean`	`/* no equivalent */` `/* use integral type */`
Generic pointer type	`Object`	`void*`

Constants	`final int MAX = 1000;`	`#define MAX 1000`
		`const int MAX = 1000;`
		`enum {MAX = 1000};`

Java vs. C: Details

	Java	C

Arrays	`int [] a = new int [10];`
	`float [][] b =`
	`  new float [5][20];`
		`int a[10];`
		`float b[5][20];`

Array bound checking	`// run-time check`	`/* no run-time check */`
Pointer type	`// Object reference is an` `// implicit pointer`	`int *p;`

Record type	`class Mine`
	`{  int x;`
	`  float y;`
	`}`
		`struct Mine`
		`{  int x;`
		`  float y;`
		`};`

Java vs. C: Details

	Java	C

Strings	`String s1 = "Hello";`
	`String s2 = new`
	`  String("hello");`
		`char *s1 = "Hello";`
		`char s2[6];`
		`strcpy(s2, "hello");`

String concatenation	`s1 + s2` `s1 += s2`	`#include <string.h>` `strcat(s1, s2);`
Logical ops *	`&&, ||, !`	`&&, ||, !`
Relational ops *	`=, !=, >, <, >=, <=`	`=, !=, >, <, >=, <=`
Arithmetic ops *	`+, -, *, /, %, unary -`	`+, -, *, /, %, unary -`
Bitwise ops	`>>, <<, >>>, &, |, ^`	`>>, <<, &, |, ^`
Assignment ops	`=, *=, /=, +=, -=, <<=, >>=, >>>=, =, &=, ^=, |=, %=`	`=, *=, /=, +=, -=, <<=, >>=, =, &=, ^=, |=, %=`

* Essentially the same in the two languages

Java vs. C: Details

	Java	C

if stmt *	`if (i < 0)`
	`  statement1;`
	`else`
	`  statement2;`
		`if (i < 0)`
		`  statement1;`
		`else`
		`  statement2;`

switch stmt *	`switch (i)`
	`{ case 1:`
	`  ...`
	`  break;`
	`  case 2:`
	`  ...`
	`  break;`
	`  default:`
	`  ...`
	`}`
		`switch (i)`
		`{ case 1:`
		`  ...`
		`  break;`
		`  case 2:`
		`  ...`
		`  break;`
		`  default:`
		`  ...`
		`}`

goto stmt	`// no equivalent`	`goto someLabel;`

* Essentially the same in the two languages

Java vs. C: Details

	Java	C

for stmt	`for (int i=0; i<10; i++)`
	`  statement;`
		`int i;`
		`for (i=0; i<10; i++)`
		`  statement;`

while stmt *	`while (i < 0)` `  statement;`	`while (i < 0)` `  statement;`

do-while stmt *	`do`
	`  statement;`
	`while (i < 0)`
		`do`
		`  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

	Java	C
return stmt *	`return 5;` `return;`	`return 5;` `return;`

Compound stmt (alias block) *	`{`
	`  statement1;`
	`  statement2;`
	`}`
		`{`
		`  statement1;`
		`  statement2;`
		`}`

Exceptions	`throw, try-catch-finally`	`/* no equivalent */`
Comments	`/* comment */` `// another kind`	`/* comment */`

Method / function call	`f(x, y, z);`
	`someObject.f(x, y, z);`
	`SomeClass.f(x, y, z);`
		`f(x, y, z);`

* 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"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