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

int main(int argc, char *argv[])
{
   printf(“Welcome to COS 217\n");
   printf("Introduction to Programming Systems\n\n");

   printf("%s %d\n", "Spring", 2019);
   return 0;
}

$ cat Makefile
CC=gcc217
welcome: welcome.o

$ make
gcc217    -c -o welcome.o welcome.c
gcc217   welcome.o   -o welcome

$ ./welcome
Welcome to COS 217
Introduction to Programming Systems

Spring, 2019
```

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 Preceptors

- Robert Dondero, Ph.D. — rdondero@cs.princeton.edu
- Xiaoyan Li — xiaoyan@cs.princeton.edu

Graduate Student Preceptors

- James Heppenstall — jwmh@princeton.edu
- Seo Young Kyung — skyung@princeton.edu
- Josh Zhang — jiashuoz@princeton.edu

Agenda

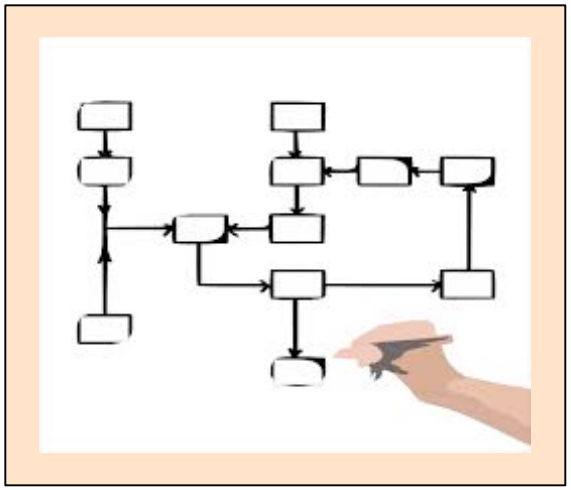
Course overview

- 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



Learn how to compose
large computer programs

Topics

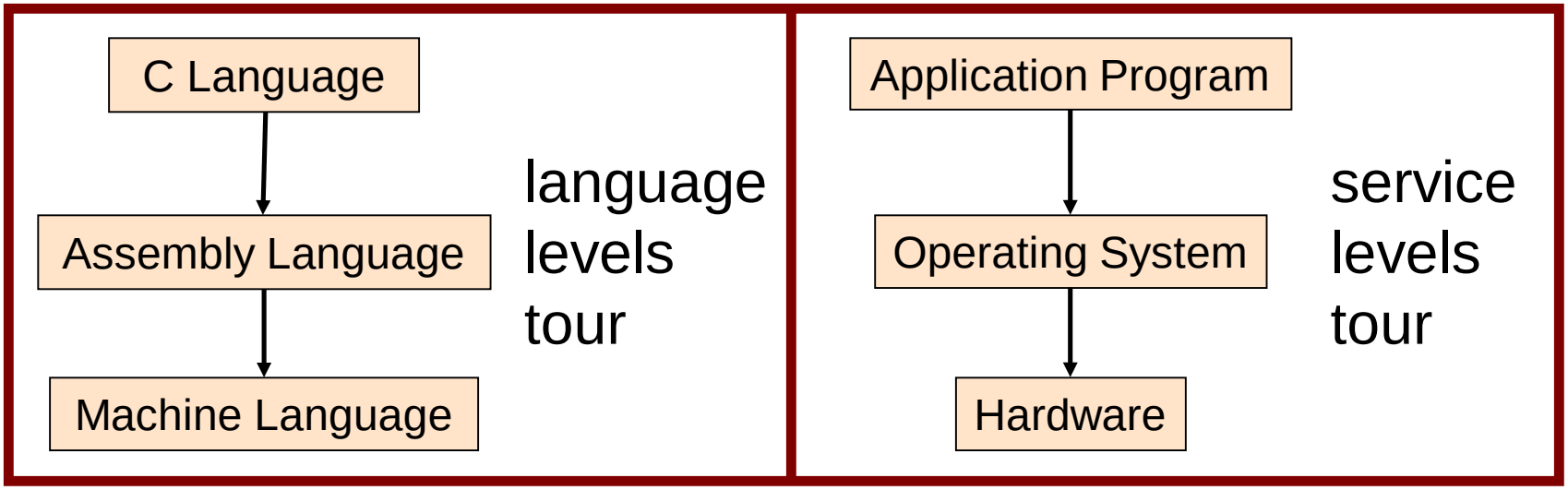
- Modularity/abstraction, information hiding, resource management, error handling, testing, debugging, performance improvement, tool support

Goal 2: Under the Hood

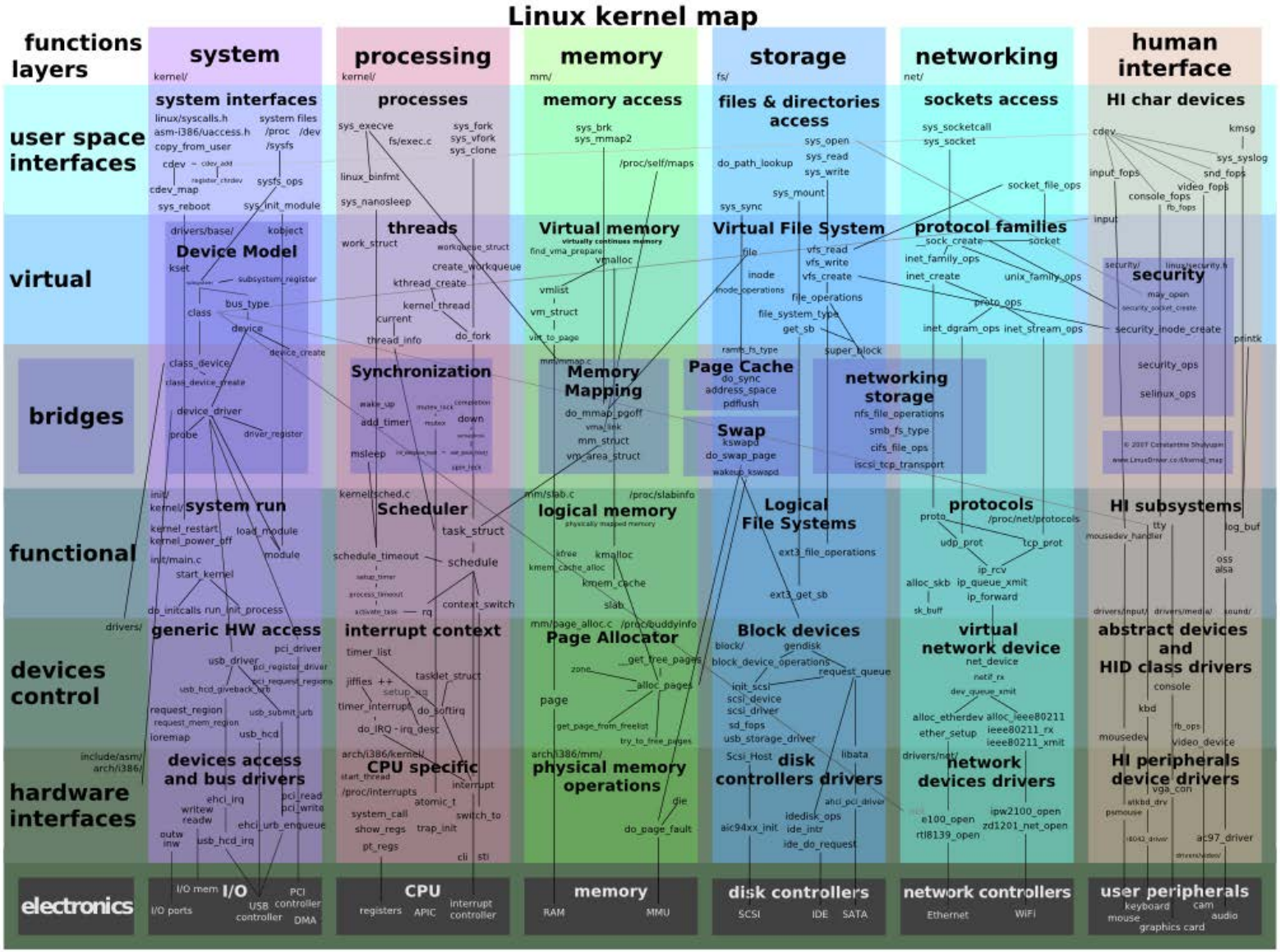
Learn what happens "under the hood" of computer systems

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

Downward tours



Modularity!


Linux kernel map
functions
layers
system
processing
memory
storage
networking
human interface
user space interfaces
virtual
bridges
functional
devices control
hardware interfaces
system interfaces
processes
memory access
files & directories access
sockets access
HI char devices
Device Model
threads
Virtual memory
Virtual File System
protocol families
security
Synchronization
Memory Mapping
Page Cache
Swap
networking storage
System run
Scheduler
logical memory
Logical File Systems
protocols
HI subsystems
generic HW access
interrupt context
Page Allocator
Block devices
virtual network device
abstract devices and HID class drivers
devices access and bus drivers
CPU specific
physical memory operations
disk controllers drivers
network devices drivers
HI peripherals device drivers
electronics
I/O
CPU
memory
disk controllers
network controllers
user peripherals

Goals: Summary

Help you to become a...

Power Programmer!!!

Specific Goal: Learn C

Question: Why C instead of Java?

Answer 1: A primary language for "under the hood" programming

Answer 2: Knowing a variety of approaches helps you "program in the large"

Specific Goal: Learn Linux

Question: Why use the Linux operating system?

Answer 1: Linux is the industry standard for servers, embedded devices, education, and research

Answer 2: Linux (with GNU tools) is good for programming (which helps 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

Etiquette

- Use electronic devices *only* for taking notes or annotating slides
  (but consider taking notes by hand – research shows it works better!)
- No SnapFaceNewsBookInstaGoo, please

iClicker

- 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 – it's a violation of our course policies!)

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

Etiquette

- Attend your precept – attendance will be taken
  - Must miss your precept? ⇒ inform preceptors & attend another
- Use SCORE to move to another precept
  - Trouble ⇒ See Colleen Kenny (CS Bldg 210)
  - But Colleen can’t move you into a full precept

Precepts begin today and tomorrow!

Website

http://www.cs.princeton.edu/~cos217/

- Home page, schedule page, assignment page, policies page

Piazza

Piazza

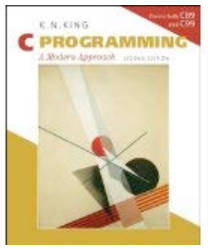
- http://piazza.com/princeton/spring2019/cos217
- Instructions provided in first precept

Piazza etiquette

- Study provided material before posting question
  - Lecture slides, precept handouts, required readings
- Read / search all (recent) Piazza threads before posting question
- Don't reveal your code!
  - See course policies

Books

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

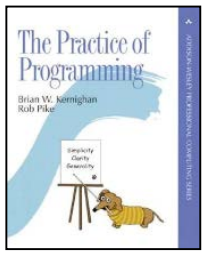


- King
- C programming language and standard libraries

ARM 64-bit Assembly Language (required)

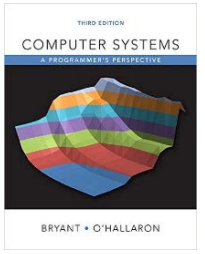
- Pyeatt & Ughetta
- Preprint will be made available through Pequod

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”

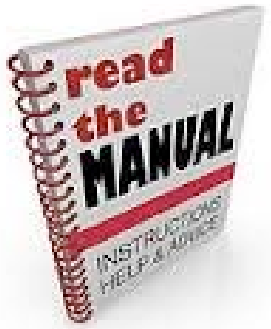
Manuals

Manuals (for reference only, available online)

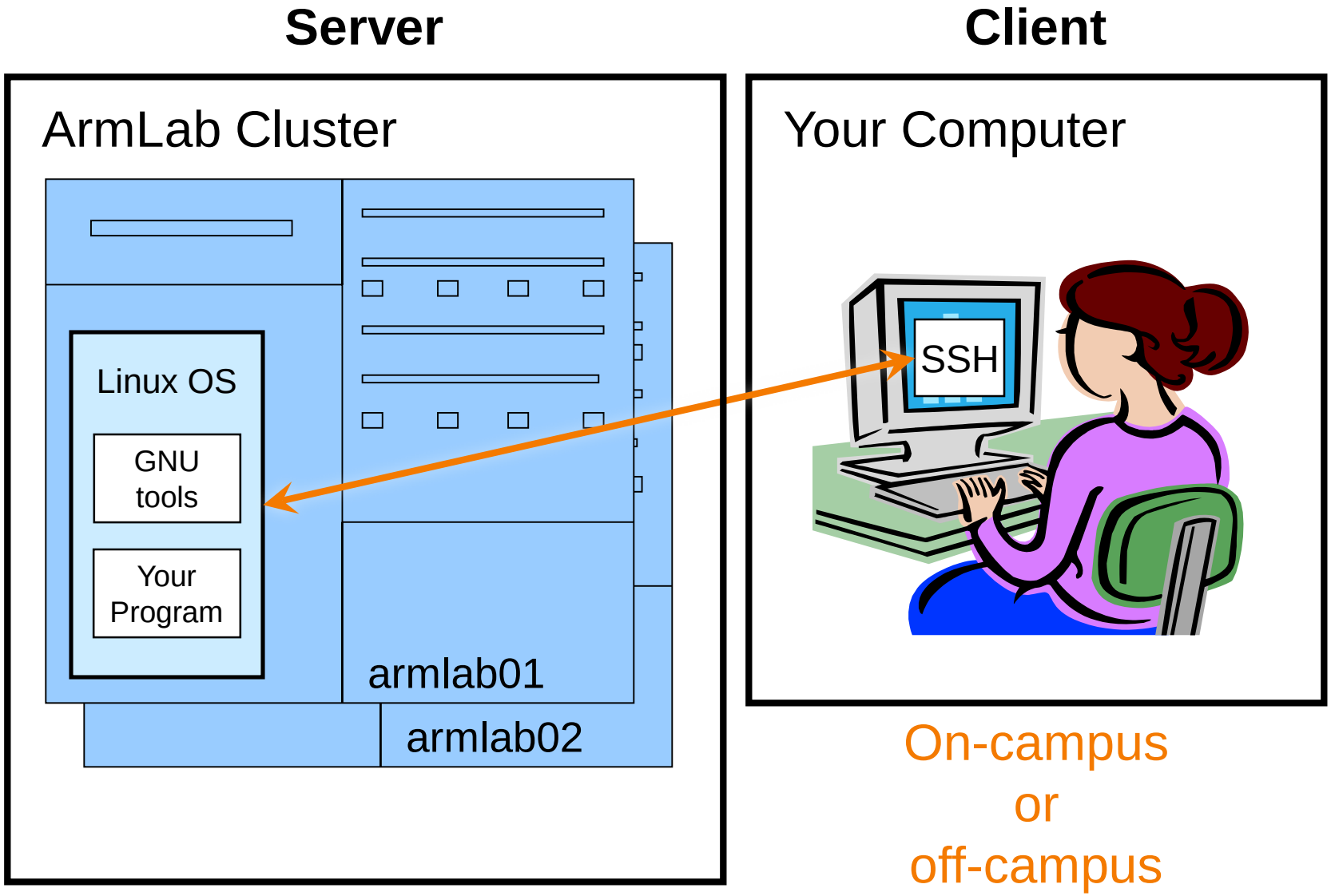
- ***ARMv8 Instruction Set Overview***
- ***ARM Architecture Reference Manual***
- ***Using as, the GNU Assembler***

See also

- Linux ***man*** command



Programming Environment



Agenda

Course overview

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

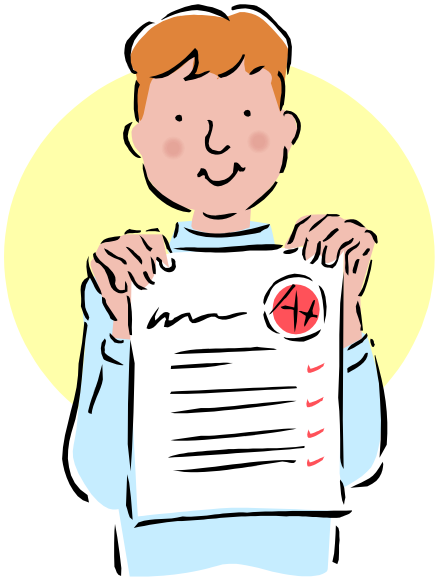
Getting started with C

- 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
Participation ***	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?

- Lecture/precept attendance and participation counts

Programming Assignments

Regular (not-quite-weekly) assignments

0. Introductory survey
1. "De-comment" program
2. String module
3. Symbol table module
4. Assembly language programs
5. Buffer overrun attack
6. Heap manager module
7. Unix shell

*(some individual, some done with a partner from your precept)

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

Learning is a collaborative activity!

- Discussions with others that help you understand concepts from class are encouraged

But programming assignments are graded!

- Everything that gets submitted for a grade must be exclusively your own work
- Don't look at code from someone else, the web, Github, etc. – **see the course "Policies" web page**
- Don't reveal your code or design decisions to anyone except course staff – **see the course "Policies" web page**

Violations of course policies

- 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 (including after the semester is over), 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

Weeks	Lectures	Precepts
1-2	C (conceptual) Number Systems	C (pragmatic) Linux/GNU
3-6	Programming in the Large	Advanced C
6	Midterm Exam	
7	Spring break!	
8-13	"Under the Hood" (conceptual)	"Under the Hood" (assignment how-to)
	Reading Period	
	Final Exam	

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)

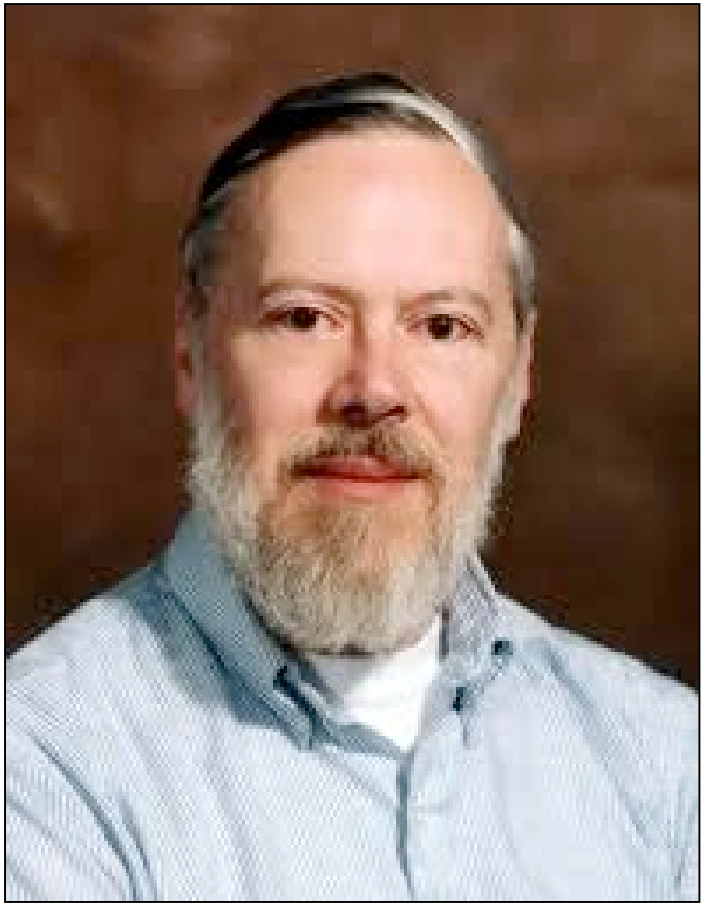
The C Programming Language

Who? Dennis Ritchie

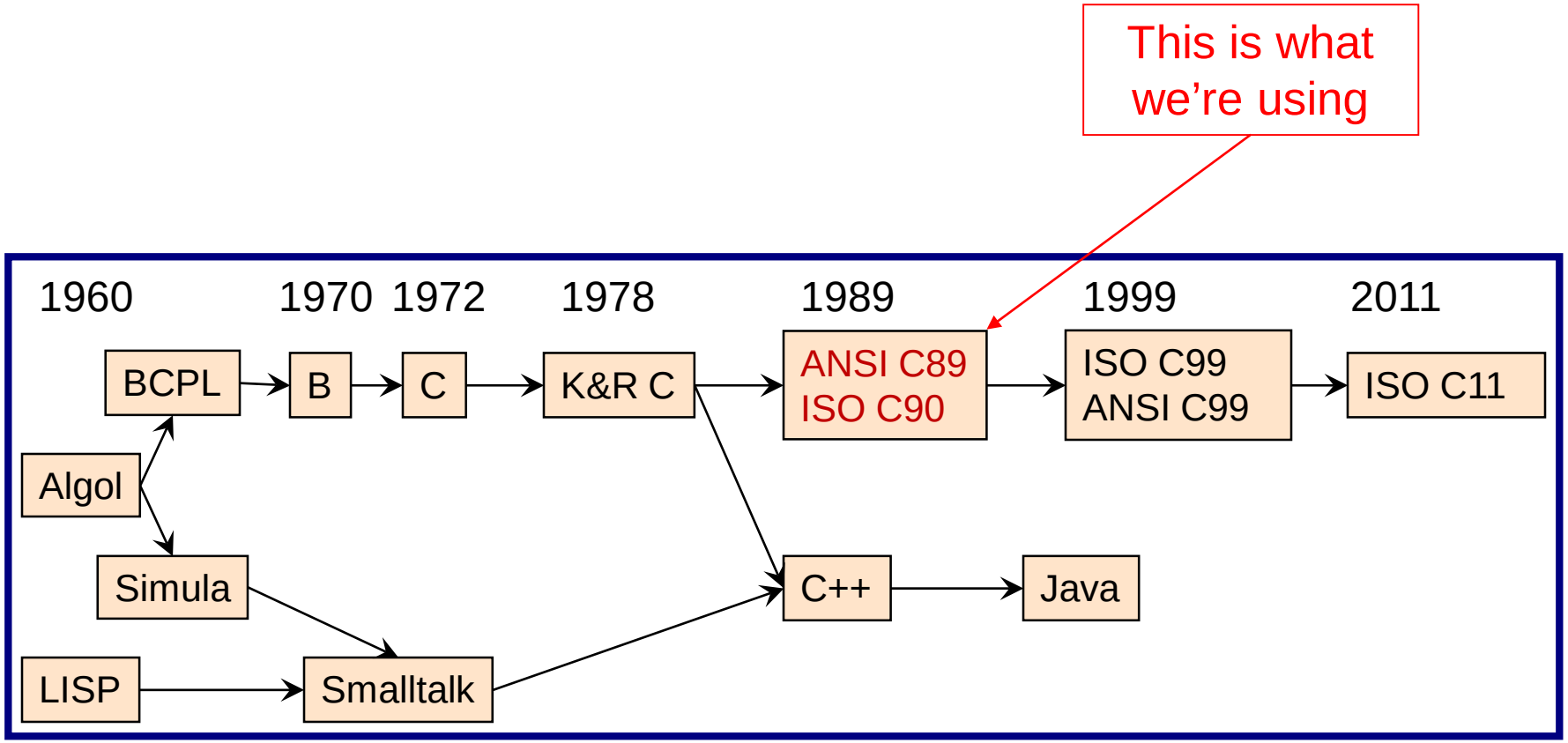
When? ~1972

Where? Bell Labs

Why? Build the Unix OS



Java vs. C: History



C vs. Java: Design Goals

C Design Goals (1972)	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

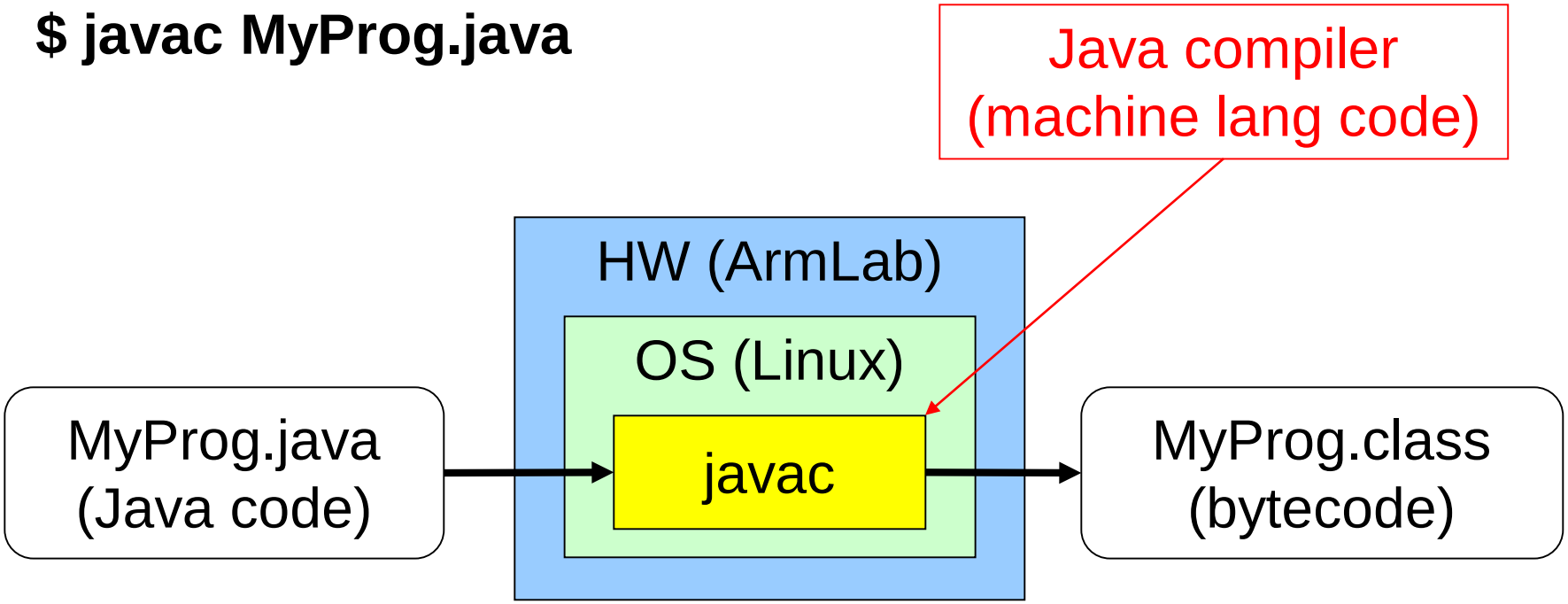
Course overview

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

Getting started with C

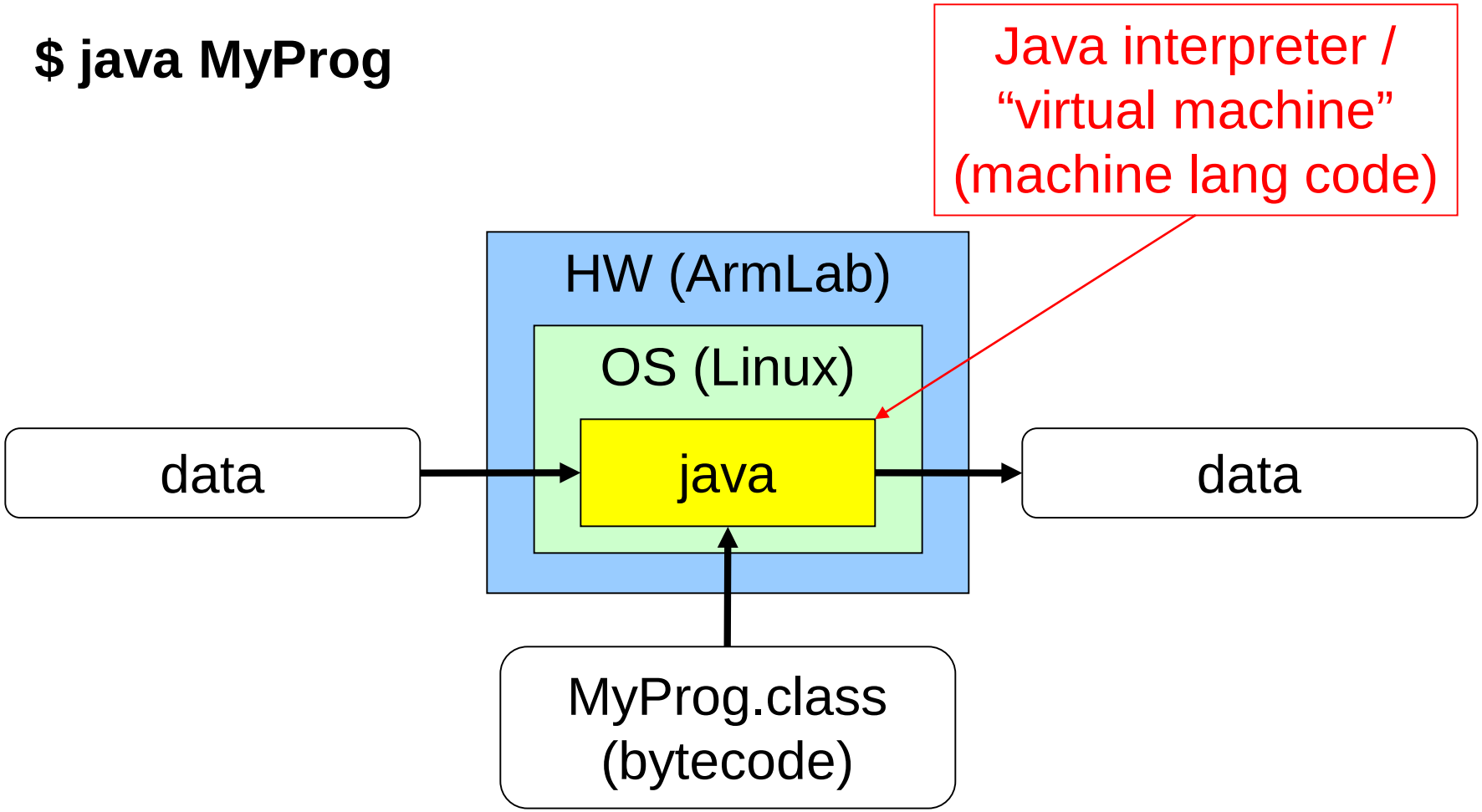
- History of C
- **Building and running C programs**
- Characteristics of C
- C details (if time)

Building Java Programs



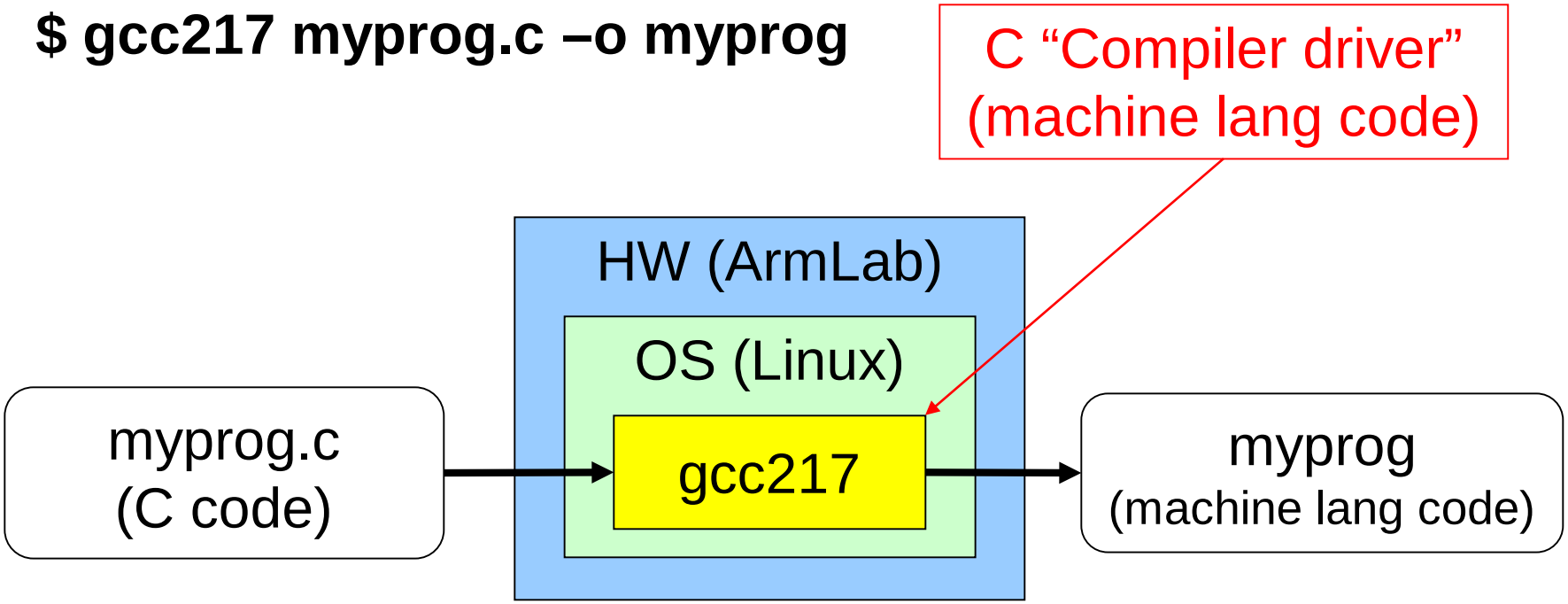
Running Java Programs

$ java MyProg

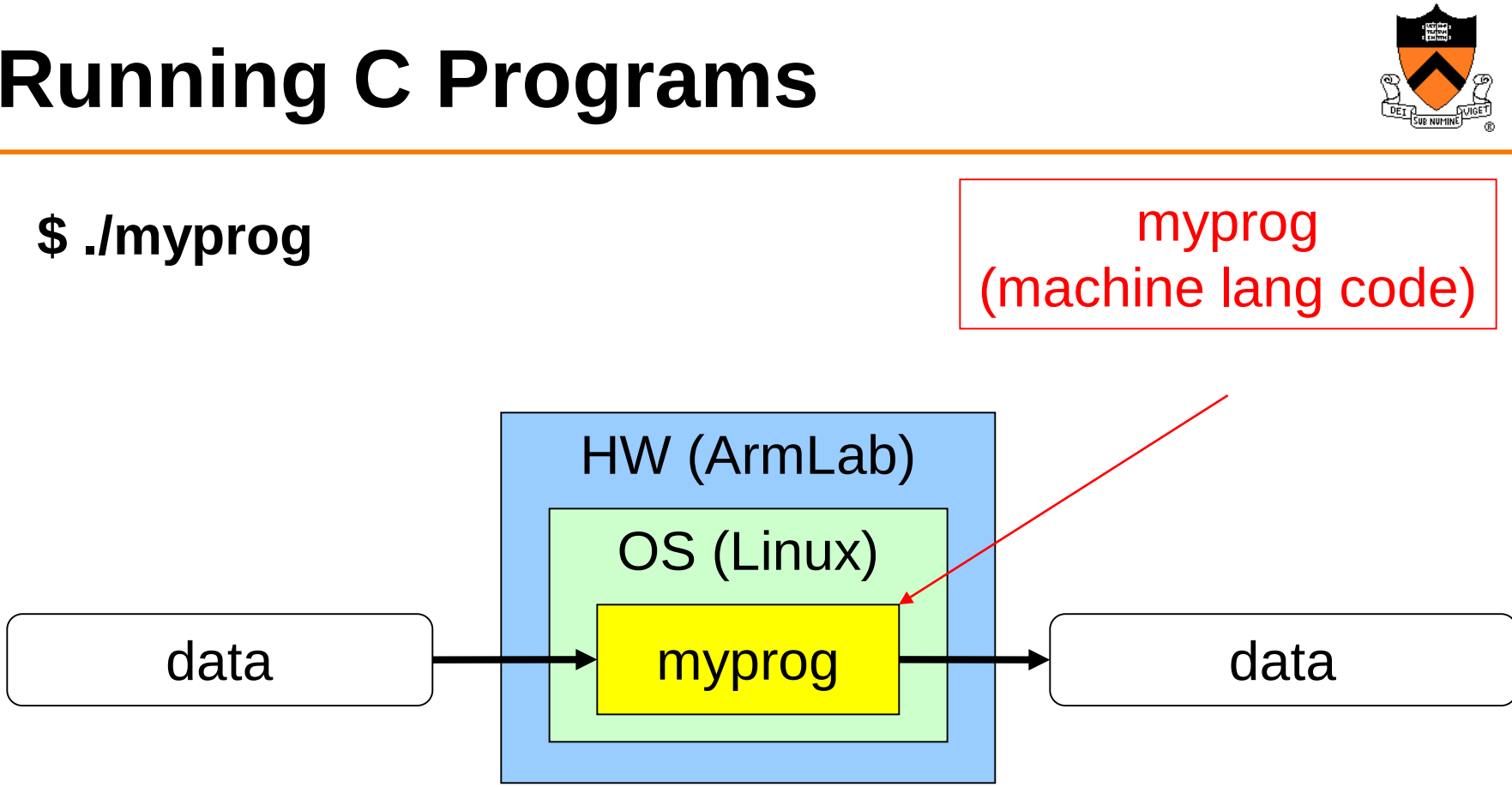


Building C Programs

$ gcc217 myprog.c –o myprog



Running C Programs



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?
MyProg.java	Java source code	Yes
myprog.c	C source code	Mostly
MyProg.class	Bytecode	Yes
myprog	Machine lang code	No

Conclusion: Java programs are more portable

(In particular, this semester we're moving from the x86_64-based "courselab" to the ARM64-based "armlab", and all of the programs must be recompiled!)

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

	Java	C
Portability	+	-
Efficiency	-	+
Safety	+	-

iClicker Question

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

	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, signed) char`
		`(unsigned, signed) short`
		`(unsigned, signed) int`
		`(unsigned, signed) long`

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

Logical type	`boolean`	`/* no equivalent */` `/* use 0 and non-0 */`
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