```
$ 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", 2022);
   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 2022
```

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

Introductions

Lead Faculty

- Aarti Gupta aartig@cs.princeton.edu

Lead Preceptor

- Christopher Moretti cmoretti@cs.princeton.edu

Preceptors

- Cedrick Argueta argueta@princeton.edu
- Huihan Li huihanl@princeton.edu
- Maxine Perroni-Scharf mp4215@princeton.edu

Agenda

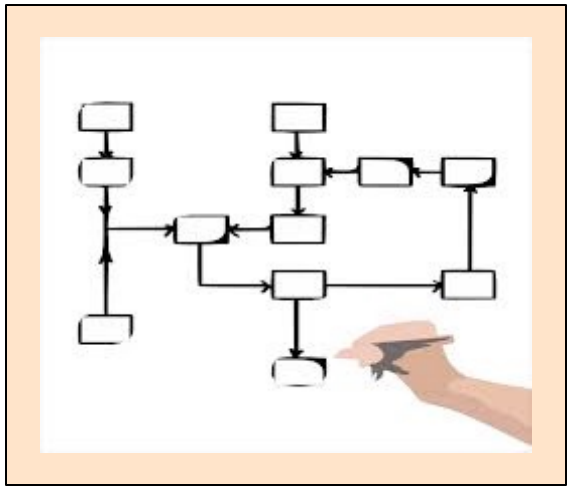
Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

Goal 1: Programming in the Large

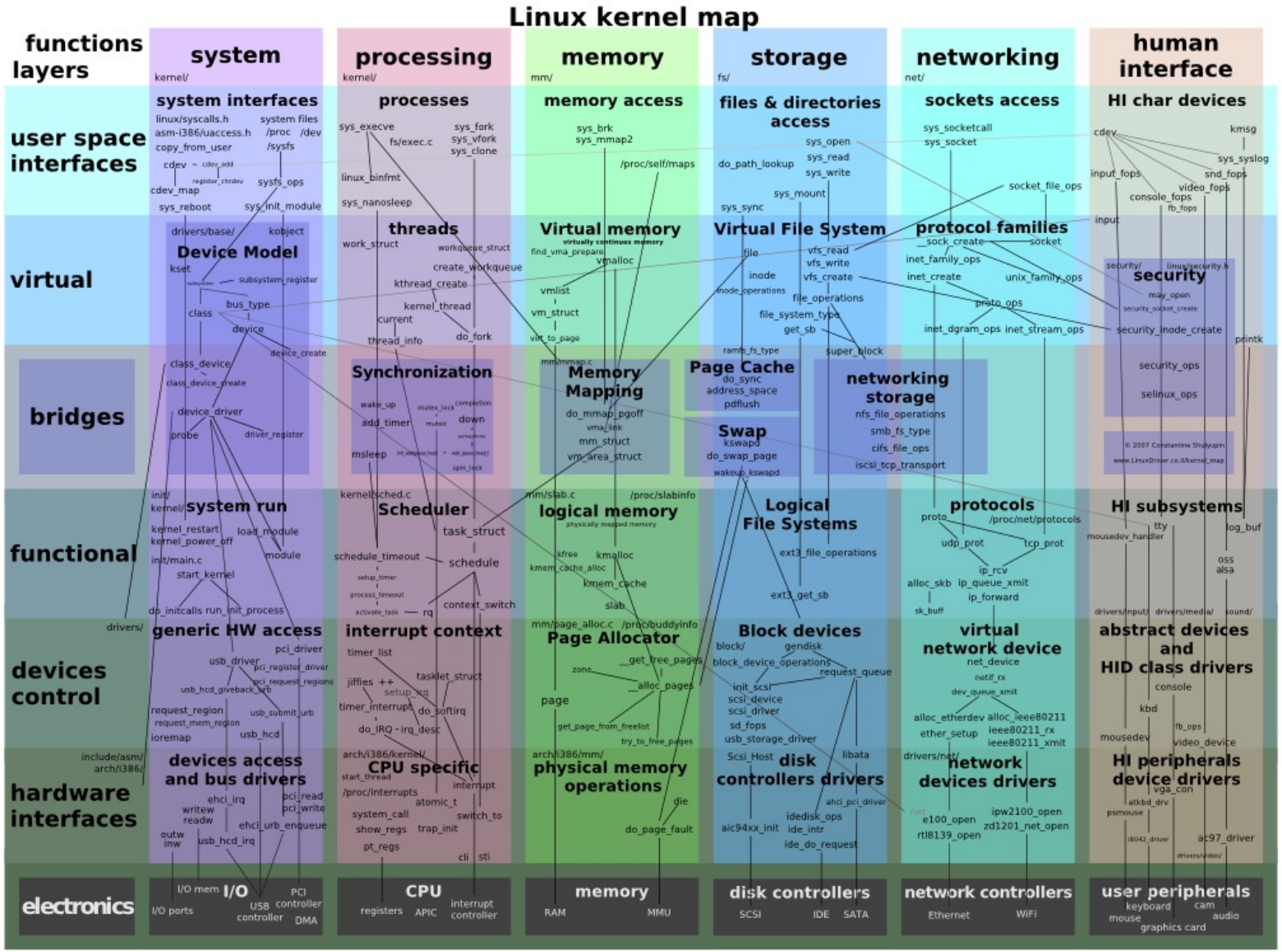


Learn how to compose
large(r) computer programs

Topics

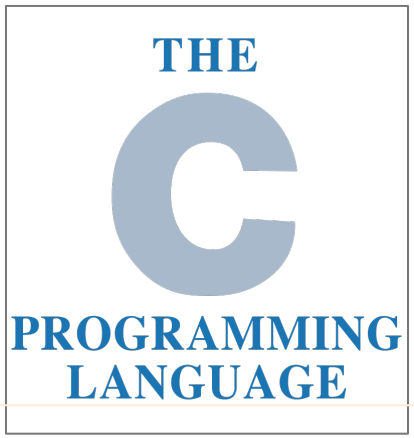
- Modularity/abstraction, information hiding, resource management, error handling, testing, debugging, performance improvement
- Tools: ssh, bash, shell utilities, emacs, git, gcc, make, gdb, gprof, valgrind

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
networking storage
Swap
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

Goal 2: Lower-level Languages

```
int main(void) {
   while ((iChar = getchar()) != EOF) {
      lCharCount++;
      if (isspace(iChar)) {
         if (iInWord) {
            lWordCount++;
            iInWord = FALSE;
         }
      }
```



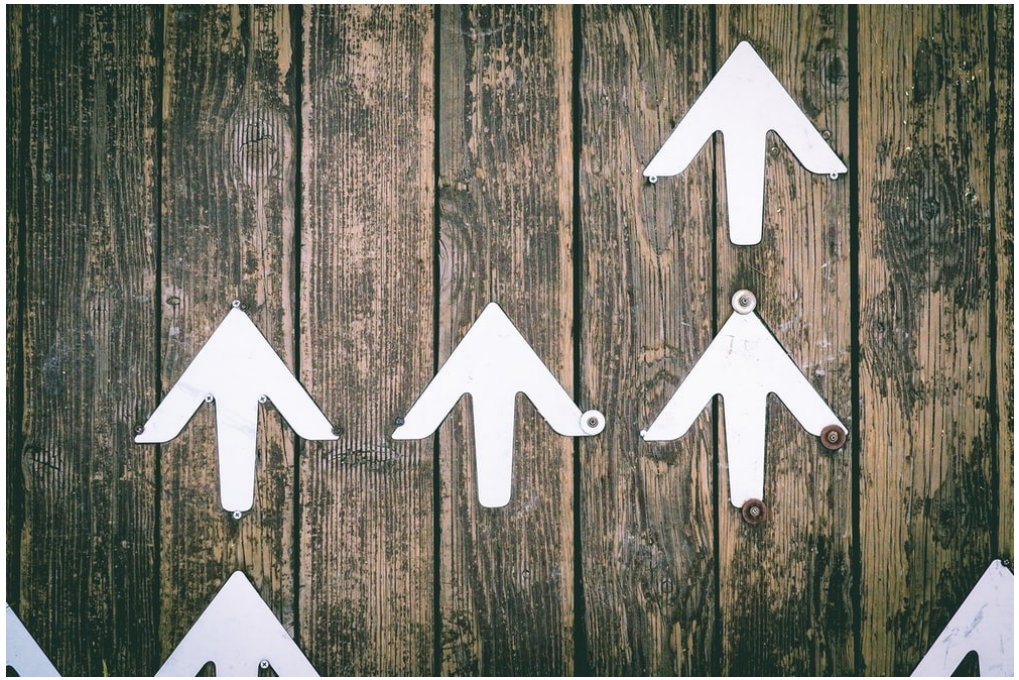

```
main:
.LFB0:
.cfi_startproc
stp x29, x30, [sp, -16]!
.cfi_def_cfa_offset 16
.cfi_offset 29, -16
.cfi_offset 30, -8
add x29, sp, 0
.cfi_def_cfa_register 29
b .L2
```

```
RELOCATION RECORDS FOR [.eh_frame]:
OFFSET           TYPE                VALUE
000000000000001c R_AARCH64_PREL32    .text

Contents of section .text:
 0000 fd7bbfa9 fd030091 39000014
00000090  .{......9.......
```

Goals: Summary

Help you to gain …



Jungwoo Hong

Programming Maturity

Specific Goal: Learn C

Question: Why C instead of Java?

Answer 1: A primary language for "under the hood" programming in real code bases.

Answer 2: A variety of experience helps you "program in the large"

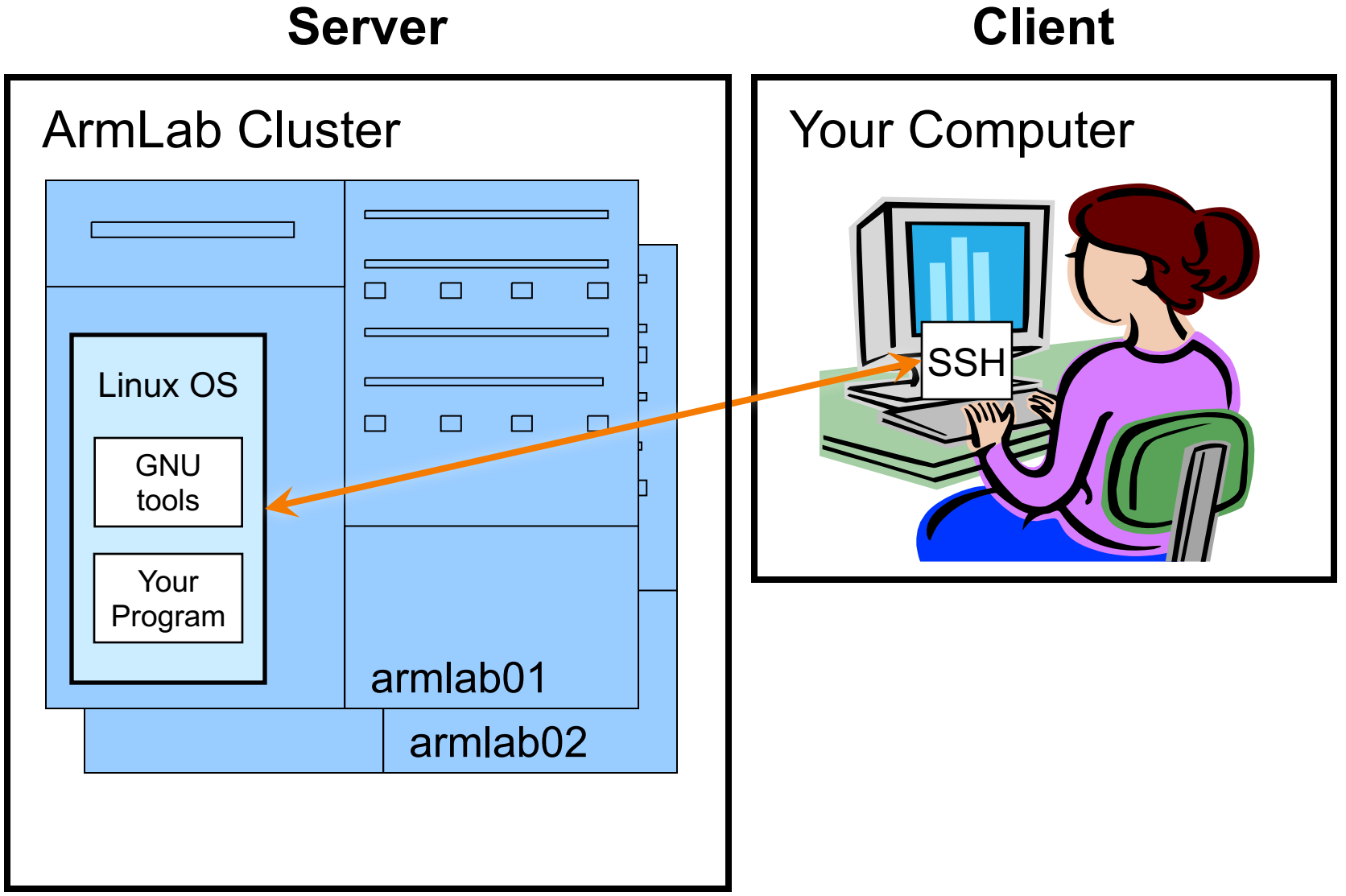
Specific Goal: Learn Linux

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)

Programming Environment



Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

Lectures

Describe material at a mix of levels

- Some conceptual (high) overview
- Some digging into details

Slides on course website

Recordings of live lectures will be posted on course website

Videos from last year (Spring, 2021) available on Youtube

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

Occasional questions in class, graded on participation
(with a generous allowance for not being able to attend)

- Can use either a physical remote, an app on your phone, or web
- Create account / register at iclicker.com
  - If asked, we're using "iClicker Cloud" and "Canvas"

iClicker Question

Q: Do you have an iClicker (remote or app) 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

Describe material at the “practical” (low) level

- Support your work on assignments
- Hard-copy handouts distributed in precept
- Handouts available via course website

Etiquette

- Attend your precept: attendance will be taken
- Must miss your precept? ⇒ inform preceptors & attend another
- Use TigerHub to move to another precept

Precepts begin today!

Websites

https://www.cs.princeton.edu/~cos217 (Course website)

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

https://princeton.instructure.com/courses/5922 (Canvas)

- Links to Ed, Library reserves and other readings, NameCoach

Ed

https://us.edstem.org/us/courses/19718/discussion/

- Also available as a Canvas link
- Q&A – post here instead of emailing staff

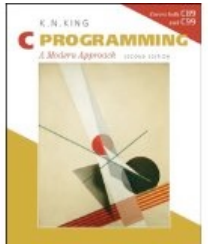
Etiquette

- Study provided material before posting question
  - Lecture slides, precept handouts, required readings
- Read / search all (recent) Ed threads before posting question
- Don’t reveal your code!
  - See course policies
  - Click “private” if in doubt

Books

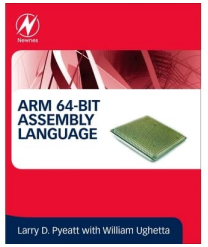
C Programming: A Modern Approach (Second Edition) (required)

- King
- C programming language and standard libraries



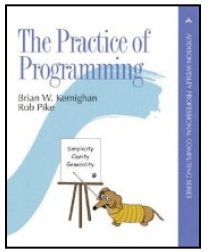
ARM 64-bit Assembly Language (online)

- Pyeatt with Ughetta



The Practice of Programming (online)

- Kernighan & Pike
- "Programming in the large"



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

Help!

Office Hours

- Preceptors: 2+ hours scheduled every weekday + Sunday, in-person and Zoom
- Me: after lecture
- Schedule is on the course website
- Zoom form / links are on Canvas

Lab TAs

- Your peers are available 4+ hours per day, every single day
- These are specific to debugging your assignments.
  For conceptual help with course materials, go to office hours.
- https://labta.cs.princeton.edu/

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

Grading

Course Component	Percentage of Grade
Assignments *	60
Midterm Exam **	10
Final Exam **	20
Participation ***	10

* 6 assignments × 10% each. Late assignments 20% off per day; 4 late days free.

** During midterms week and final exam period, respectively. Closed book/notes.

*** Did your involvement benefit the course?

- Lecture/precept attendance and precept/Ed participation

Programming Assignments

Regular (every 1.5-2.5 weeks) assignments

0. Introductory survey
1. "De-comment" program
2. String module
3. Symbol table module
4. Debugging directory and file trees *
5. Assembly language programming *
6. Buffer overrun attack *

*(partnered assignment)

Assignments 0 and 1 are available now: start early!!

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

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 0
- Typical University-level penalty is suspension

Sanity

COS 1xx/2xx courses are hard under the best of circumstances

- Information-dense
- Programming is a new skill, or “craft”: not like writing essays or doing problem sets

These are not the best of circumstances

- We are all worried about ourselves, friends, family
- We all feel stressed, anxious, uncertain – but when these veer into panic or depression…

Say something, and get help

- Reach out to CPS, your residential college dean, course staff
- No judgment – the rest of us are feeling it too

Questions?

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

The C Programming Language

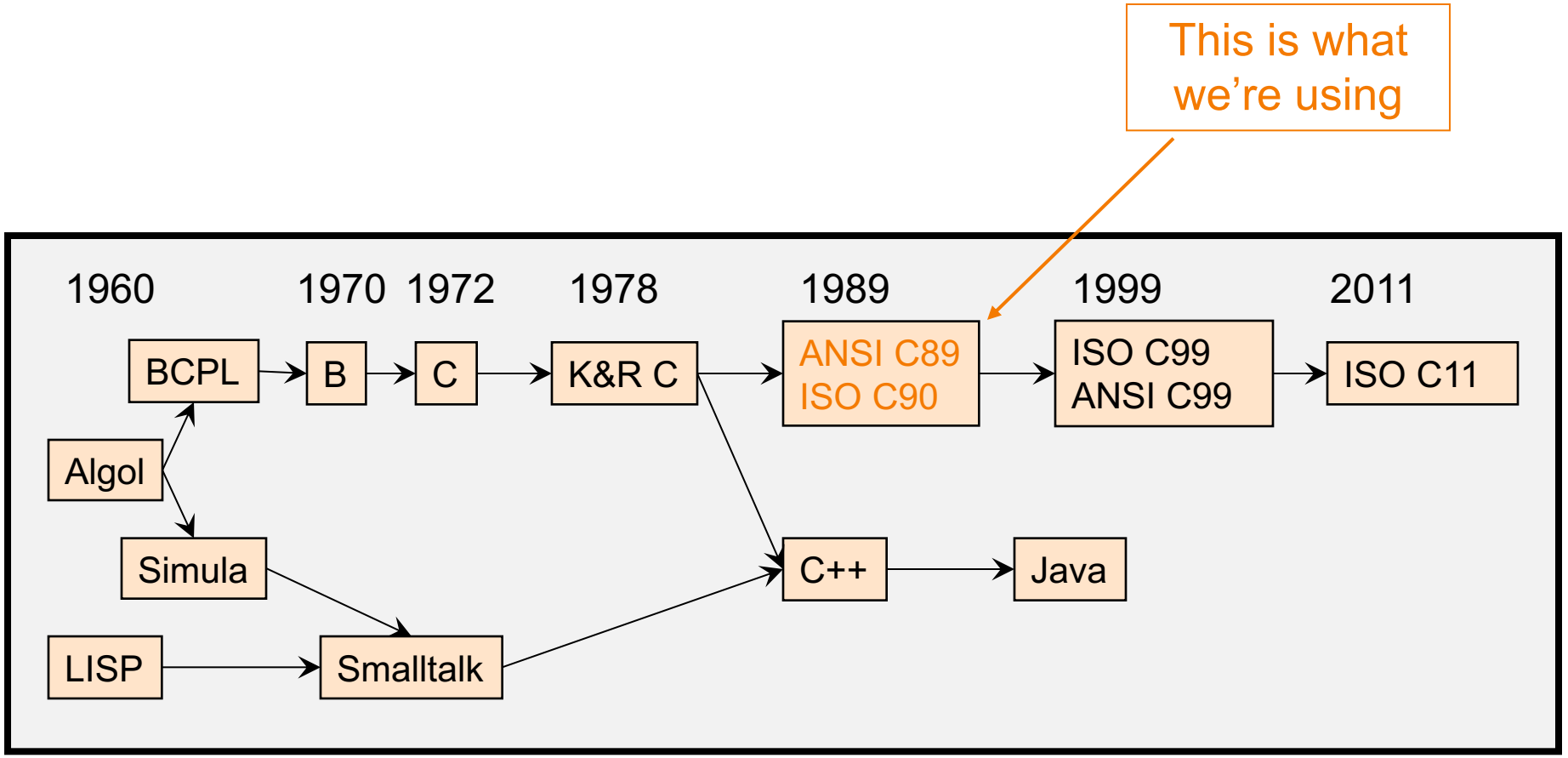
Who? Dennis Ritchie

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

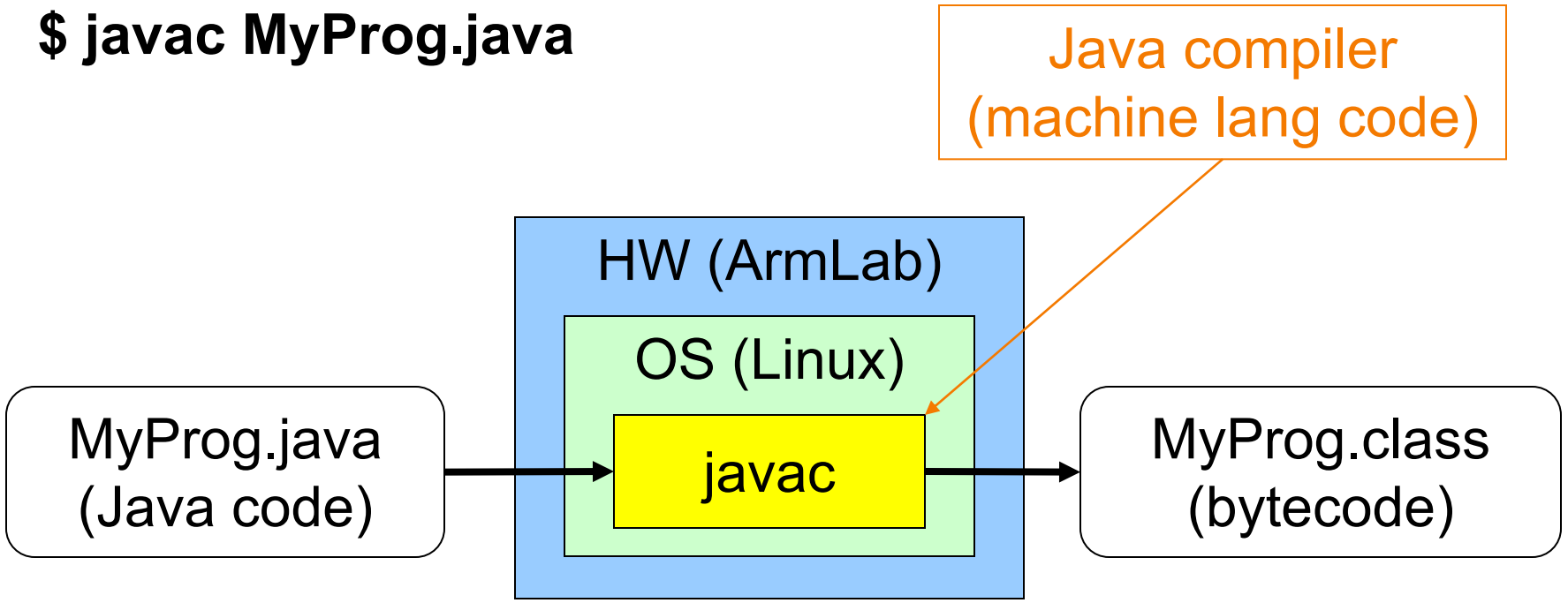
Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

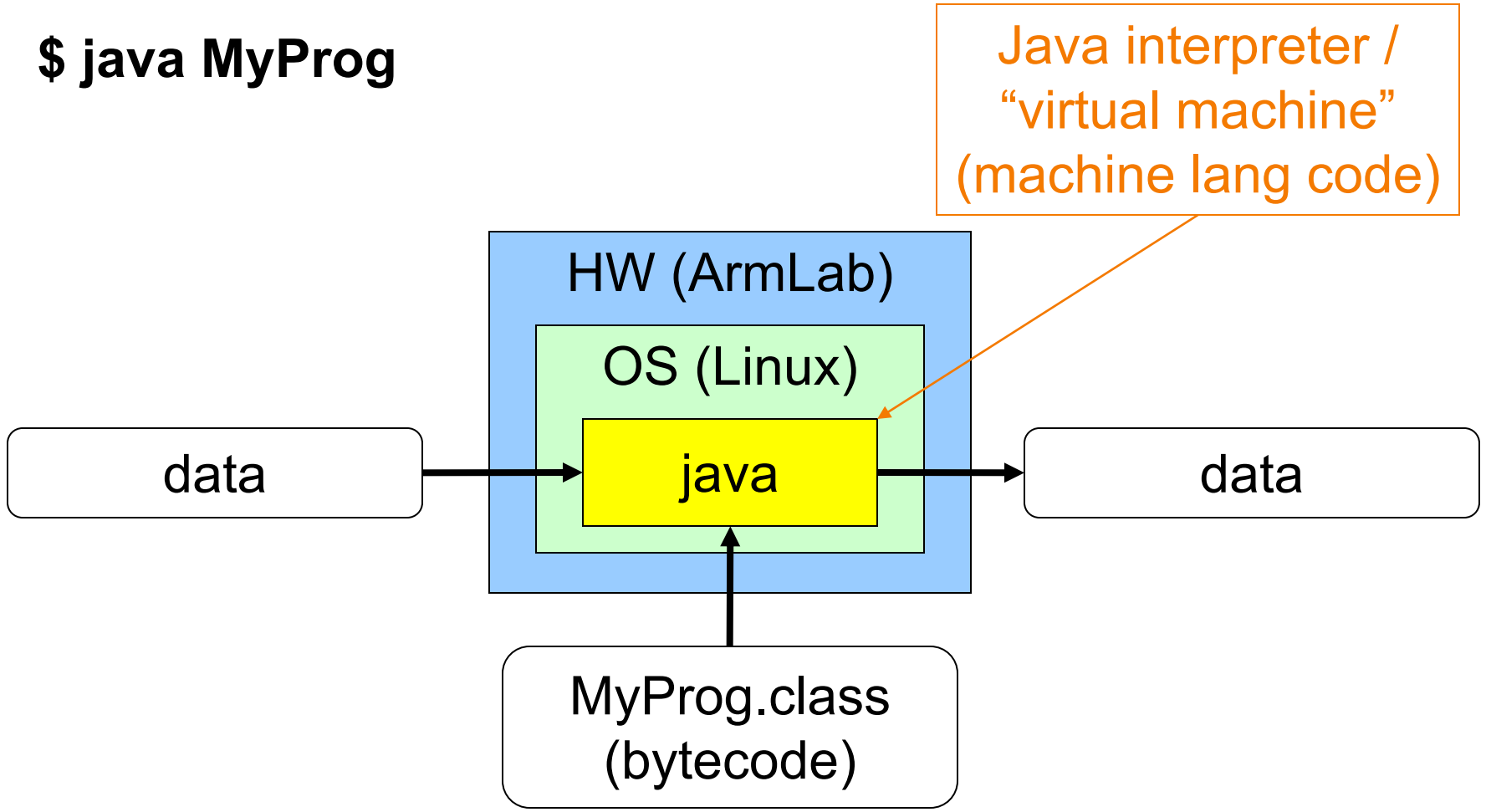
A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

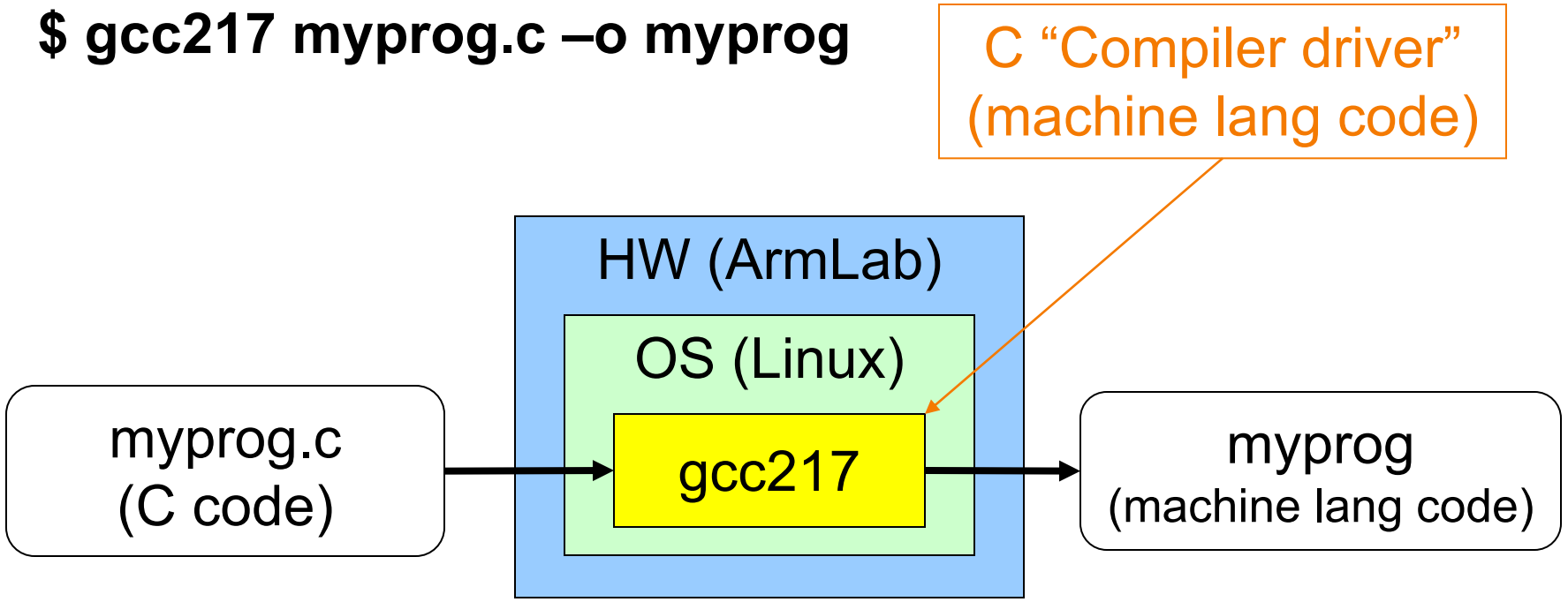
Building Java Programs



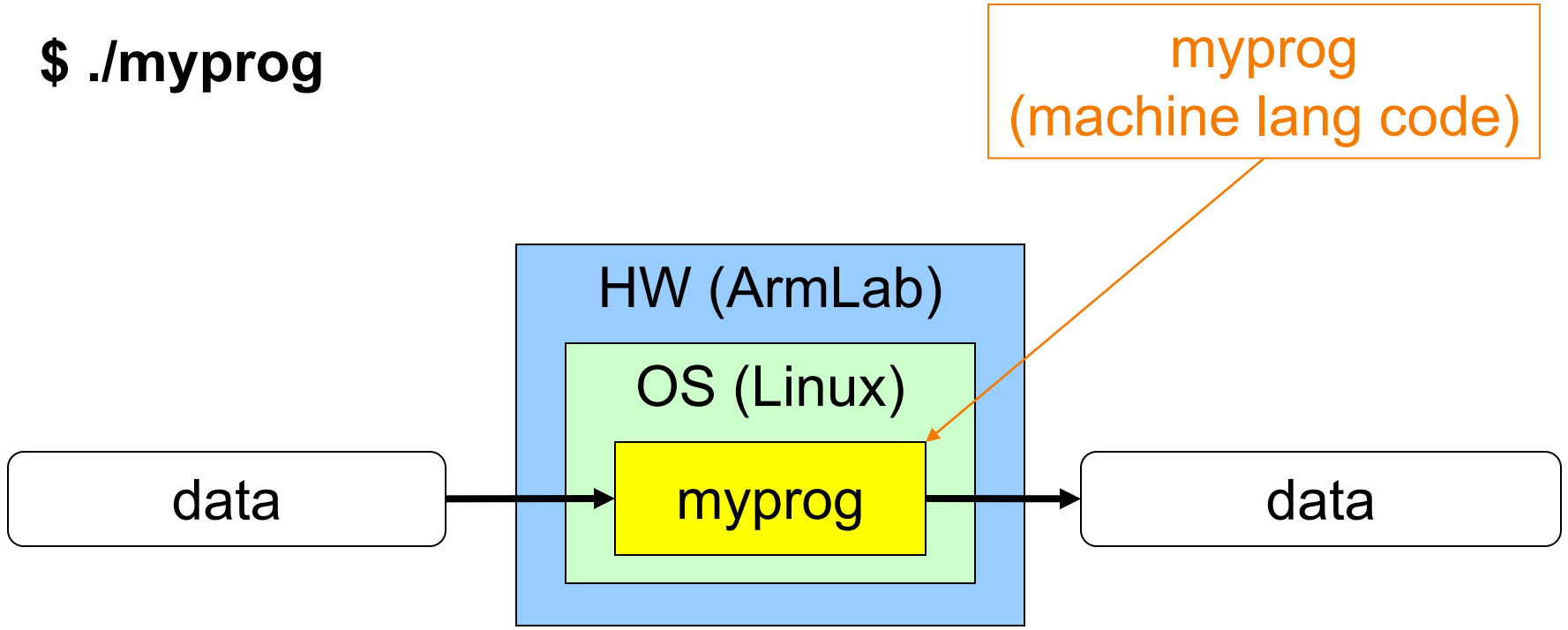
Running Java Programs



Building C Programs



Running C Programs



Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

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

(For example, COS 217 has used many architectures over the years, and every time we switched, all our programs had to 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

iClicker Question

Q: Which corresponds to the C programming language?

A.

B.

C.

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;
}
```

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

A taste of C

- History of C
- Building and running C programs
- Characteristics of C
- Java vs C

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

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, Ed, textbooks
  - Course website: access via https://www.cs.princeton.edu/~cos217
- Grading
- Policies

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

On Wednesday: computing environment

- In preparation for assignments 0 and 1