

```
$ 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", 2020);
    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 2020

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

- Christopher Moretti

cmoretti@cs.princeton.edu

Lead Preceptor

- Xiaoyan Li

xiaoyan@cs.princeton.edu

Preceptors

- Donna Gabai
- Alberto Mizrahi
- Josh Zhang

dgabai@cs.princeton.edu

albertob@princeton.edu

jiashuoz@princeton.edu

Agenda



Course overview

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

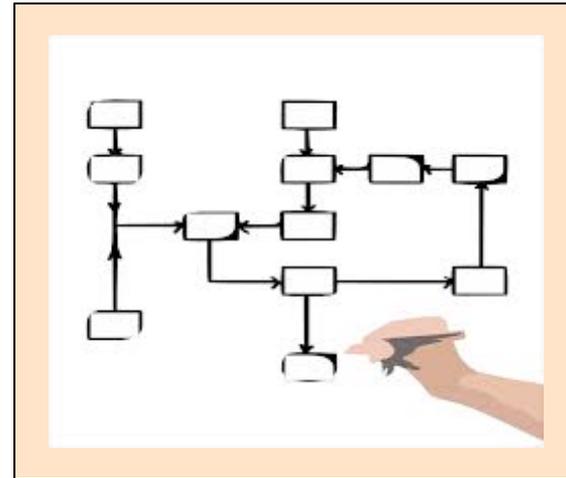
Getting started with C

- 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

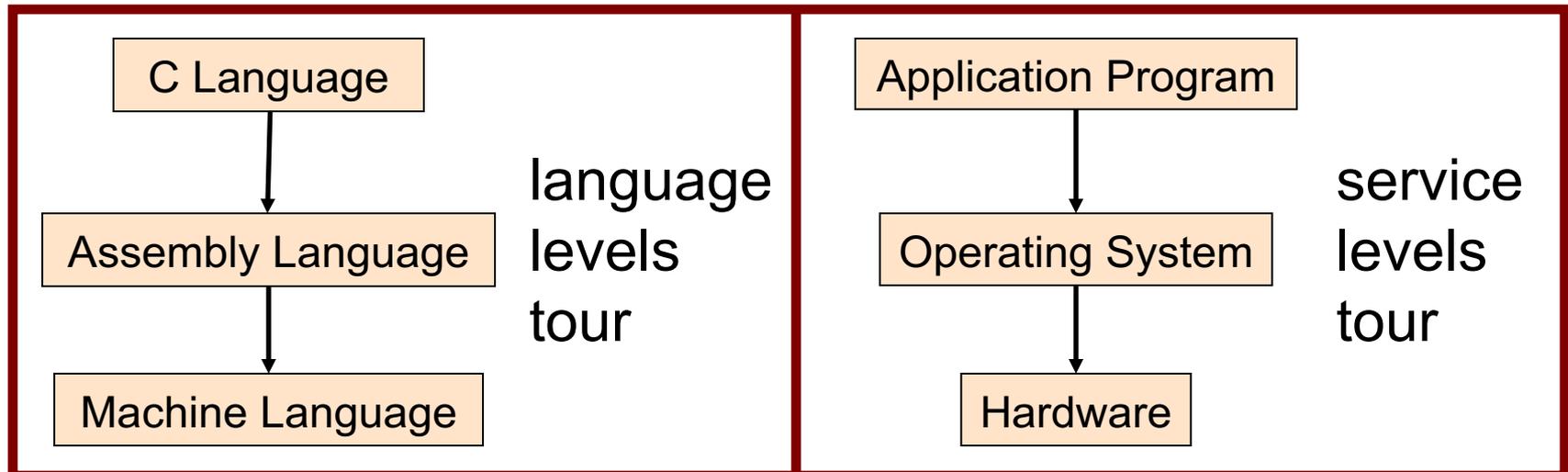
Learn what happens
“under the hood” of
computer systems



Learn “how to be
a client of an
operating system”



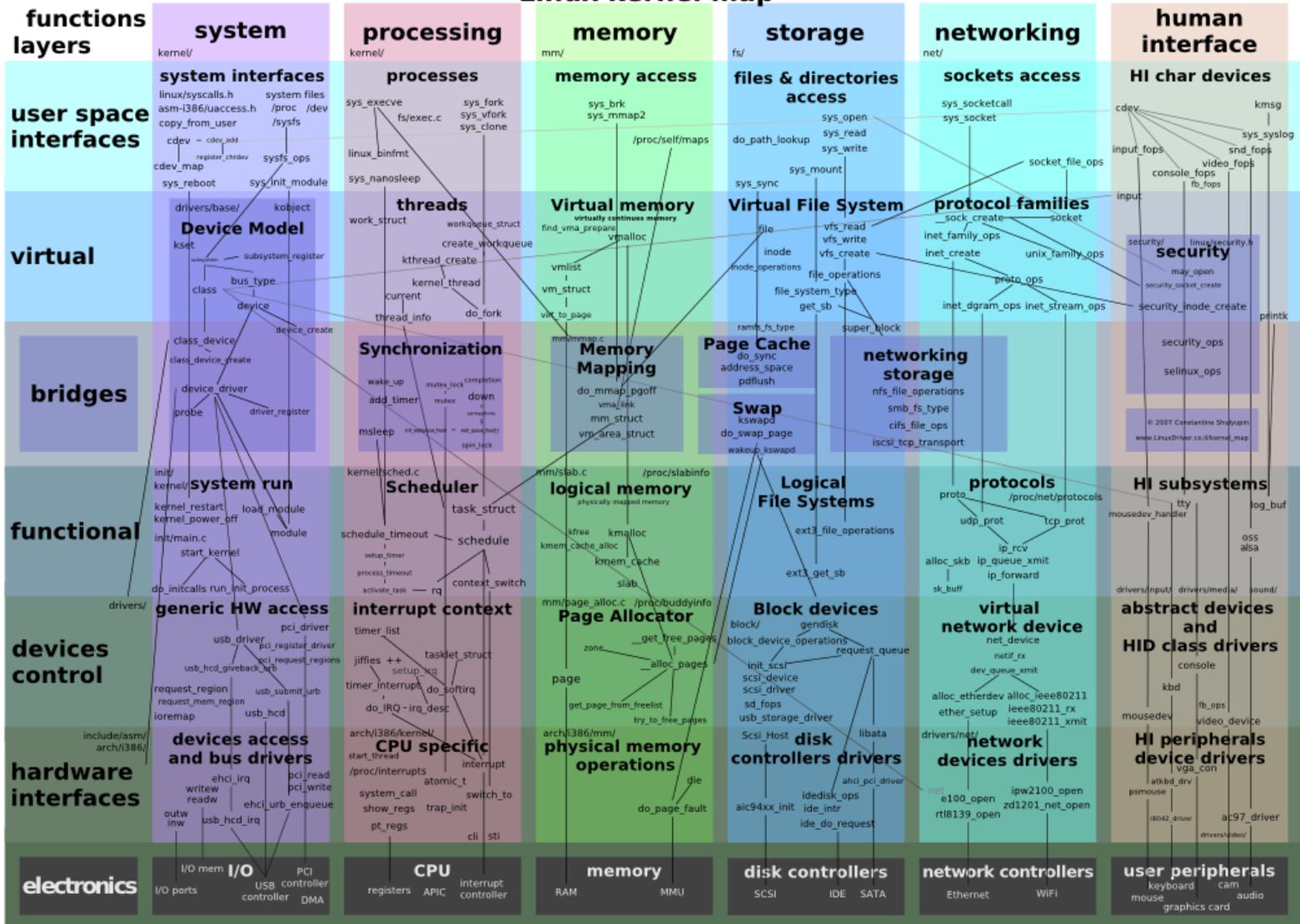
Downward tours



Modularity!



Linux kernel map



Goals: Summary



Help you to gain ...



Programming Maturity

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)

Linux™



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 TigerHub 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!

Website



<https://www.cs.princeton.edu/courses/archive/spr20/cos217/>

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



Piazza



Piazza

- <http://piazza.com/princeton/spring2020/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

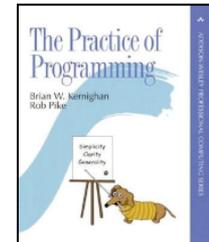


ARM 64-bit Assembly Language (required)

- Pyeatt & Ughetta

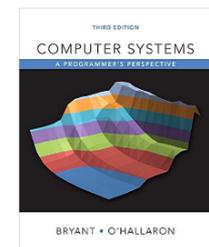
The Practice of Programming (recommended)

- Kernighan & Pike
- “Programming in the large”



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

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



Manuals

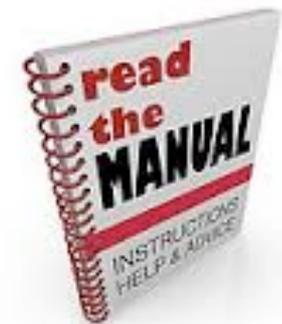


Manuals (for reference only, available online)

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

See also

- Linux *man* command

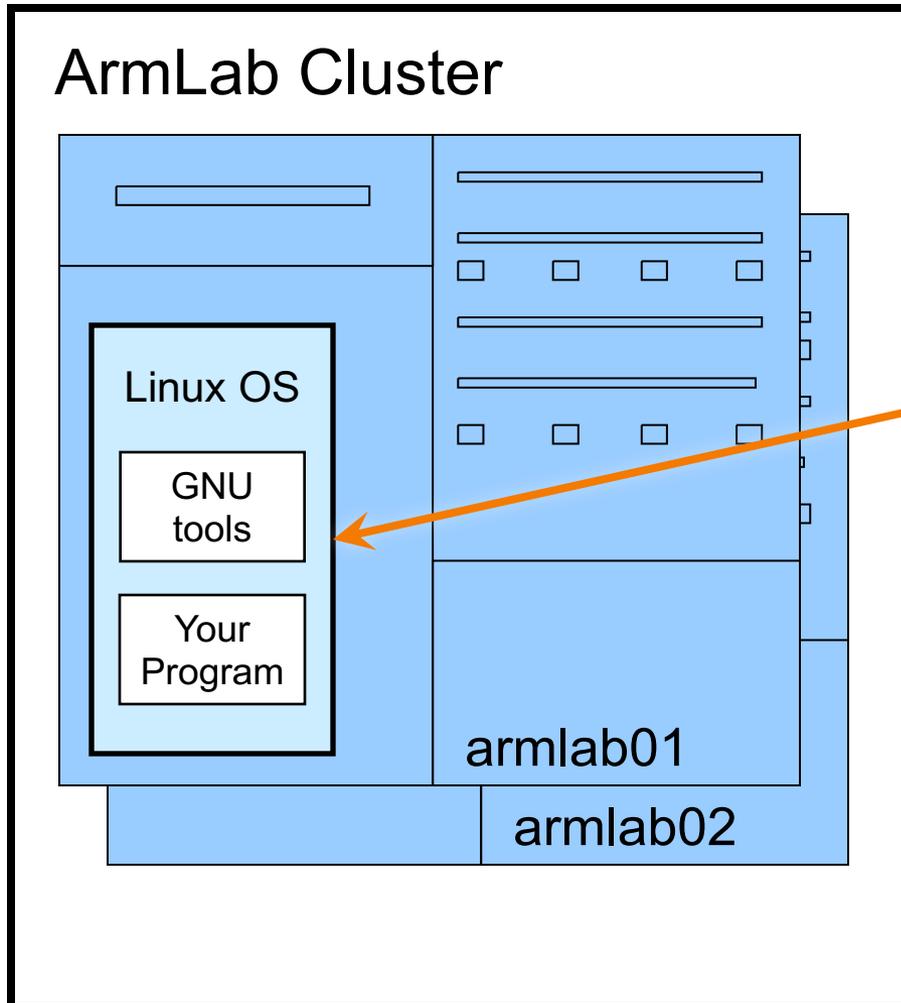


Programming Environment



Server

Client



On-campus or
off-campus

Agenda



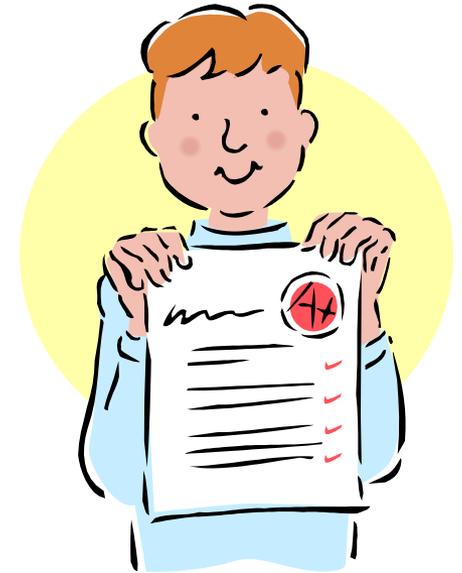
Course overview

- 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
Participation ***	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/precept attendance and participation counts



Programming Assignments

Regular (~every 10-14 days) 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

*(partnered assignment)

Assignments 0 and 1 are available now

Start early! Assignment 1 is due in 10 days!

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 **0 on the assignment**
- Typical University-level penalty is **suspension from University**

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
3-5	Programming in the Large	Advanced C
6	Midterm Exam	
7	Fall 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)

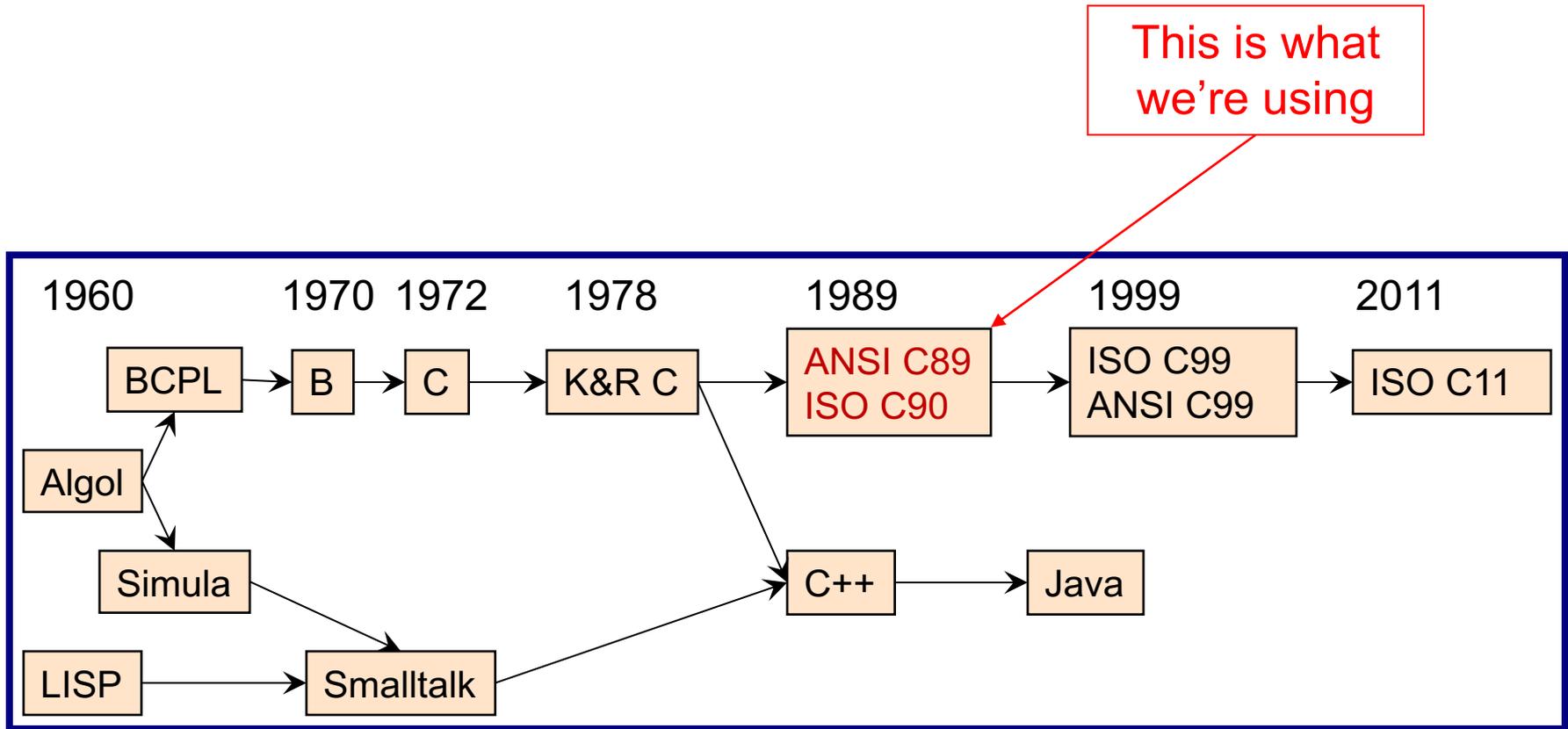
The C Programming Language



- 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

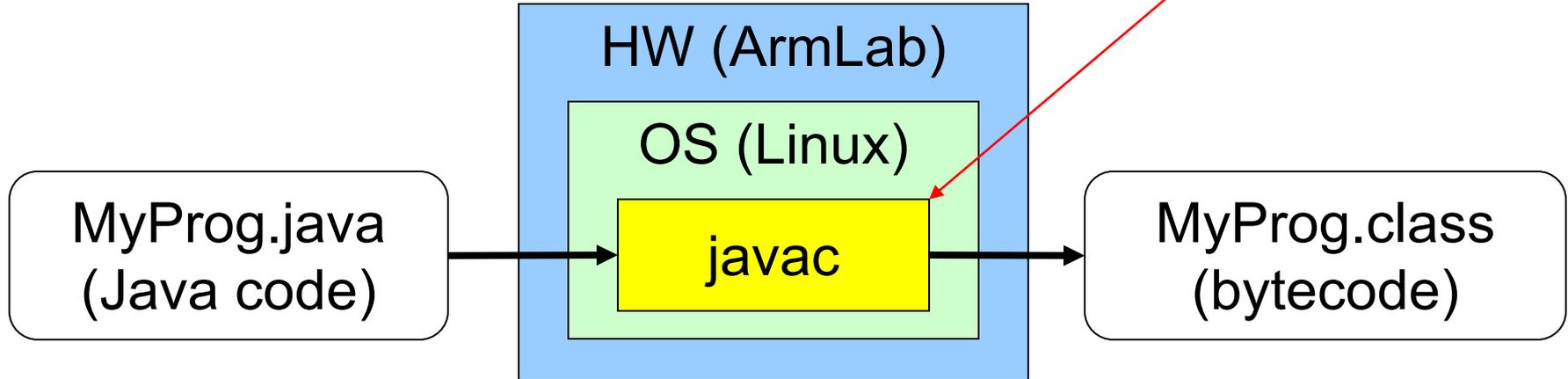
- History of C
- **Building and running C programs**
- Characteristics of C
- C details (if time)



Building Java Programs

```
$ javac MyProg.java
```

Java compiler
(machine lang code)

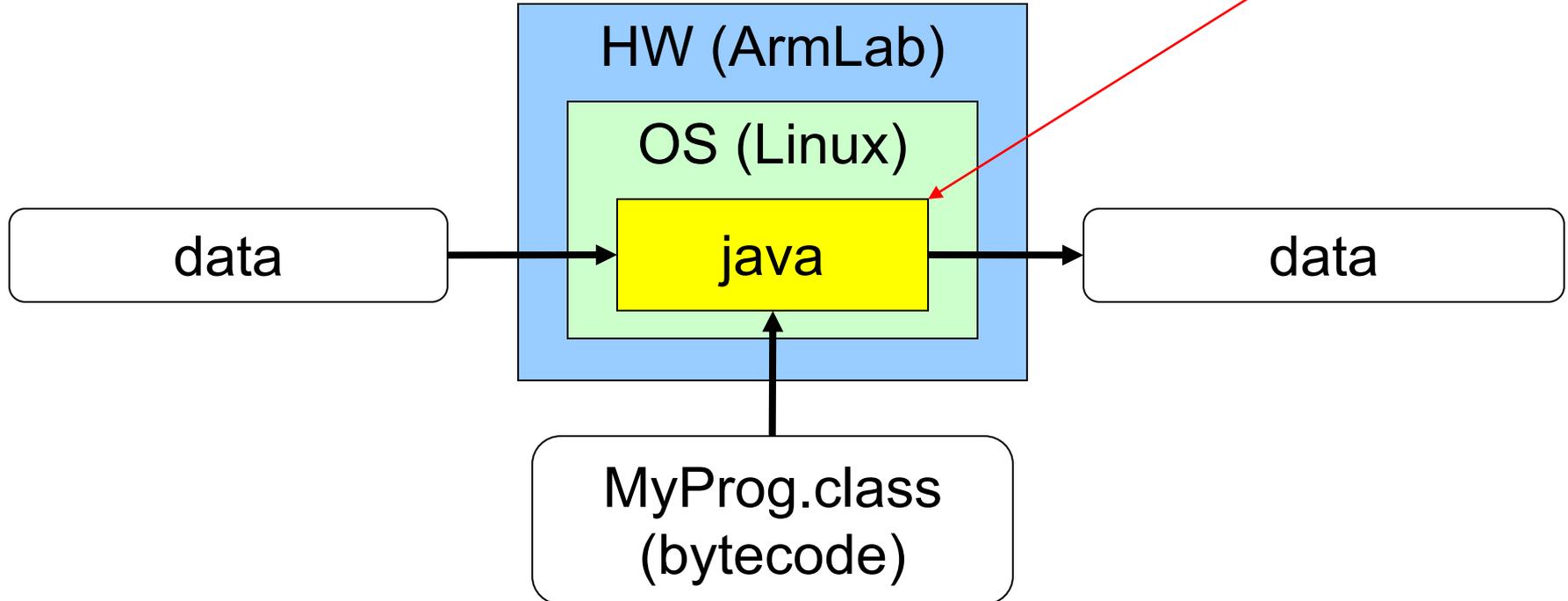




Running Java Programs

`$ java MyProg`

Java interpreter /
“virtual machine”
(machine lang code)

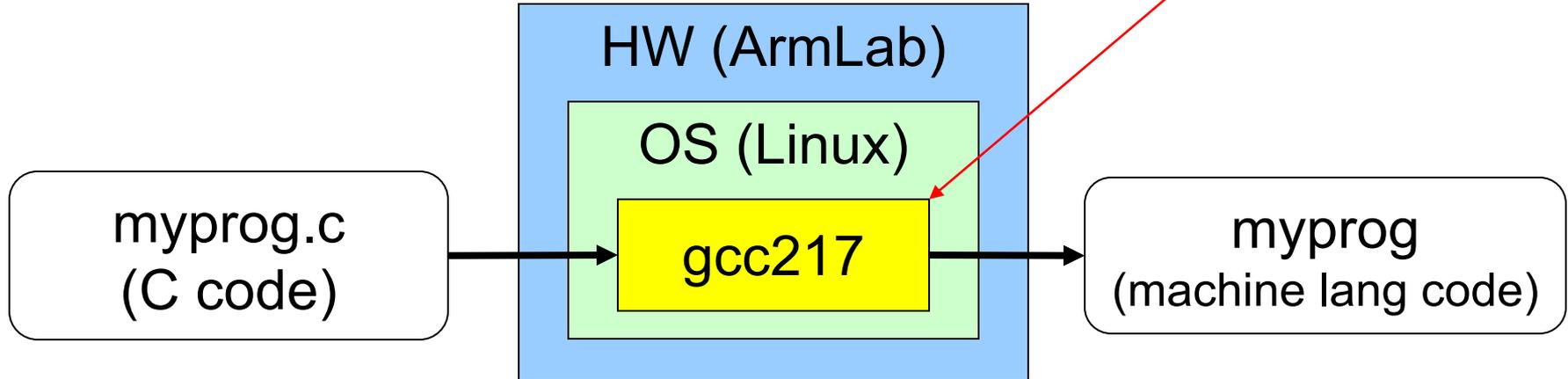




Building C Programs

```
$ gcc217 myprog.c -o myprog
```

C “Compiler driver”
(machine lang code)

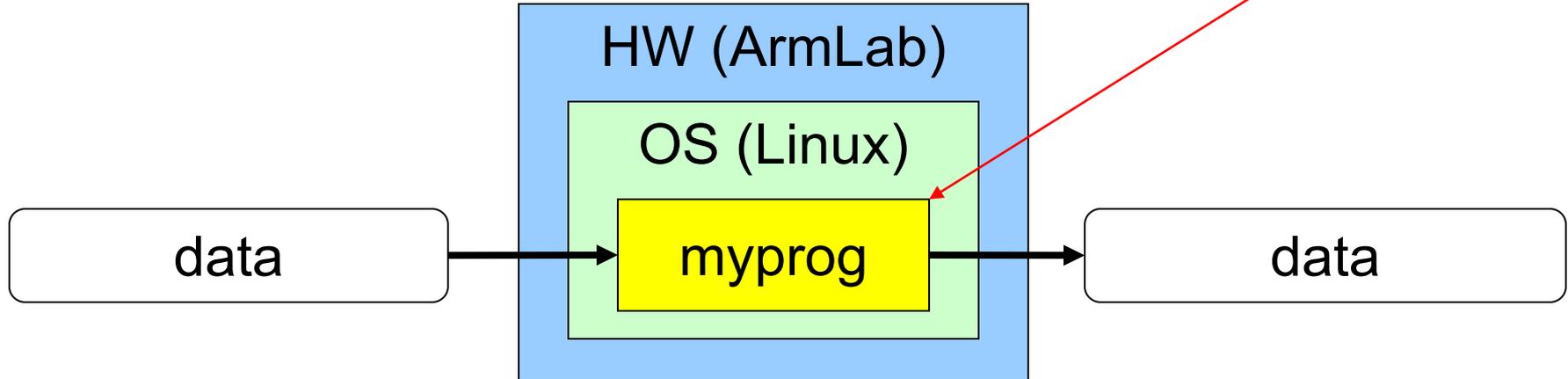


Running C Programs



`$./myprog`

myprog
(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?
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, last spring we moved from the x86_64-based “courselab” to the ARM64-based “armlab”, and 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

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	<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, signed) char</code> <code>(unsigned, signed) short</code> <code>(unsigned, signed) int</code> <code>(unsigned, signed) 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 0 and non-0 */</code>
Generic pointer type	<code>Object</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	<code>String s1 = "Hello"; String s2 = new String("hello");</code>	<code>char *s1 = "Hello"; char s2[6]; strcpy(s2, "hello");</code>
String concatenation	<code>s1 + s2 s1 += s2</code>	<code>#include <string.h> strcat(s1, s2);</code>
Logical ops *	<code>&&, , !</code>	<code>&&, , !</code>
Relational ops *	<code>=, !=, <, >, <=, >=</code>	<code>=, !=, <, >, <=, >=</code>
Arithmetic ops *	<code>+, -, *, /, %, unary -</code>	<code>+, -, *, /, %, unary -</code>
Bitwise ops	<code><<, >>, >>>, &, ^, , ~</code>	<code><<, >>, &, ^, , ~</code>
Assignment ops	<code>=, +=, -=, *=, /=, %= <<=, >>=, >>>=, &=, ^=, =</code>	<code>=, +=, -=, *=, /=, %= <<=, >>=, &=, ^=, =</code>

* 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	<pre>goto someLabel;</pre>

* Essentially the same in the two languages

Java vs. C: Details



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