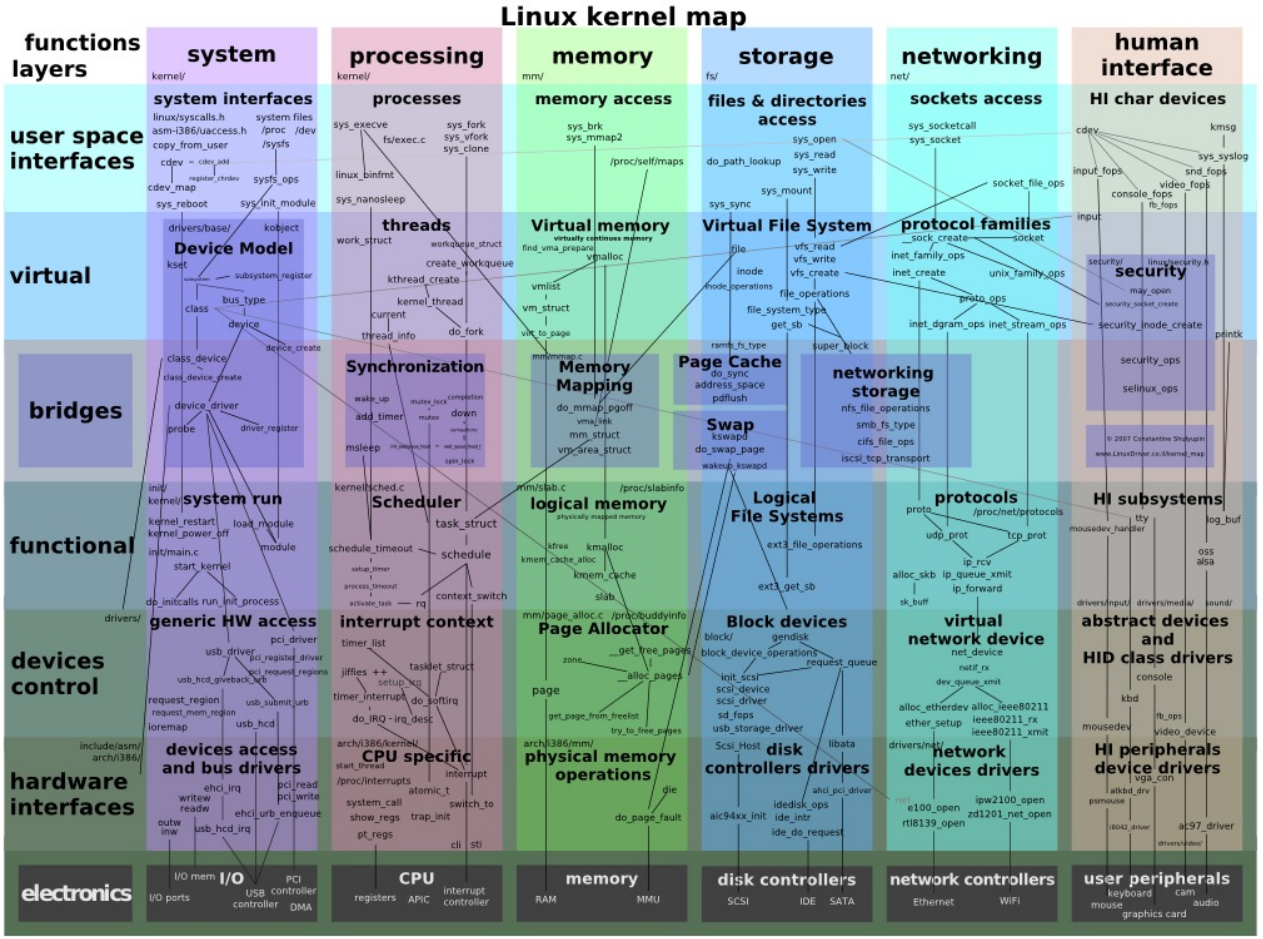
Welcome to COS 217

Introduction to Programming Systems

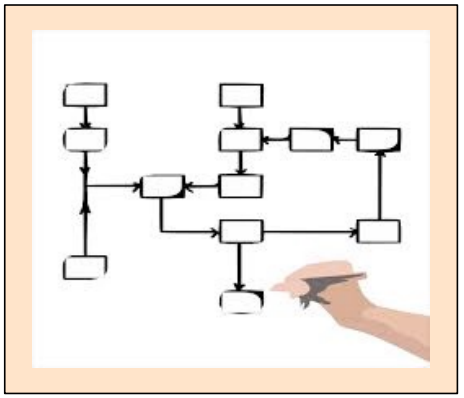
Spring 2025

Why Are We Here?

1. To Learn to Manage This



Programming in the Large



Large(r) computer programs are made up of many modules

Topics

- Modularity, abstraction, well-defined interfaces, separation of interface from implementation, information hiding), resource management, error handling
- Testing, debugging, performance improvement
- Tools: version control (git), managing compilation (make), profiling, ...

2. To Learn What Actually Happens

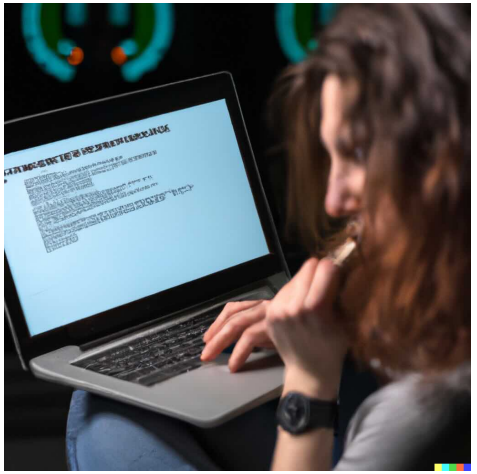
```
int main(void) {
   while ((iChar = getchar()) != EOF) {
      lCharCount++;
      if (isspace(iChar)) {
         if (iInWord) {
            lWordCount++;
            iInWord = FALSE;
         }
      }
```

arm

```
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.......
```



“Under the Hood” Interactions

With the operating system and the hardware

- By using lower-level languages (C and Assembly, not Java/Python)
- By building on an open-source but industry-standard OS (Linux)

3. To Prepare for Higher-level COS Courses

COS 333: Advanced Programming Techniques

COS 316: Principles of Computer System Design

COS 318: Operating Systems

COS 418: Distributed Systems

Today's Agenda

Administrative things

- Introductions
- Activities and Resources
- Grading
- Policies

Computing environment

- Key software / terminology
- Navigating the filesystem
- Demo (time permitting)

Introductions

Course Faculty

- Jaswinder Pal Singh — jps@cs.princeton.edu
- Christopher Moretti — cmoretti@cs.princeton.edu
- Kevin Alarcón Negy — kn3496@princeton.edu

Graduate Preceptors

- Amelia Dobis — amelia.dobis@princeton.edu
- Lana Glisic ‘24 — lglisic@princeton.edu
- Andrew Johnson — aj3189@princeton.edu
- Nicholas Yap — zy4586@princeton.edu

Today's Agenda

Administrative things

- Introductions
- Activities and Resources
- Grading
- Policies

Our computing environment

- Key software / terminology
- Navigating the filesystem
- Demo (time permitting)

Lectures

Conceptual overview, plus some digging into details

Slides on course website

Videos from some previous offerings available on YouTube
… but be careful, you are responsible for any differences

Etiquette

- Ask questions as they come up
- Use electronic devices for taking notes or annotating slides
- Limit browsing / social media use in class, please -- for yourself and neighbors

iClicker

Occasional questions in class: graded on participation, not correctness.

- You can use an app on your phone or the web client
- Setup is "iClicker Cloud", integrated with our course's Canvas.
- Register, select Princeton University, and find course "COS 217 – Spring 2025"

iClicker Question

Q: Can you answer this iClicker question today?

A. Yes

B. I would prefer not to

C. I'm not here, but someone is iClicking for me
(don't do this – it's an academic violation!)

Precepts

Describe material at the “practical” (low) level

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

Etiquette

- Attend your precept: attendance will be taken, AND you should learn something
- Use TigerHub to move to another precept if timing is a problem
- Must miss your precept once or twice? ⇒ inform preceptors & attend another precept in place of the one you miss

Precepts begin *today*

Websites

1. https://www.cs.princeton.edu/~cos217

(Course website)

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

2. https://princeton.instructure.com/courses/17063

(Canvas)

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

3. Ed

https://edstem.org/us/courses/74019/discussion

- Also available as a Canvas link from the course website
- Q&A – post here instead of emailing staff when possible

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 or design decisions in a public post
  - See course policies
  - Click "private" if in doubt – we can make it public after-the-fact

4. codePost

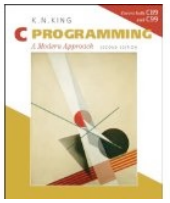
We will use `codePost.io` to annotate your assignment submissions with feedback and grades.

More information on this when we get ready to return Assignment 1.

Books

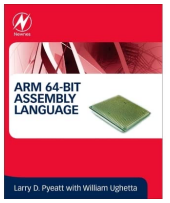
C Programming: A Modern Approach (Second Edition) (required)

- King
- C programming language and standard libraries



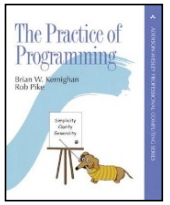
ARM 64-bit Assembly Language (required / online)

- Pyeatt with Ughetta



The Practice of Programming (online)

- Kernighan and Pike
- "Programming in the large"



Help!

Office Hours, Concept Hours, and Study Hall

- Off Hours: 3+ hours per day 6 days per week; schedule is on the course website
- Office hours: offer help on assignments, as well as lecture and precept material
- “Concept” office hours after lecture: focus on course material, not debugging
- McGraw Study Hall: like Concept hours, but with peers

Intro COS Lab Hours

- Intro Lab TAs are your peers who have already completed this course.
- Available 4+ hours per day, every single day in Lewis Library:
  https://introlab.cs.princeton.edu/
- These sessions are specific to **debugging** your assignments.
  Go to (regular or concept) office hours or study hall
  for **conceptual** help with course materials.

Today's Agenda

Administrative things

- Introductions
- Resources
- Grading
- Policies

Our computing environment

- Key software / terminology
- Navigating the filesystem
- Demo (time permitting)

Grading

Course Component	Percentage of Grade
Assignment Submissions *	25
Assignment Quizzes	20
Midterm Exam **	20
Final Exam **	30
Participation ***	5

* Late assignments 10% off per day; first 4 late days waived.

** During midterms week and final exam period, respectively – dates are on website.

*** 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

Administrative things

- Introductions
- Resources
- Grading
- Policies

Our computing environment

- Key software / terminology
- Navigating the filesystem
- Demo (time permitting)

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
- Treat AI chatbots or assistants at least as carefully as you would treat interaction with classmates. Even better: avoid entirely.

@jdent

Violations of course policies

- Typical course-level penalty is 0
- Typical University-level penalty is probation or suspension

Questions?

Today's Agenda

Administrative things

- Introductions
- Resources
- Grading
- Policies

Our computing environment

- Key terminology and software
- Navigating the filesystem

Terminology: Operating System

Narrow definition:

A piece of software that controls the interaction between programs and hardware (CPU, memory, storage, peripherals).

Also sometimes called a “kernel”.

Modern Kernel Examples

- Unix lineage: Linux, XNU
- VMS lineage: Windows NT

Looser definition:

The kernel plus a variety of libraries and tools built upon it, that provide a specific experience to users (e.g., GUI).

Modern OS Examples

- Linux kernel: Linux/GNU, Android
- XNU kernel: macOS, iOS
- Windows NT kernel: Windows

Terminology: User Interface

Graphical User Interface (GUI):

Graphical “point and click” or “swipe and tap” paradigm for interacting with programs.

Programs usually designed to respond to “events”, and display output via “widgets”.

Often more user-friendly.

Mac Finder, Windows Start Menu

Command Line Interface (CLI):

Text-based paradigm for interacting with programs.d

Programs usually designed to accept typed (text-based) input and produce text-based output.

Easier to code, more flexible, easier to execute remotely, and easier to automate/script!

Terminal emulator

Terminology: Terminal and Shell

Terminal

Terminal Emulator:

GUI program that relays typed input to a CLI program and displays its output on the screen.

Shell:

CLI or GUI program for managing files and running other programs.

CLI example: `bash`

Terminology: `ssh`

`ssh`:

Stands for "secure shell"
(but it's not a shell!)

CLI program that connects to `sshd` on another computer and relays text back/forth securely.

`sshd`:

Program that runs continuously on a server, accepts network connections from ssh clients, and relays text back/forth to a local shell (e.g., `bash`).

Terminology: Text Editor

Text Editor:

Allows editing *plain text*:
just a sequence of characters.
Examples: TextEdit, Notepad, Sublime Text, emacs, vi

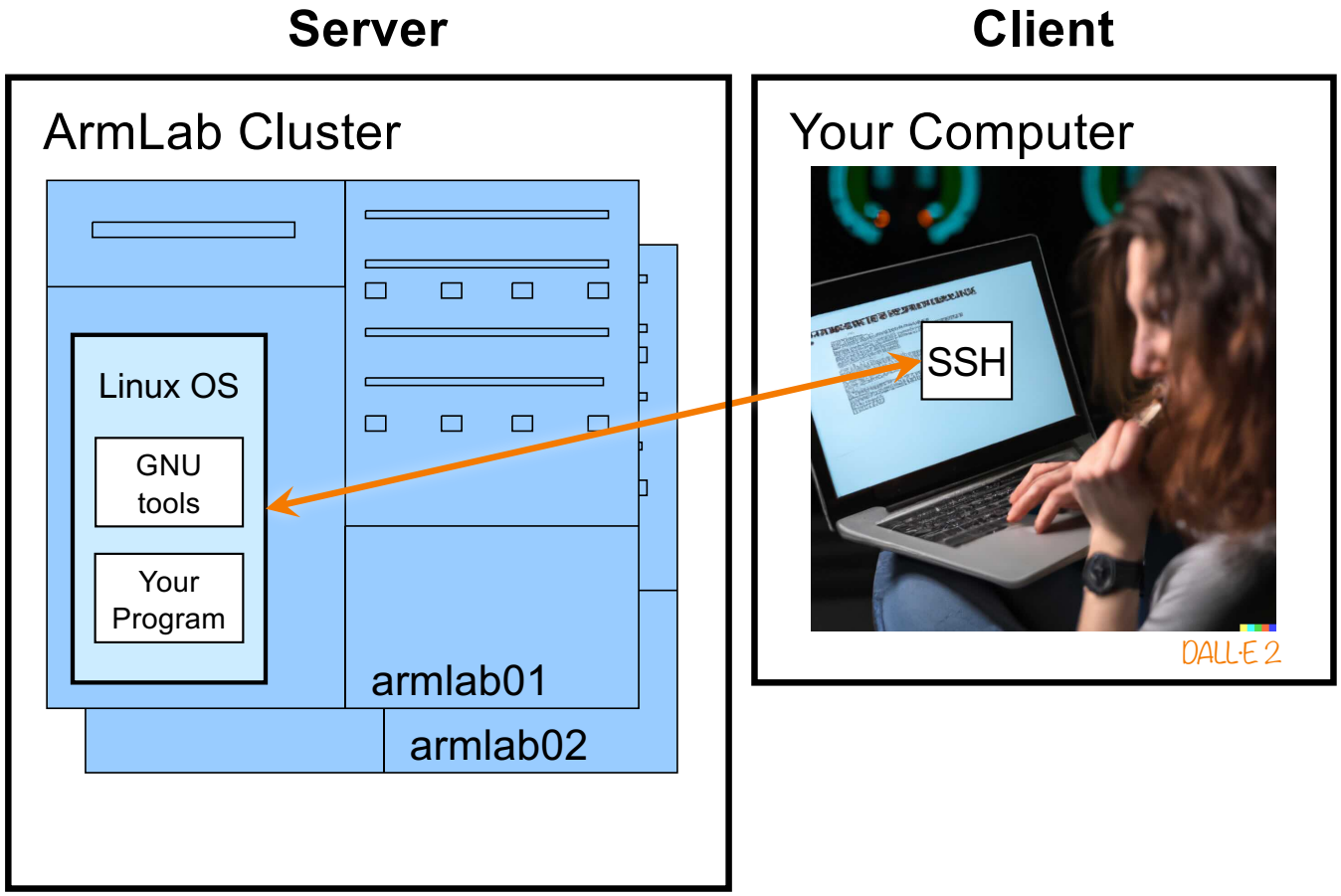
Word Processor:

Allows editing text with formatting (various fonts, paragraphs, etc.)
Does *not* output plain-text.
Examples: Word, Pages

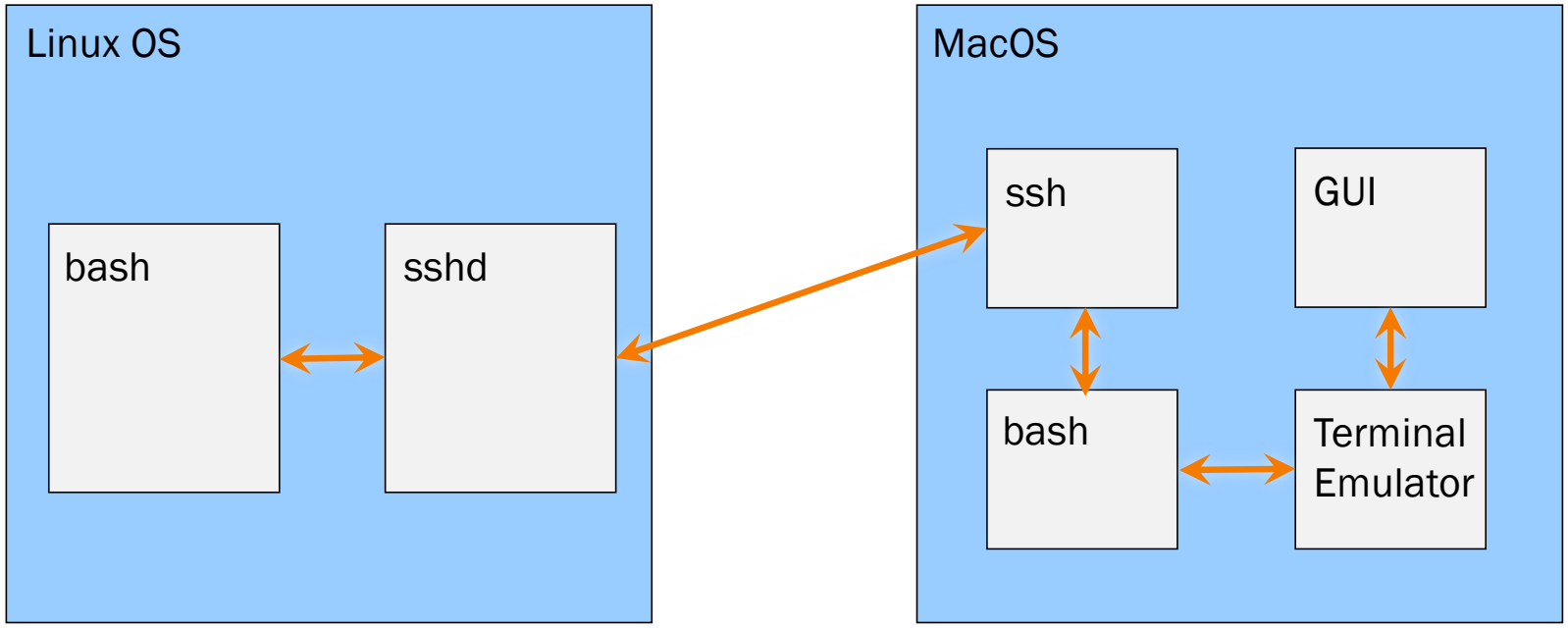
Integrated Development Environment (IDE):

Text editor optimized for code – usually integrates syntax coloring, compiling, searching for errors, sometimes suggesting variable names or code snippets.
Examples: IntelliJ, VS Code, {emacs, vi} *with the appropriate configuration*

So What's Our Programming Environment?



Programming Environment – Under the Hood



armlab

Your Mac computer
(Windows would be similar)

Today's Agenda

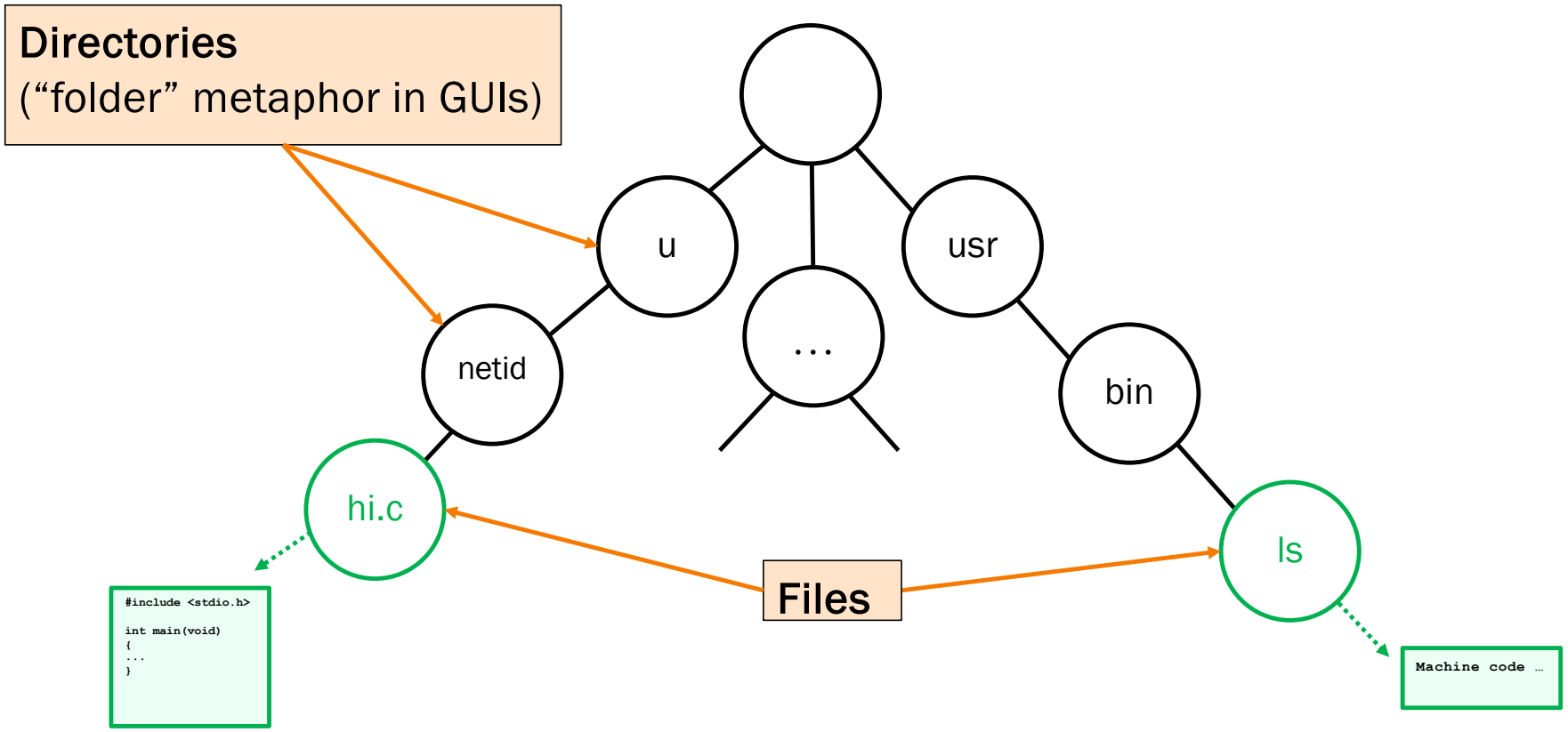
Administrative things

- Introductions
- Resources
- Grading
- Policies

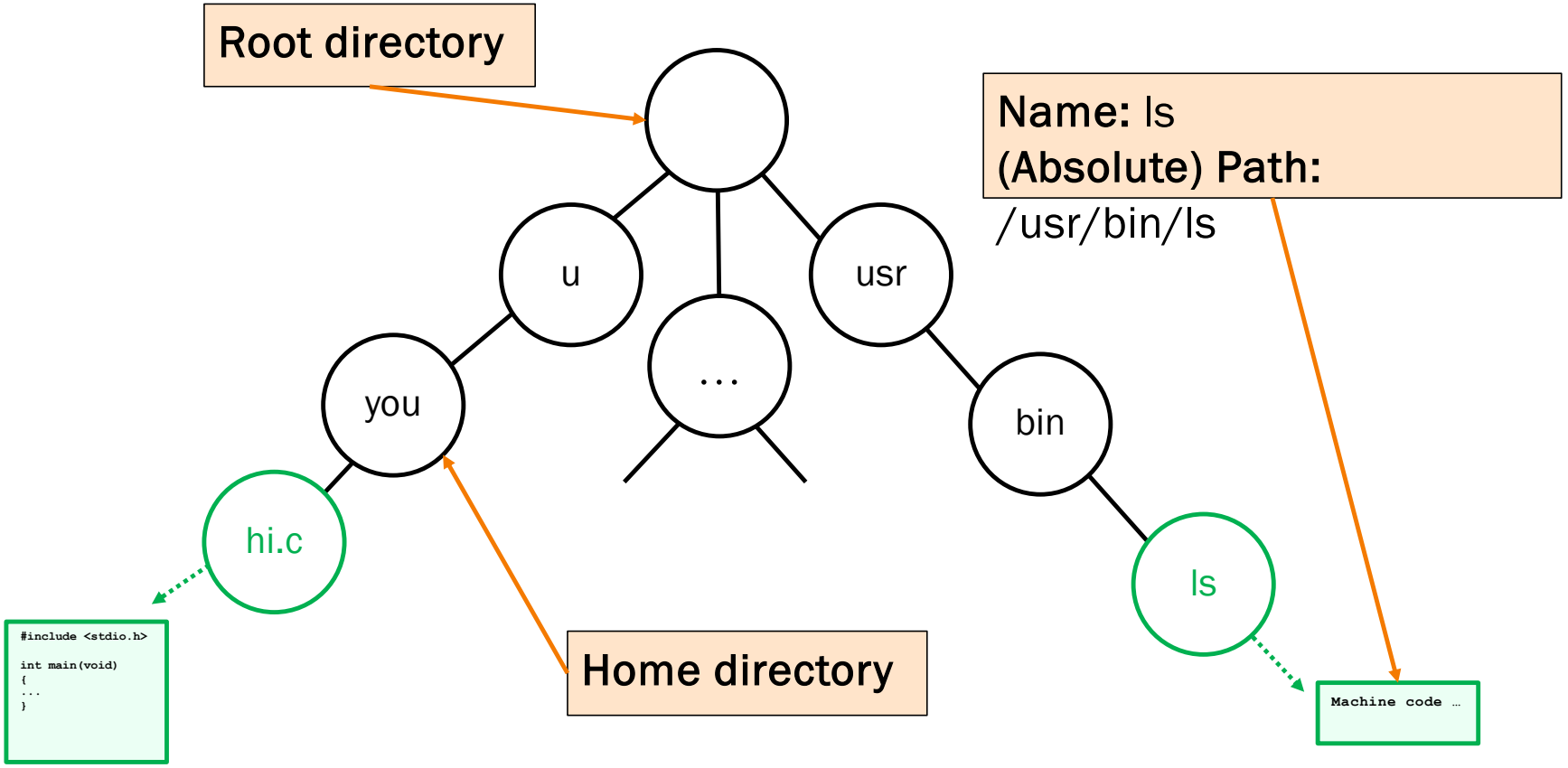
Our computing environment

- Key terminology and software
- Navigating the filesystem

Filesystems

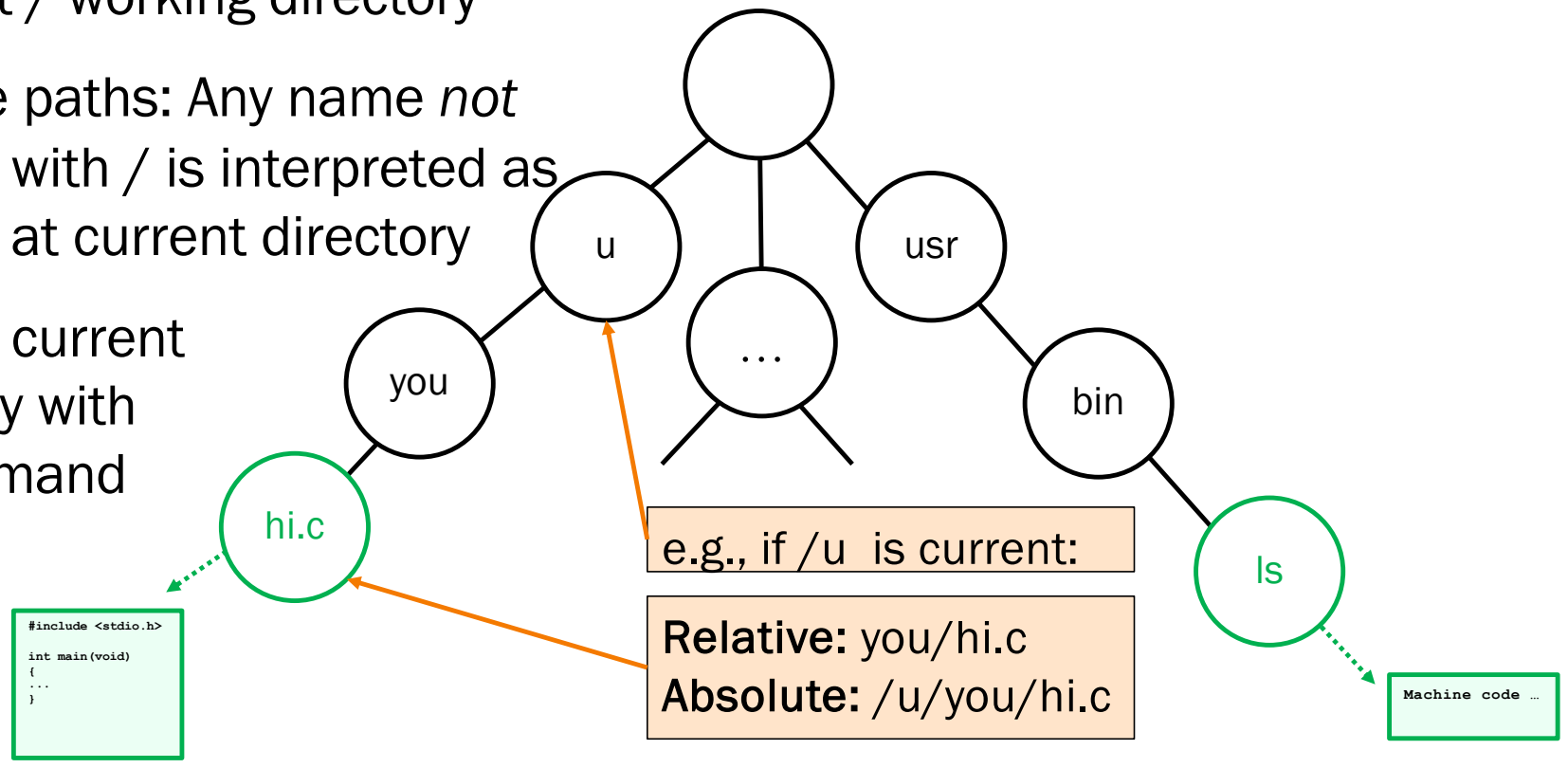


Filesystems



Filesystem Shortcuts

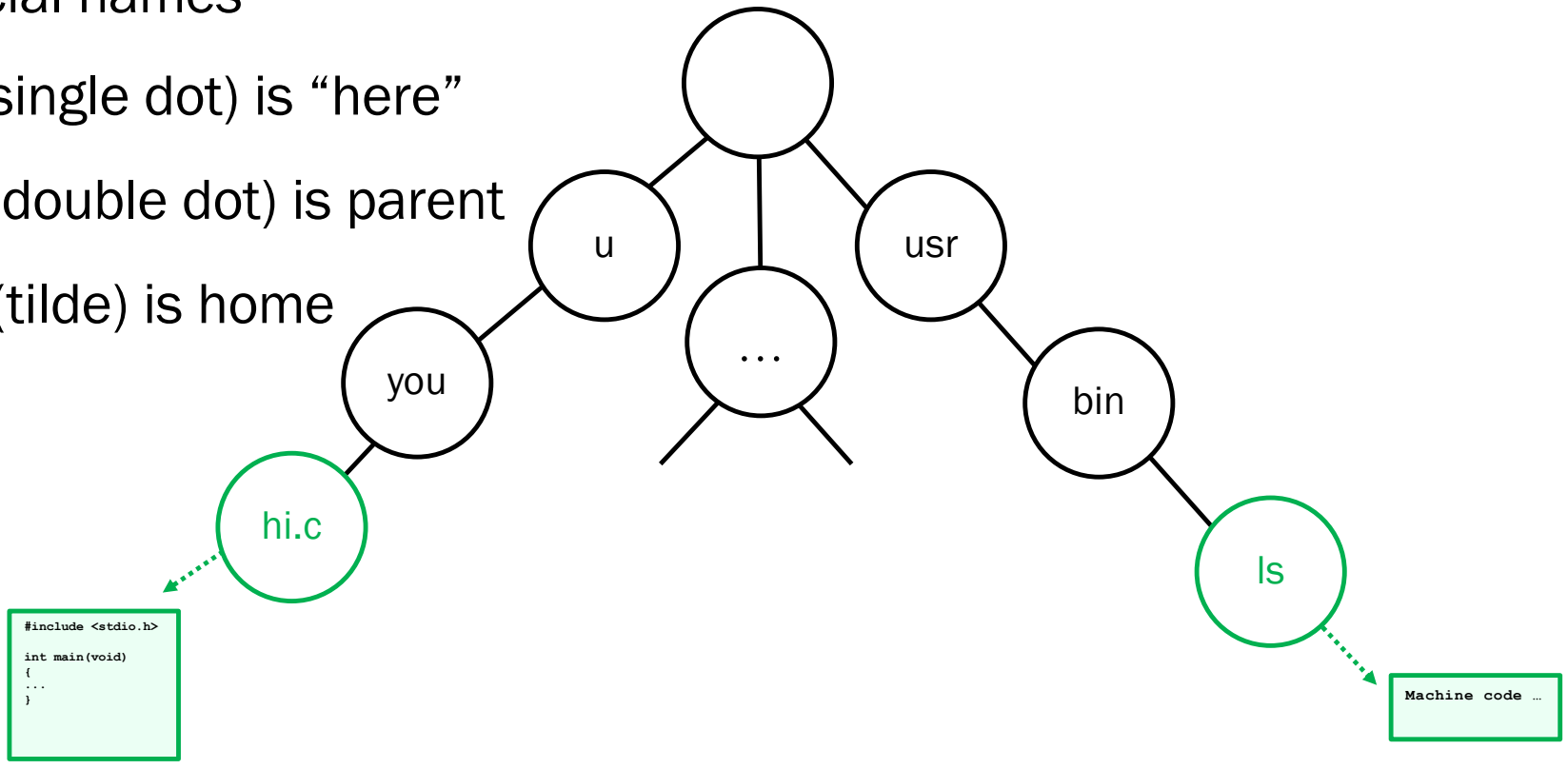
- Current / working directory
- Relative paths: Any name *not* starting with / is interpreted as starting at current directory
- Change current directory with *cd* command



Filesystem Shortcuts

Special names

- . (single dot) is “here”
- .. (double dot) is parent
- ~ (tilde) is home



Filesystem: What Did We Learn?

- Directory hierarchy
- Absolute pathnames
- Some shortcuts: relative pathnames and special names

Next steps ...

- Check out website and policies **soon**
  https://www.cs.princeton.edu/~cos217
- Please attend all precepts diligently
- For more on Linux/Shell: optional (but strongly encouraged) lecture videos from Fall 2020:
  - "Getting Started with bash" walkthrough
  - Advanced bash walkthrough