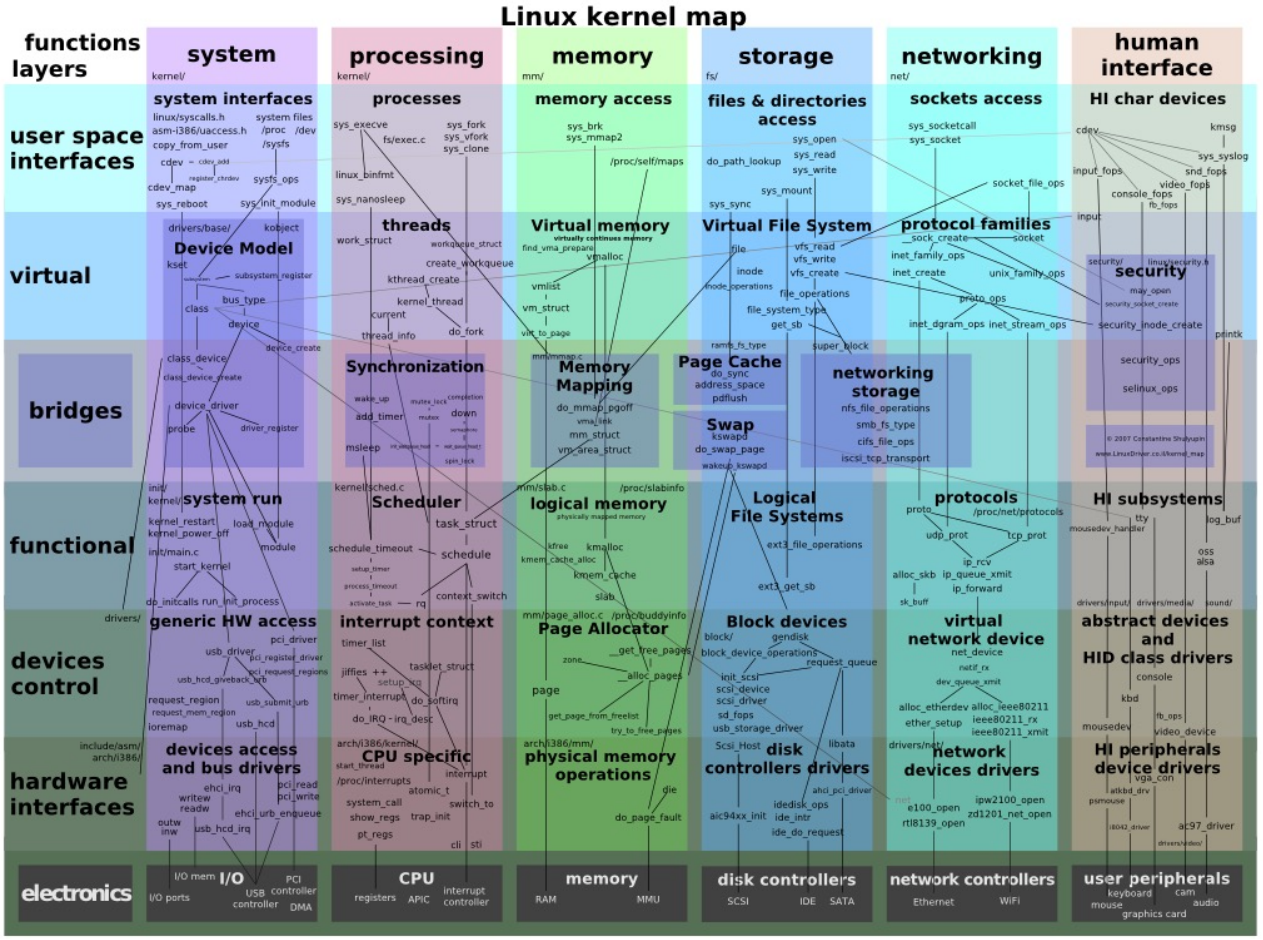
Welcome to COS 217

Introduction to Programming Systems

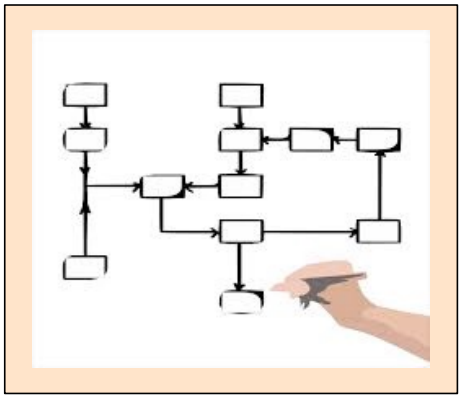
Fall 2026

Why Are We Here?

Goal 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, ...

Goal 2: To Learn What Actually Happens

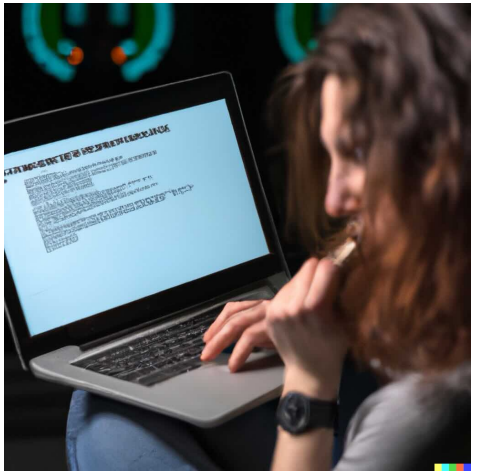
```
int main(void) {
   while ((iChar = getchar()) != EOF) {
      lCharCount++;
      if (isspace(iChar)) {
         if (iInWord) {
            lWordCount++;
            iInWord = FALSE;
         }
      }
```

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)

Goal 3: To Prepare for Higher-level COS Courses

COS 333: Advanced Programming Techniques

COS 316: Principles of Computer System Design

COS 417: 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

Introductions

Course Faculty

- Brian Kernighan: Faculty Preceptor — bwk@cs.princeton.edu
- Christopher Moretti: Co-lead Preceptor — cmoretti@cs.princeton.edu
- Kevin Alarcón Negy: Co-lead Preceptor — kn3496@princeton.edu
- Jaswinder Pal Singh — jps@cs.princeton.edu

Graduate Preceptors

- Khai Evdaev — ke3574@princeton.edu
- Baiang Li — baiang.li@princeton.edu
- Zhe Kai 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
- Please try to not use electronic devices in class, at least after the iClicker answer
- Limit browsing / social media use in class, please -- for yourself and neighbors

iClicker

Question in first 10 min of lecture: 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 – Fall 2026"

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/22546

(Canvas)

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

3. Ed

https://edstem.org/us/courses/103538/discussion

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

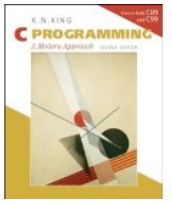
Etiquette

- 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

Books

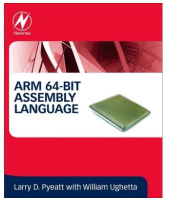
C Programming: A Modern Approach (Second Edition) (required)

- King
- C programming language and standard libraries



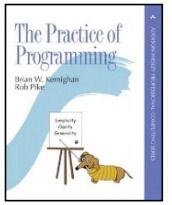
ARM 64-bit Assembly Language (required / online)

- 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: At least one every weekday; 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 CS Building:
  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 *	20
Exams During Term **	35
Final Exam ***	35
Participation ****	10

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

** One before Midterm week and one after– dates are on website.

*** During Finals Week – date on website

**** Lecture/precept attendance, precept/Ed participation that is helpful to course

Exams are in-person

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 a grade of 0
- Typical University-level penalty is probation or suspension

Questions?

Today's Agenda

Administrative things

- Introductions
- Resources
- Grading
- Policies

Our computing environment

- Key software / terminology
- Navigating the filesystem

ssh! While I **bash** this shell…

A quick COS217 ⟷ English dictionary so that we're on the same page

avcj.com

What's an 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

What's a Command Line 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.

Command Line Interface (CLI):

Text-based paradigm for interacting with programs.

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!

What's a Terminal and a Shell?

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.

GUI examples:

Mac finder / dock,
Windows file mgr / start menu

CLI example: `bash`

Waits for input, parses input into commands, executes commands, waits for next input.

What's `ssh`?

`ssh`:

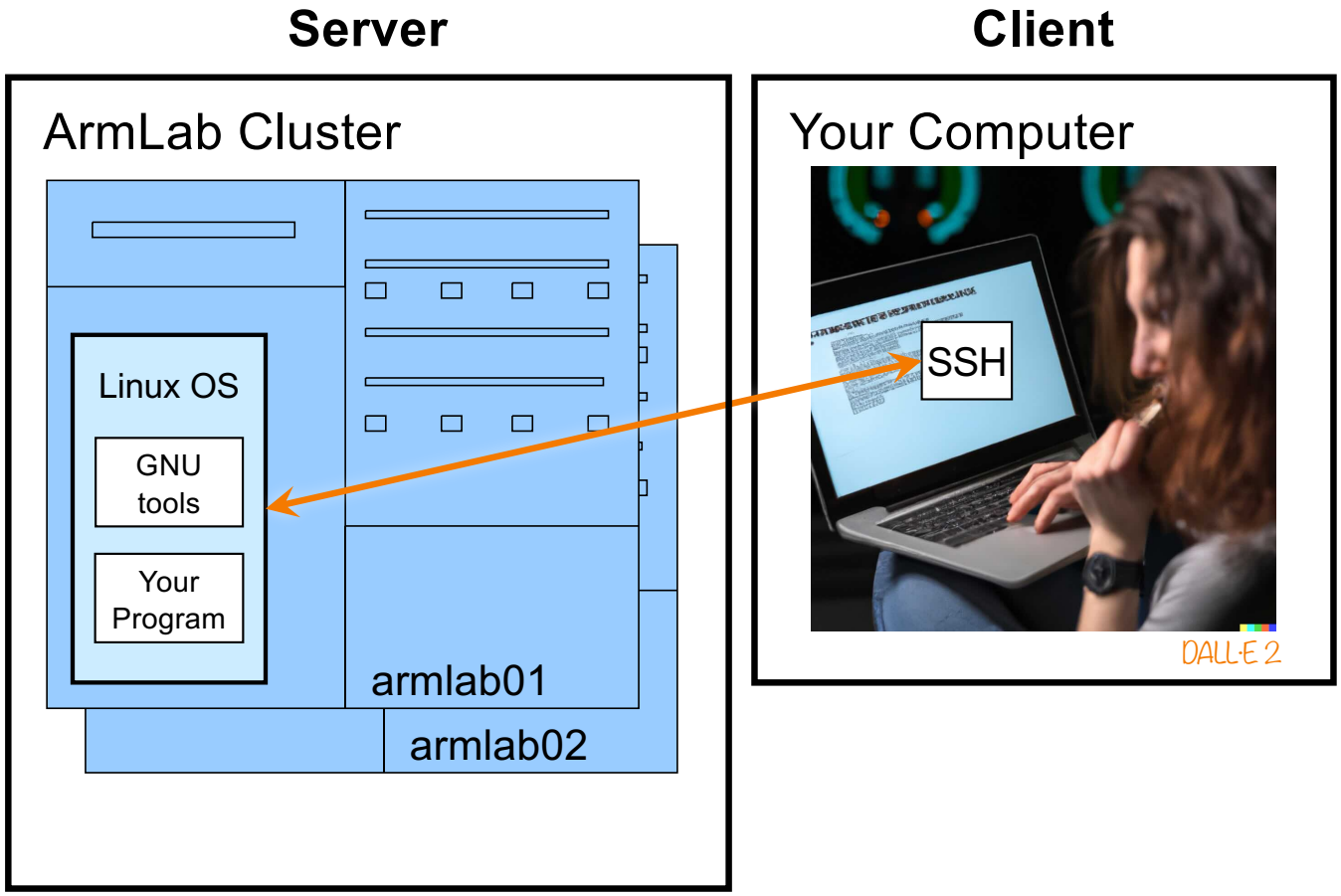
Stands for "secure shell" (but it's not a shell – it's a connection facilitator)

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

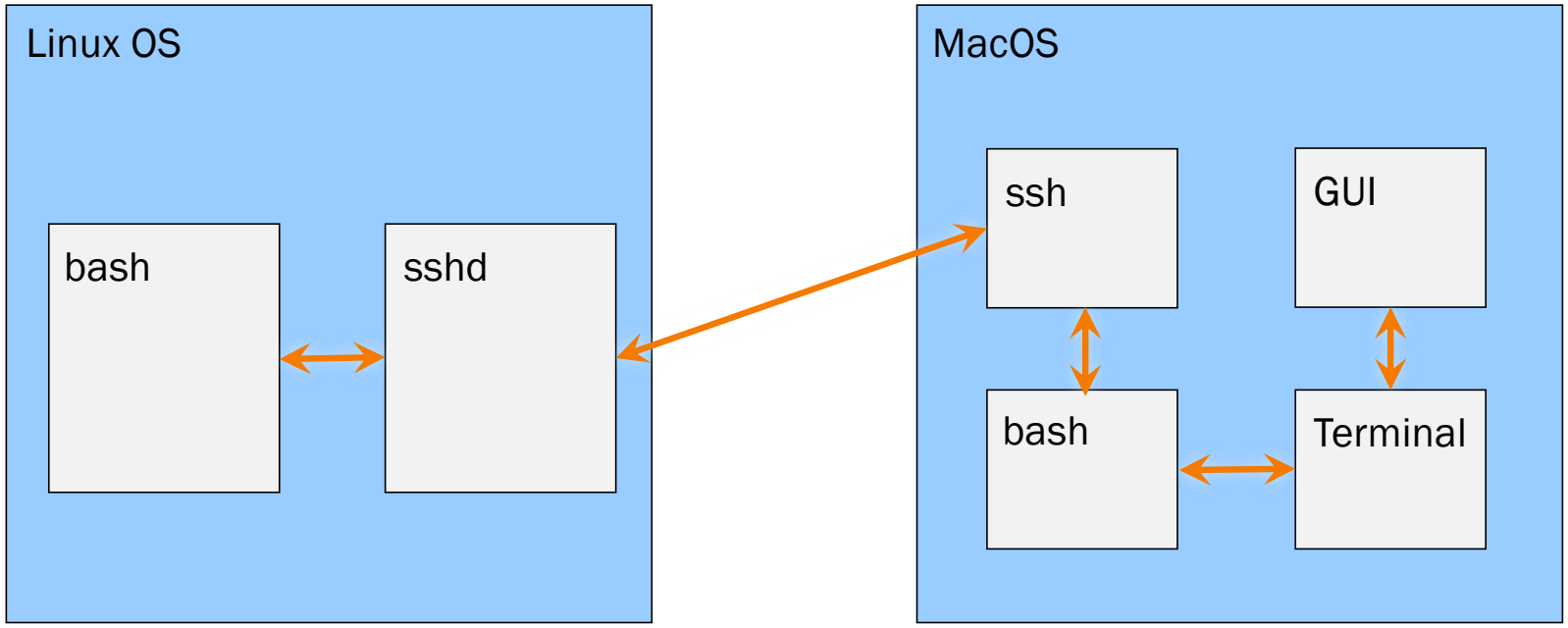
`sshd`:

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`).

Programming Environment – The Illusion



Programming Environment – The Reality



armlab

Your Mac computer
(Windows would be similar)

What's a Text Editor?

Text Editor:

Allows editing *plain text*: just a sequence of characters.

Examples: TextEdit, Notepad, Sublime Text, emacs, vi

Increasingly, customized for code

Word Processor:

Allows editing text with formatting (various fonts, paragraphs, etc.)

Does *not* output plain-text. Please don't submit these files.

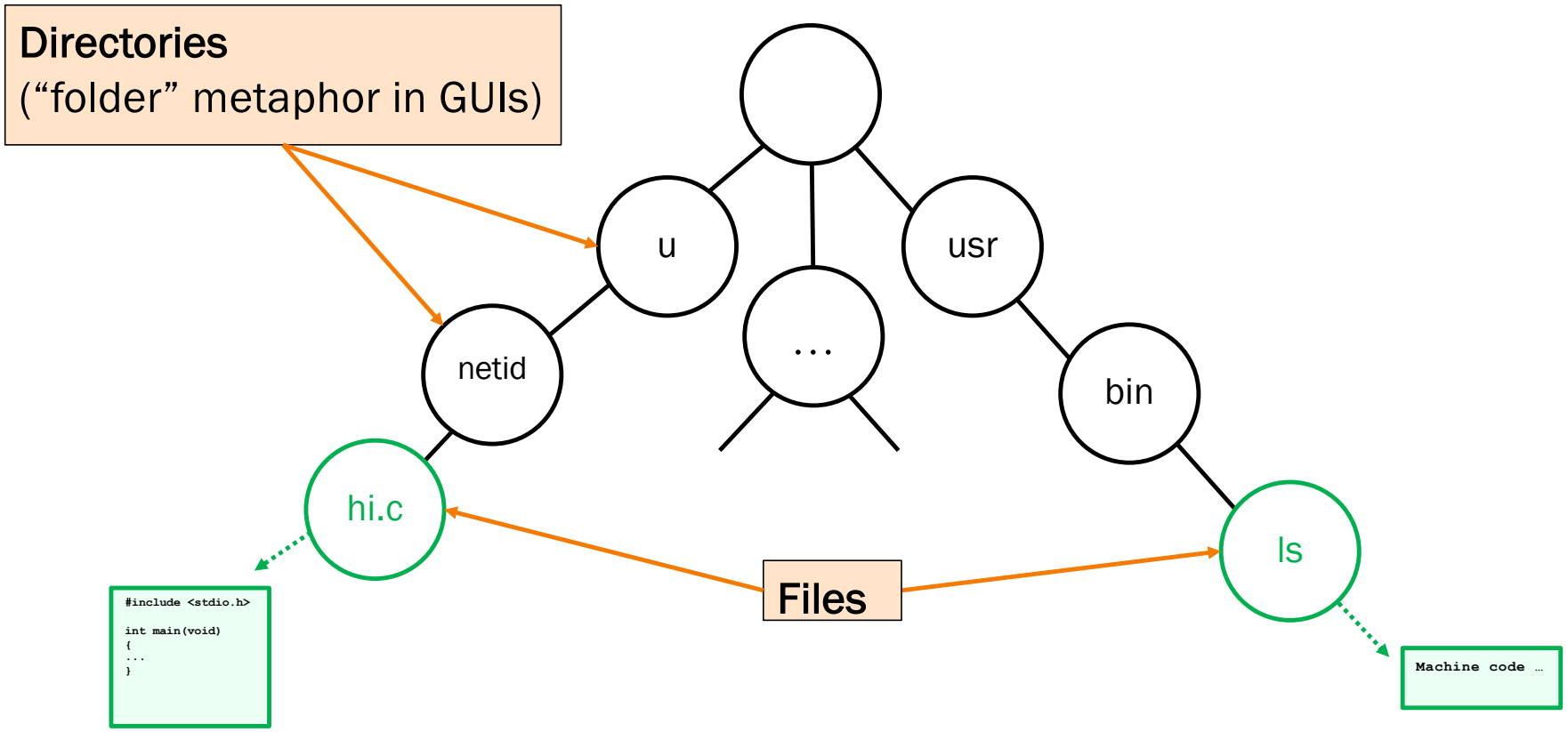
Examples: Word, Pages

Integrated Development Environment (IDE):

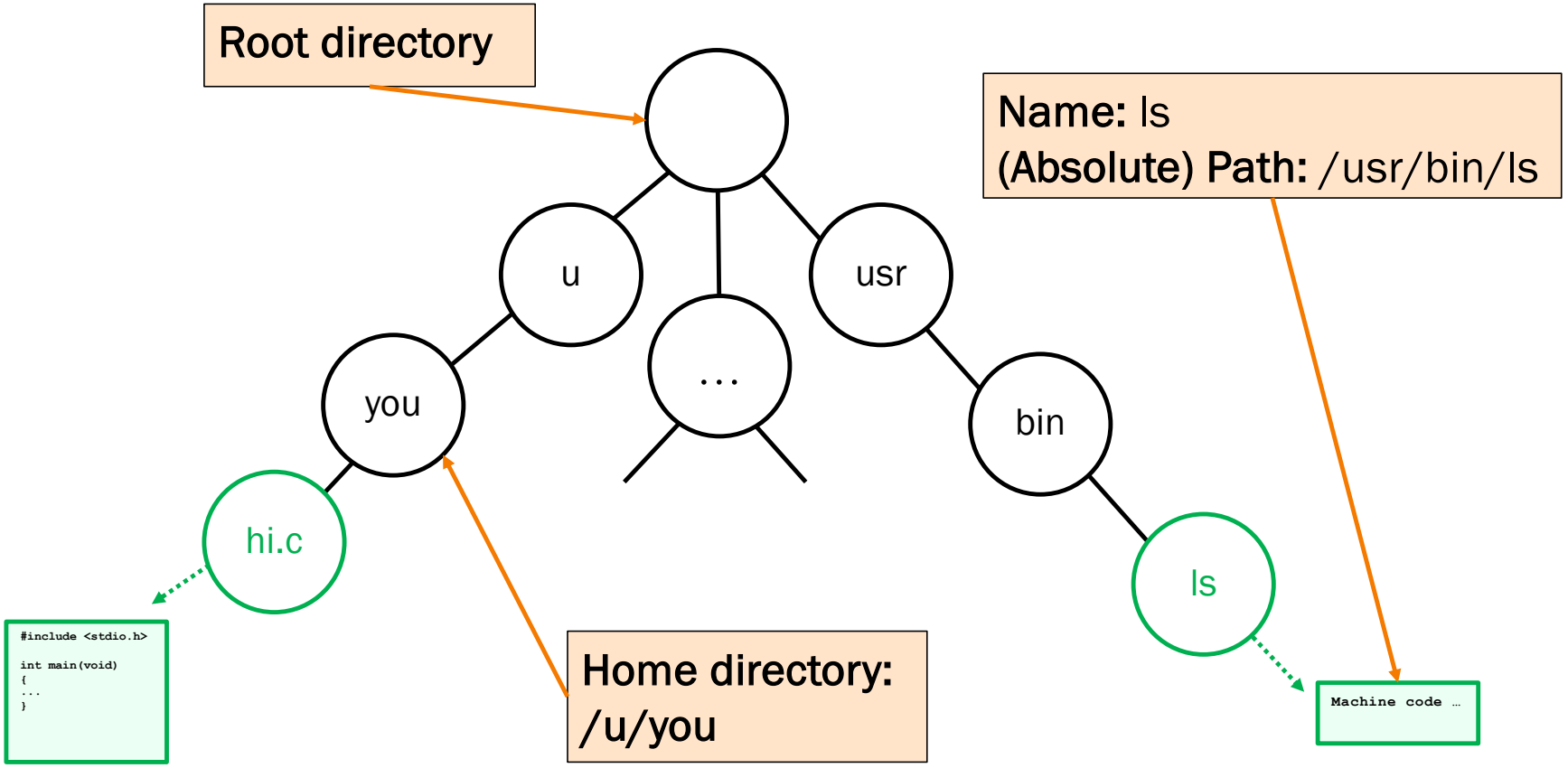
Text editor optimized for code – usually integrates syntax coloring, compiling, searching for errors, sometimes suggesting variable names or code snippets.

Examples: IntelliJ, VS Code, {emacs, vi} *with the appropriate configuration*

Filesystems



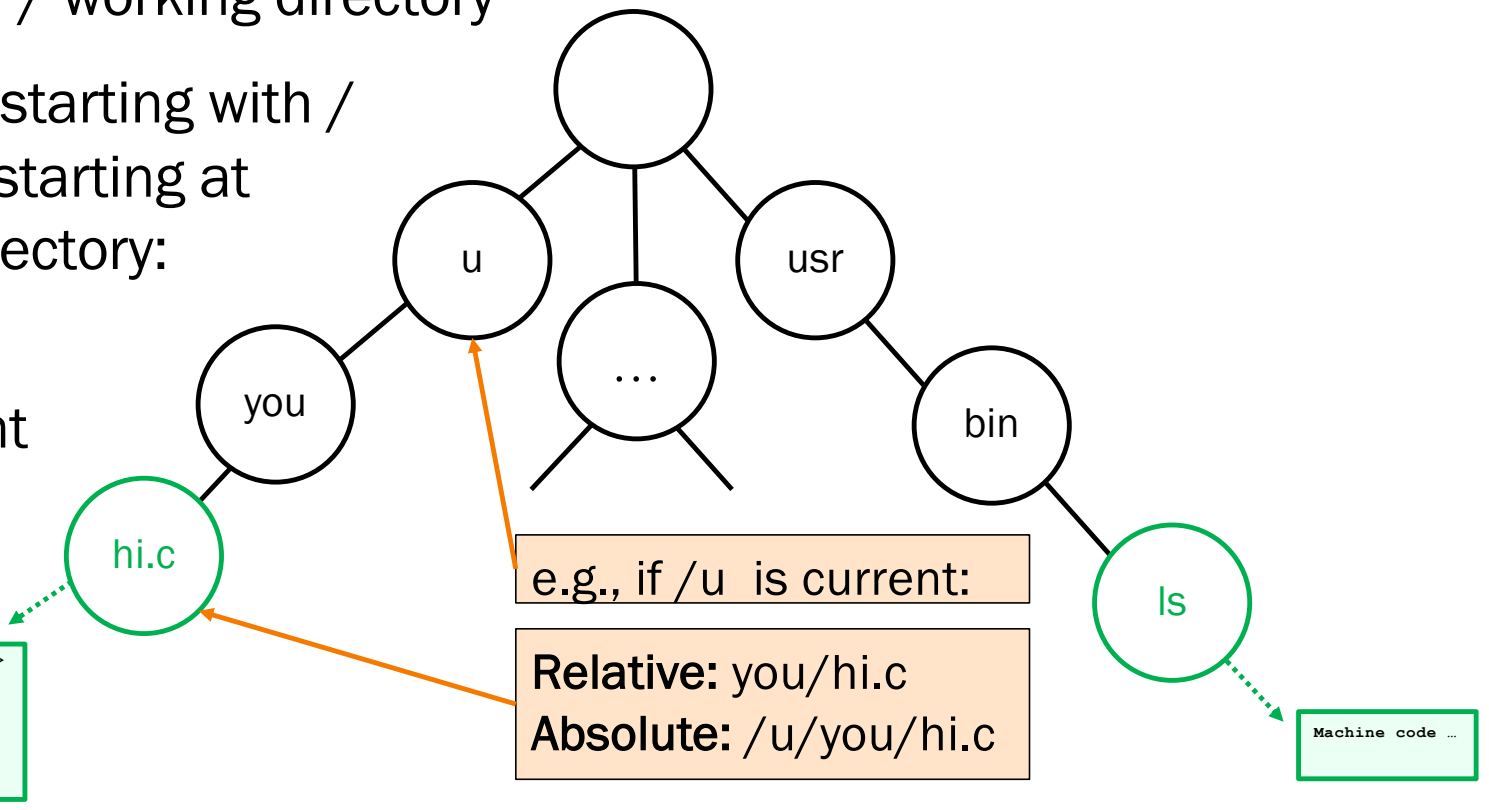
Filesystems



Filesystems

Current directory / working directory

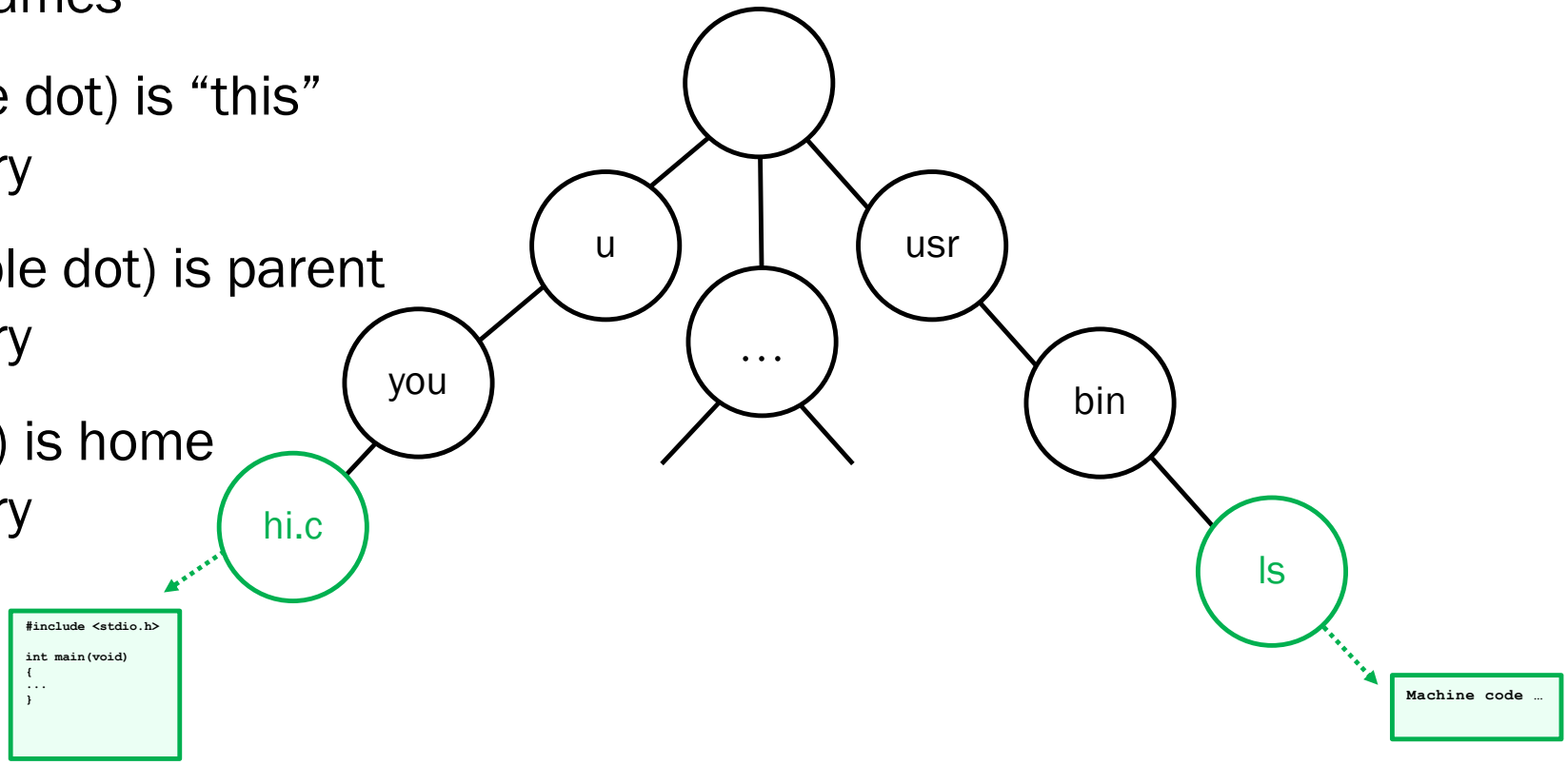
- Any name *not* starting with / is interpreted starting at the current directory: *relative* path
- Change current directory with *cd* command



Filesystems

Special names

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



Next steps ...

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

Upcoming lectures: Git, C programming

- In preparation for assignments 0 and 1