```
$ 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", 2023);
   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 2023
```

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

Our computing environment

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

Introductions

Instructor of Record

- Christopher Moretti — cmoretti@cs.princeton.edu

Faculty Preceptor

- Donna Gabai — dgabai@cs.princeton.edu

Graduate Preceptors

- Samuel Ginzburg — ginzburg@cs.princeton.edu
- Guðni Nathan Gunnarsson — gudni.nathan@princeton.edu
- Jianan Lu — jiananl@princeton.edu
- Wei Luo — wl4563@princeton.edu
- Ashwini Raina — araina@cs.princeton.edu
- Wei Tang — weit@princeton.edu

Agenda

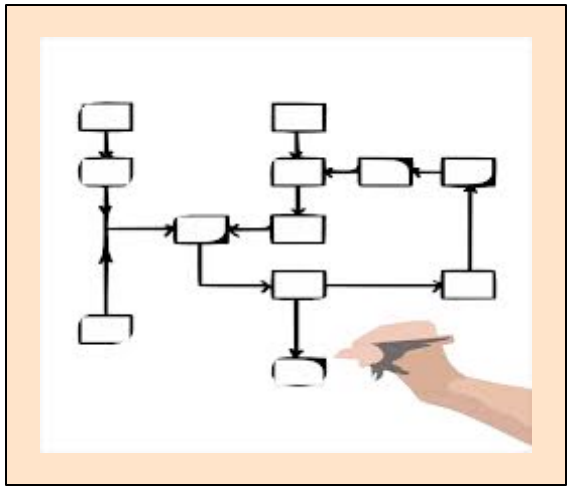
Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

Our computing environment

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

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, git, gcc, make, gdb, gprof, valgrind, splint

Along the Way: Learn Linux

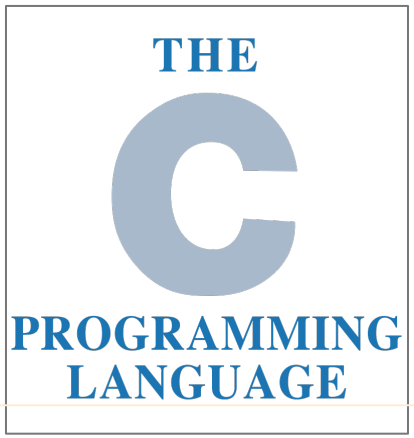
Question: Why use the Linux operating system?

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)

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.......
```

Along the Way: Learn C

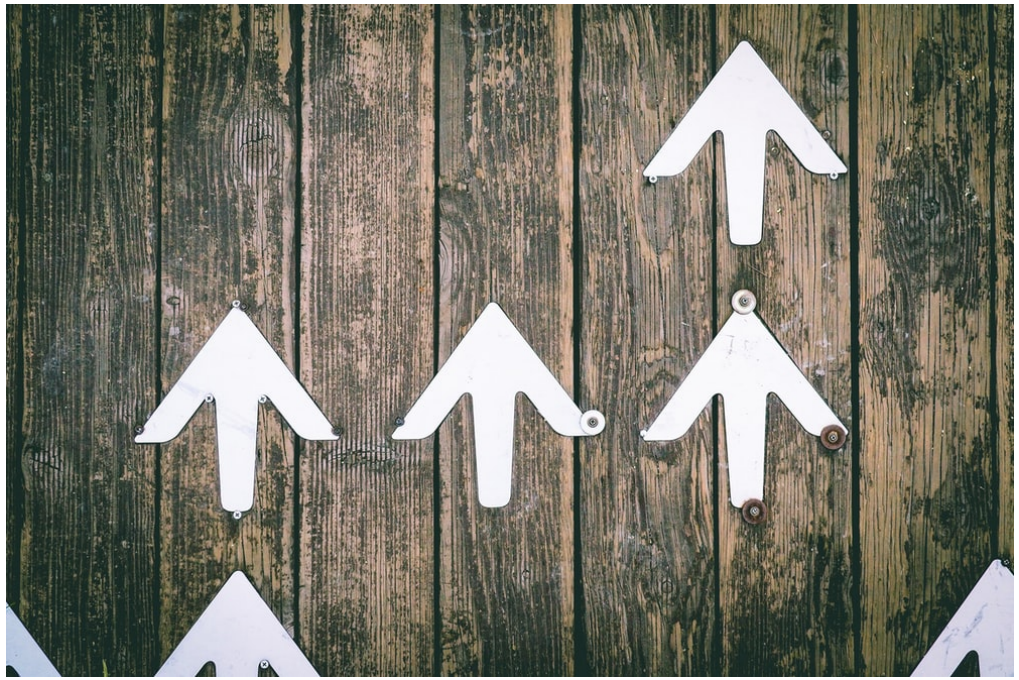
Question: Why C instead of Java?

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"

Goals: Summary

Help you to gain …



Jungwoo Hong

Programming Maturity

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

Our computing environment

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

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

Videos from some previous offerings are available on Youtube

Etiquette

- Ask questions as they come up!
- Use electronic devices *primarily* for taking notes or annotating slides
- Limit your SnapFaceInstaRedGooTok, please -- for yourself and your neighbors

iClicker

Occasional questions in class, graded on participation not correctness.

- Using 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 2023"

iClicker Question

Q: Can you answer this iClicker question 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 also available via course website

Etiquette

- Attend your precept: attendance will be taken
- Use TigerHub to move to another precept if timing is a problem
- Must miss your precept once or twice? ⇒ inform preceptors & attend another

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/9659

(Canvas)

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

Ed

https://edstem.org/us/courses/36005/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!
  - See course policies
  - Click “private” if in doubt – we can make it public after-the-fact

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.

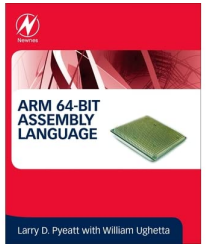
Books

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

- King
- C programming language and standard libraries

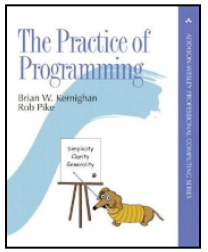
ARM 64-bit Assembly Language (required / online)

- Pyeatt with Ughetta



The Practice of Programming (online)

- Kernighan & Pike
- "Programming in the large"



Help!

Office Hours

- 3+ hours per day 7 days per week: some in-person, some Zoom
- "Concepts" office hours after lecture: focus on course material, not debugging
- Schedule is on the course website
- Zoom office hours links to queue form and status page are on Canvas

Intro COS Lab Hours

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

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

Our computing environment

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

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.

*** 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

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

@jdent

Violations of course policies

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

Mental Health

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 all feel stressed, anxious, uncertain at times – 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! Many of us have been there!

Questions?

Agenda

Course overview

- Introductions
- Course goals
- Resources
- Grading
- Policies

Our computing environment

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

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 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?

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, and *easier to execute remotely*.

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`

What's `ssh`?

`ssh:`

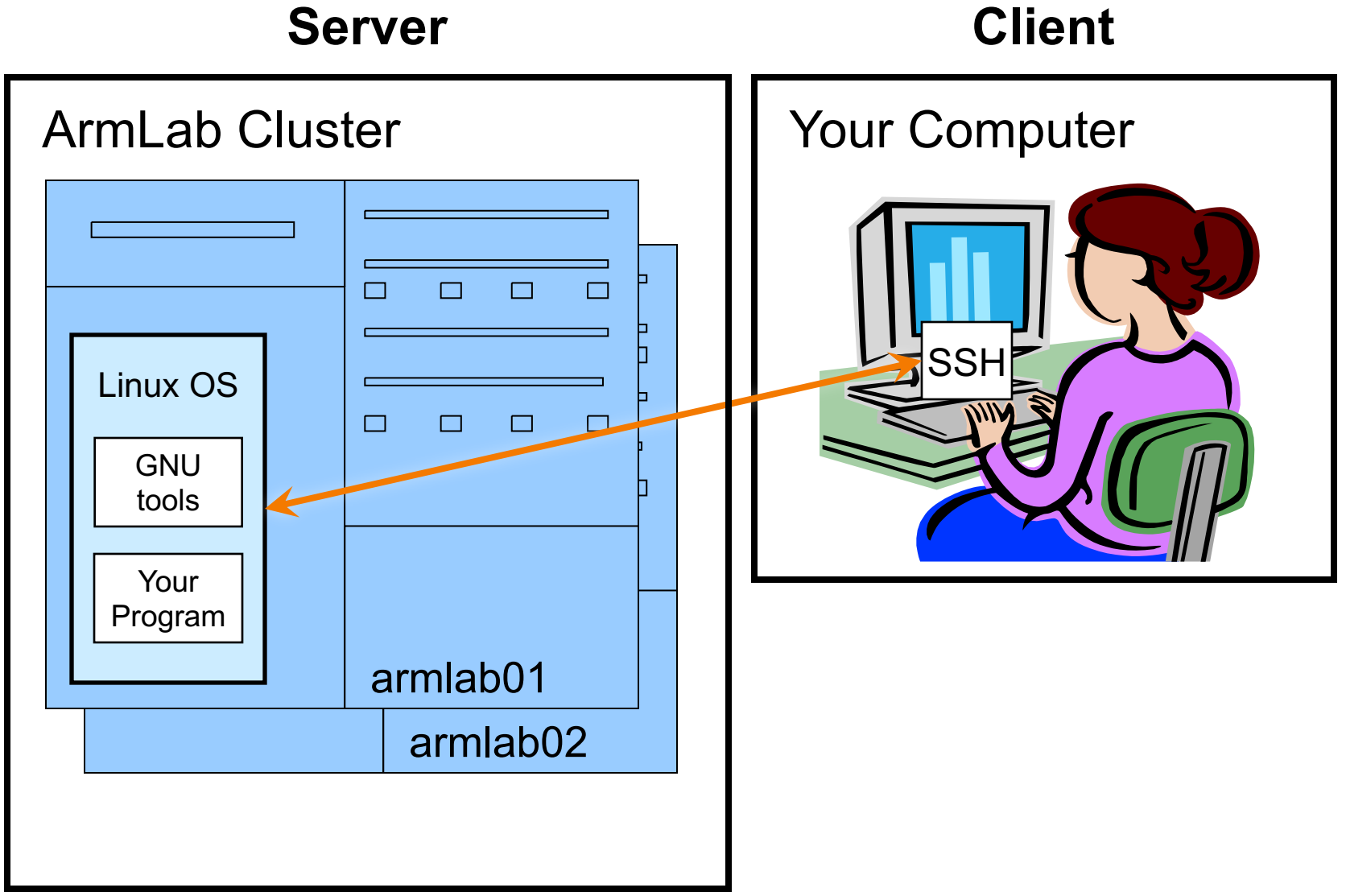
Stands for "secure shell" but it's not a shell!

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

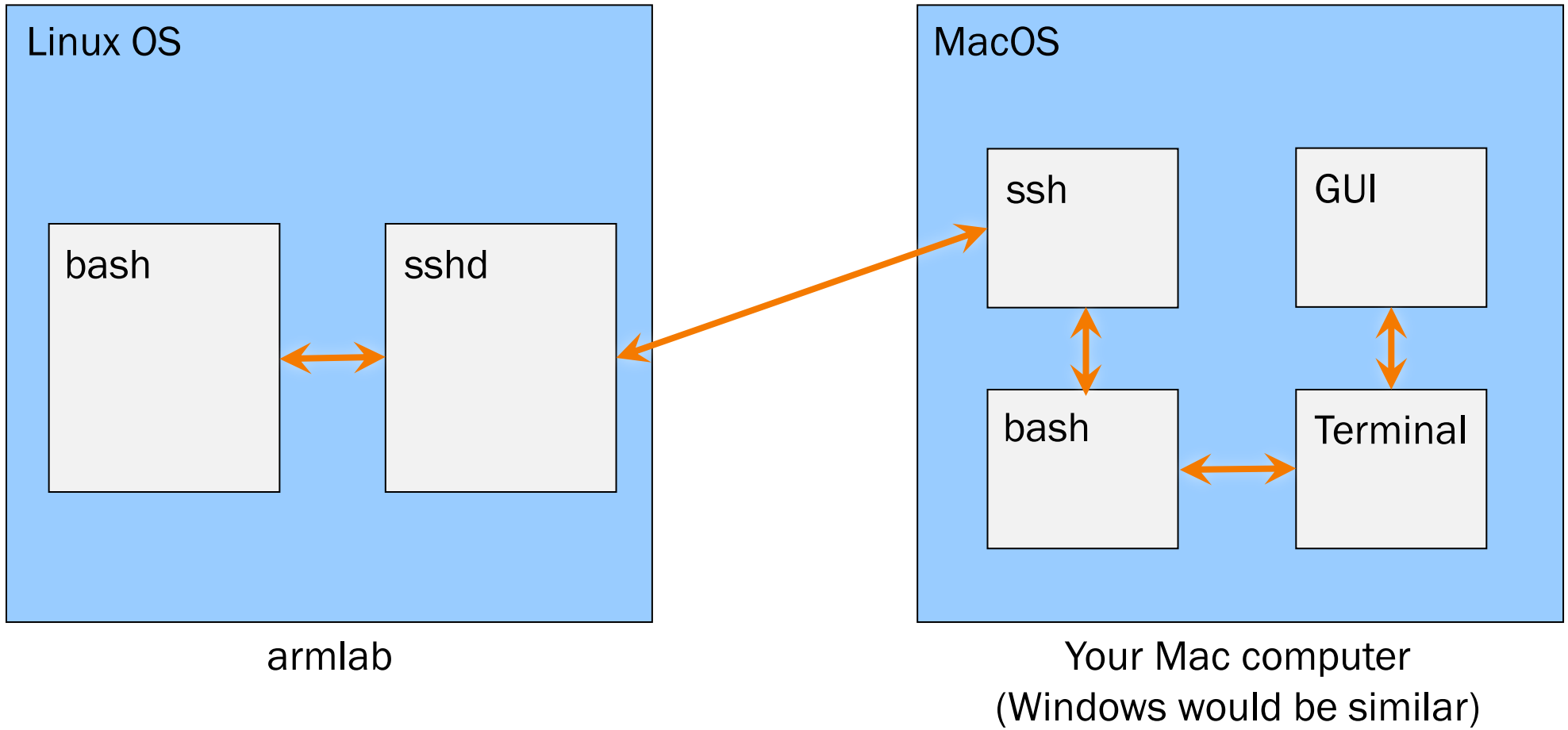
`sshd:`

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`).

Programming Environment – The Illusion



Programming Environment – The Reality



What's a Text Editor?

Text Editor:

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

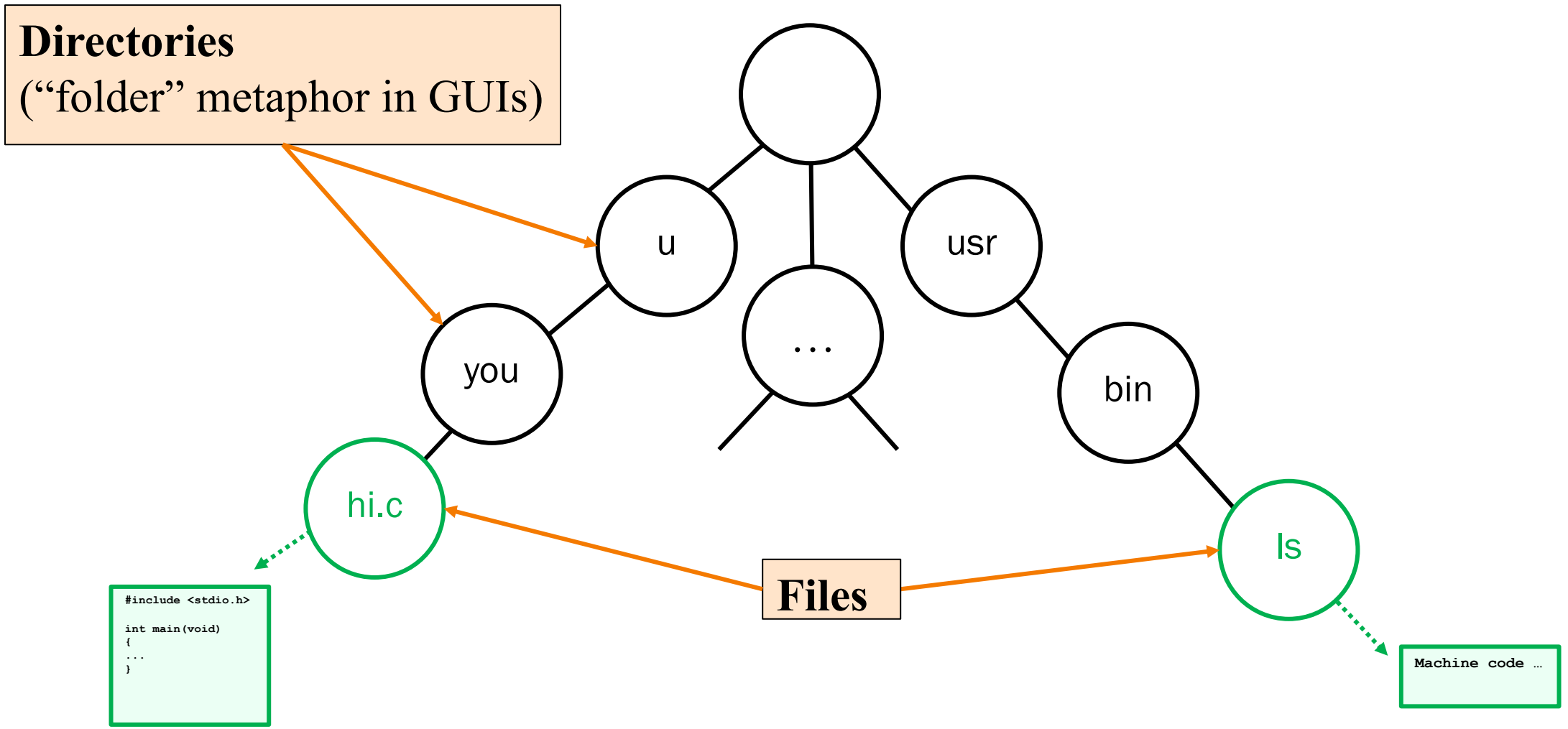
Word Processor:

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

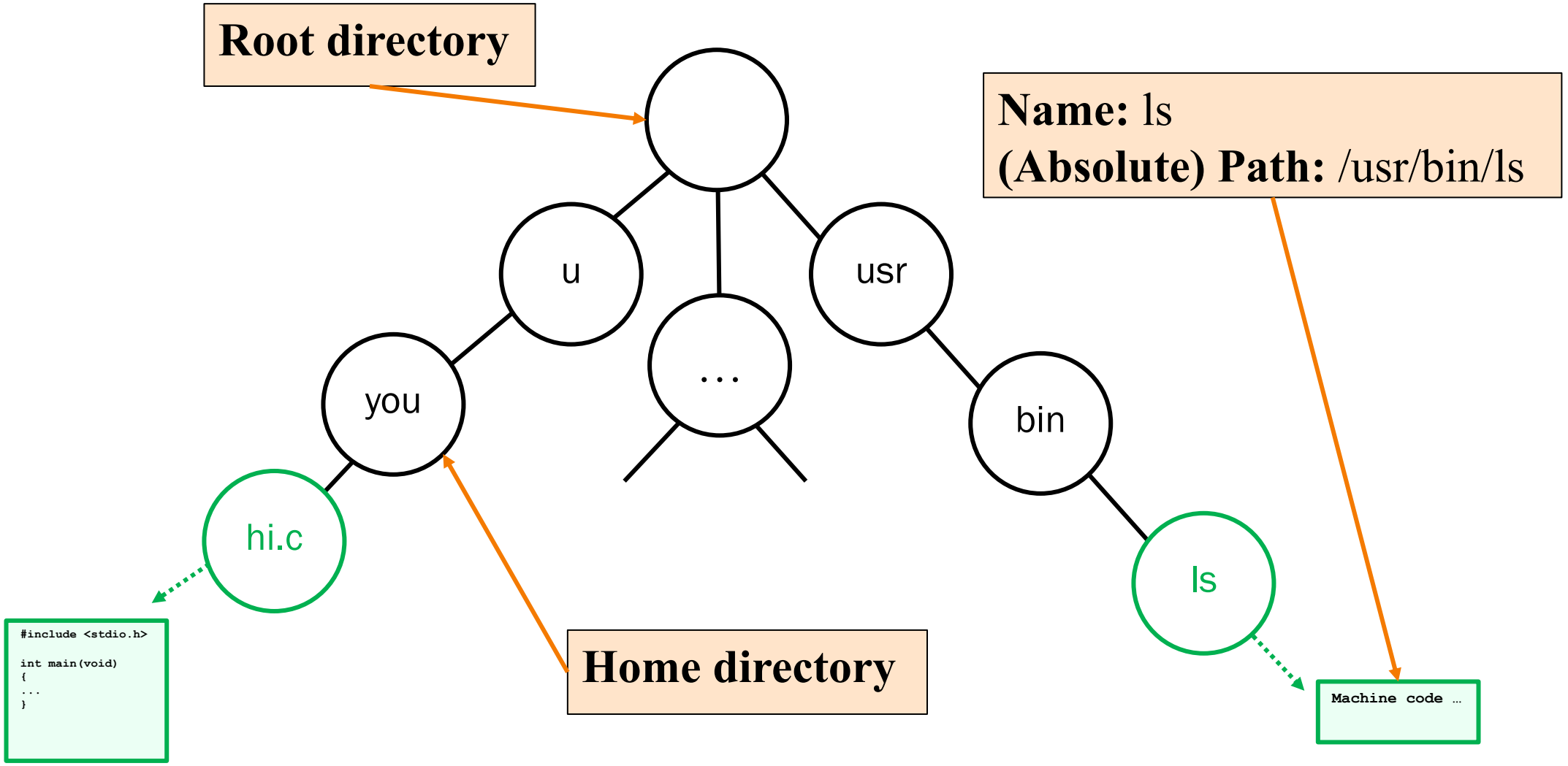
Integrated Development Environment (IDE):

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

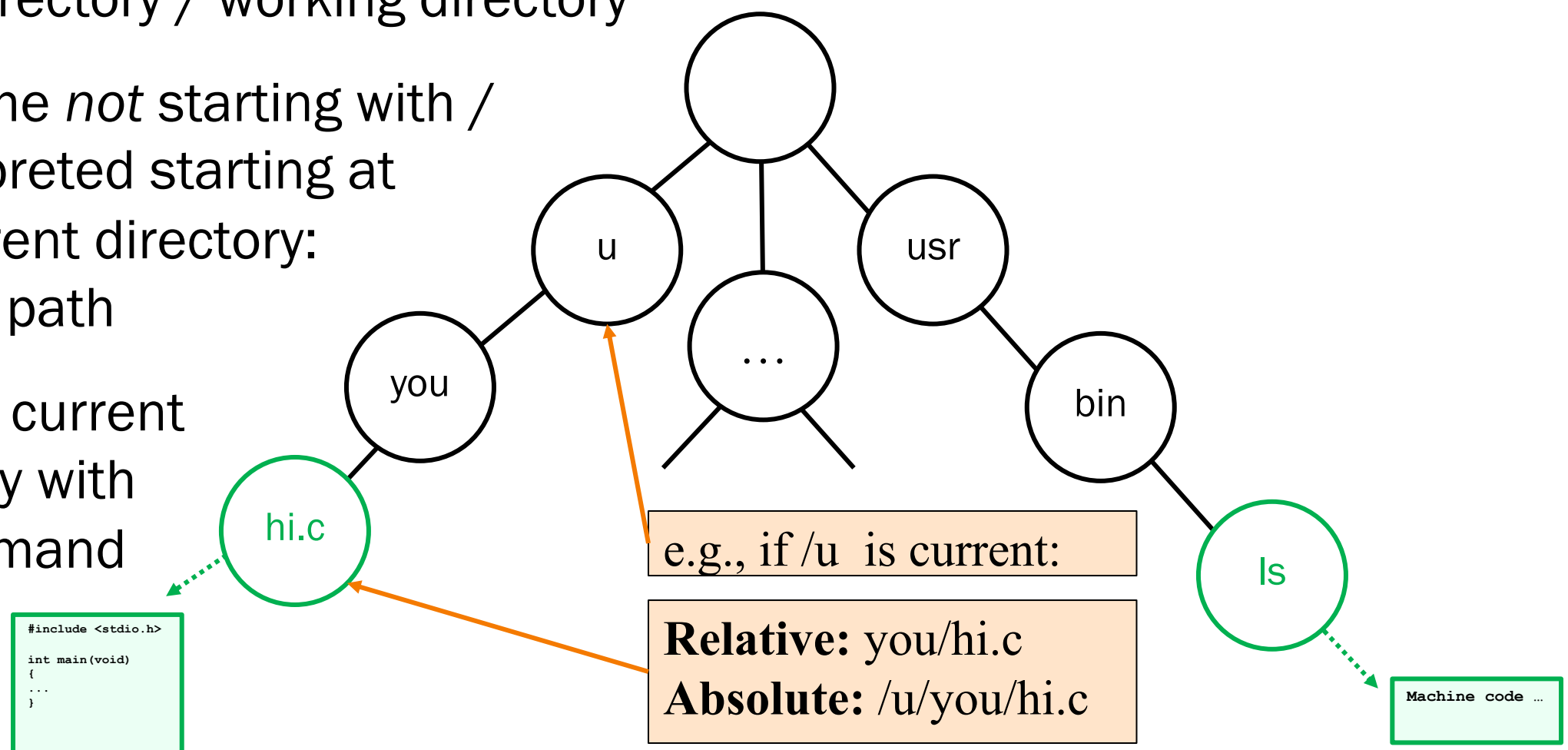
Filesystems



Filesystems



Filesystems



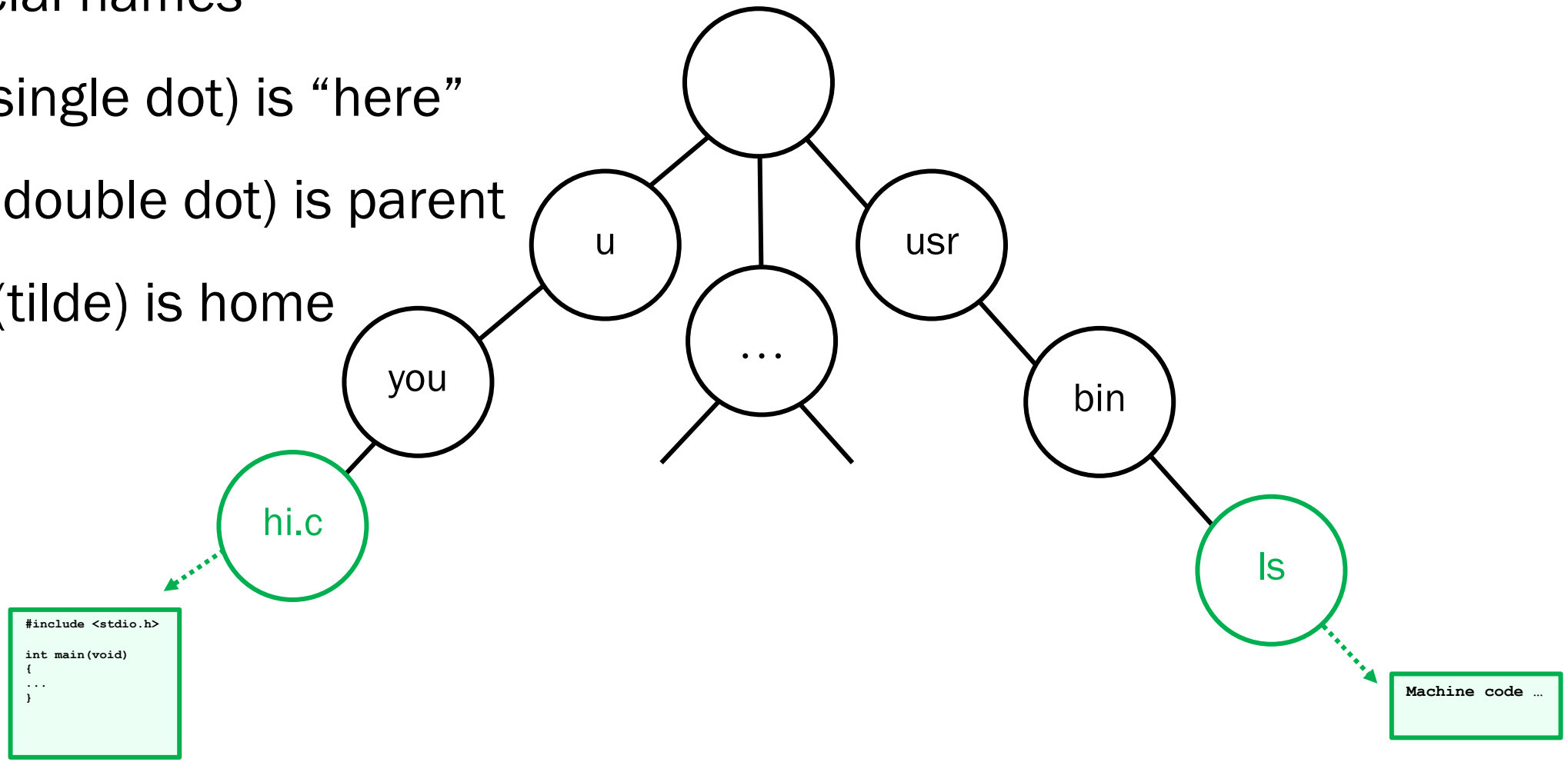
Current directory / working directory

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

Filesystems

Special names

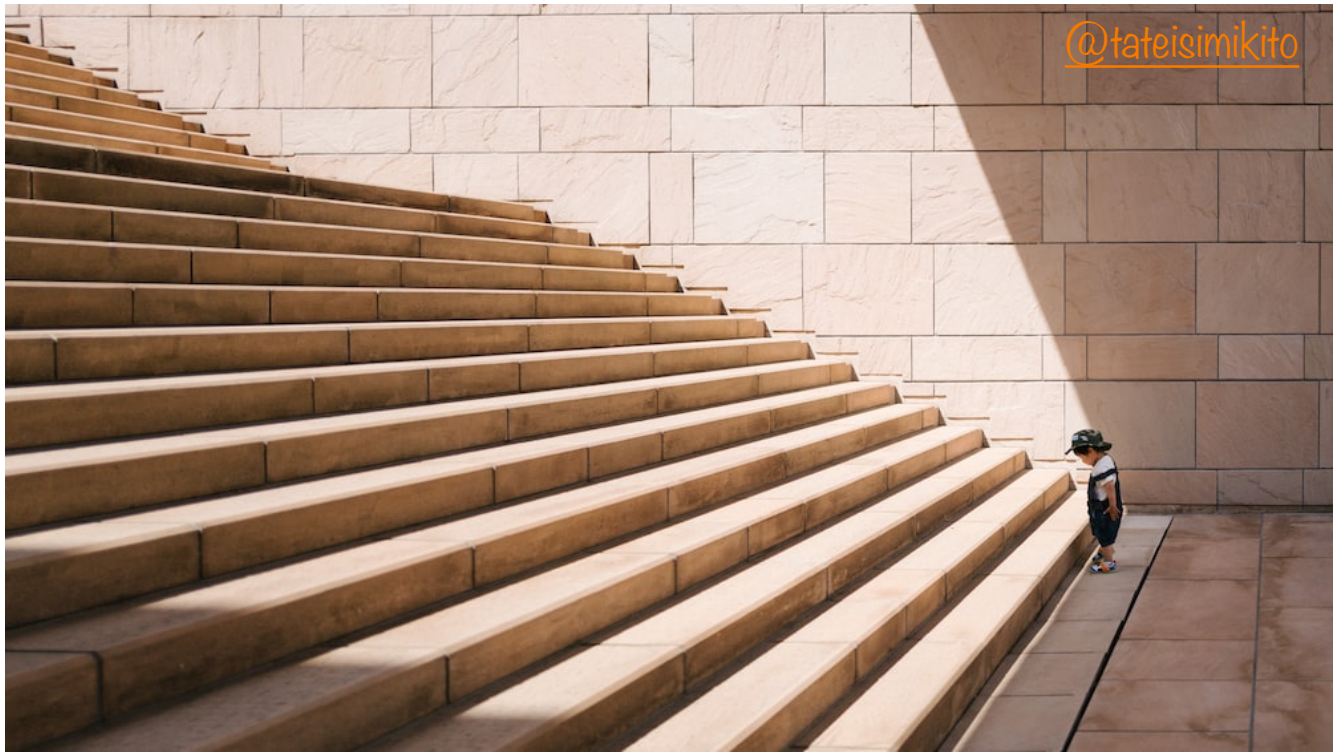
- . (single dot) is “here”
- .. (double dot) is parent
- ~ (tilde) is home



Next steps …

Check out course website **soon**

- Study "Policies" page



Upcoming lectures: Git, C programming

- In preparation for assignments 0 and 1