CS 211: Intro to Computer Architecture

1.2: C, Linux, and ilab

Minesh Patel

Spring 2025 – Thursday 23 January

Agenda

- **More Logistics**
- **Part 1:** C programming
- **Part 2:** Linux
- **Part 3:** iLab Demo

Announcements

- **Survey #1 is out** (linked on both Canvas + Website)
  - Lots of MCQs, but I filled it out in an average (N=5) of 94 seconds
  - Please help us help yourself
- **PA1 will be released tonight** (announcements through Ed + Canvas)
  - We are testing Google Docs as a dynamically-editable format
  - We will post comments into the Doc + make announcements for any clarifications
- **Ed should be working**
  - We have enrolled the entire Canvas roster
  - We will make announcements through it
- **GradeScope enrollment is in progress**

Note on Course Communication

- **Recall:** this course is about **fundamentals**
- The only “bad” question is the one you don’t ask- no judgement from us

- Where to seek answers:
  - Ed discussion
  - Instructor mailing list
  - Recitations
  - Office hours
  - Department / university resources
  - Online material

Office Hours and Recitations

- Both start next week!

- Announcements through Ed (+Canvas, for now)

Note on Adds/Drops

- **We cannot influence** course enrollment – contact the department

- Being added to course platforms (Ed, GradeScope) is **manual**
  - We are adding the entire canvas roster,
  - Please be patient

- Please e-mail the instructional mailing list IFF:
  - You recently added/dropped this course
  - You aren't added by Week 2

Note on Programming Assignment Grading

- We are still designing PA2-PA6
  - PA1 does not have any programming – just the submission process

- Grading plan:
  - **Correctness**
    - **Supplied tests cases:** provided with the assignment code
    - **Hidden test cases:** run during grading only
  - **Code quality** (maybe- we will specify precisely if we do this)
    - Presence of important compiler warnings and errors
    - Egregious use of error-prone code styling

- We do not have instantaneous autograders (we are looking into it)
  - You can run the supplied test cases as many times as you like
  - Only your "final" single submission will be graded

Agenda

- More Logistics
- **Part 1:** C programming
- **Part 2:** Linux
- **Part 3:** iLab Demo

Layers of Abstraction for CS 211

Real-world Demands

Problem

Algorithm

Program

CS 415: Compilers

Runtime

CS 416: Operating Systems

System software

HW/SW Interface (ISA, ABI)

Microarchitecture

Logic gates

Circuits

Technology

Physics

Real-world Constraints

High-Level Language Program

```
int i;
for(i = 0; i < 10; i++)
    a[i] = b[i] * c[i];
```

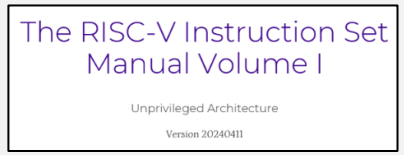
Assembly Language Program

```
loop:  ld   a4, 0(a5)
       ld   a3, 8(a5)
       addi a5, a5,   -8
       add  a4, a4,   a3
       sd   a4, 8(a5)
       bne  a0, a5,   loop
```

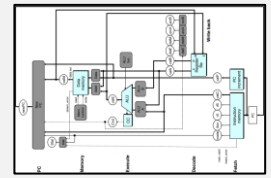
Machine Language Program

```
0000_0000_0000_0111_1011_0111_0000_0011
0000_0000_1000_0111_1011_0110_1000_0011
1111_1111_1000_0111_1000_0111_1001_0011
0000_0000_1101_0111_0000_0111_0011_0011
0000_0000_1110_0111_1011_0100_0010_0011
1111_1110_1111_0101_0001_0110_1110_0011
```

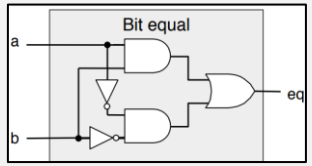
Instruction Set Architecture (ISA)



Simple RISC-V Processor



Logic Circuits



Layers of Abstraction for CS 211

Real-world Demands

Problem

Algorithm

Program

Runtime

System software

HW/SW Interface (ISA, ABI)

Microarchitecture

Logic gates

Circuits

Technology

Physics

Real-world Constraints

High-Level Language Program

```
int i;
for(i = 0; i < 10; i++)
    a[i] = b[i] * c[i];
```

Assembly Language Program

```
loop:  ld   a4, 0(a5)
       ld   a3, 8(a5)
       addi a5, a5,  -8
       add  a4, a4,  a3
       sd   a4, 8(a5)
       bne  a0, a5,  loop
```

Machine Language Program

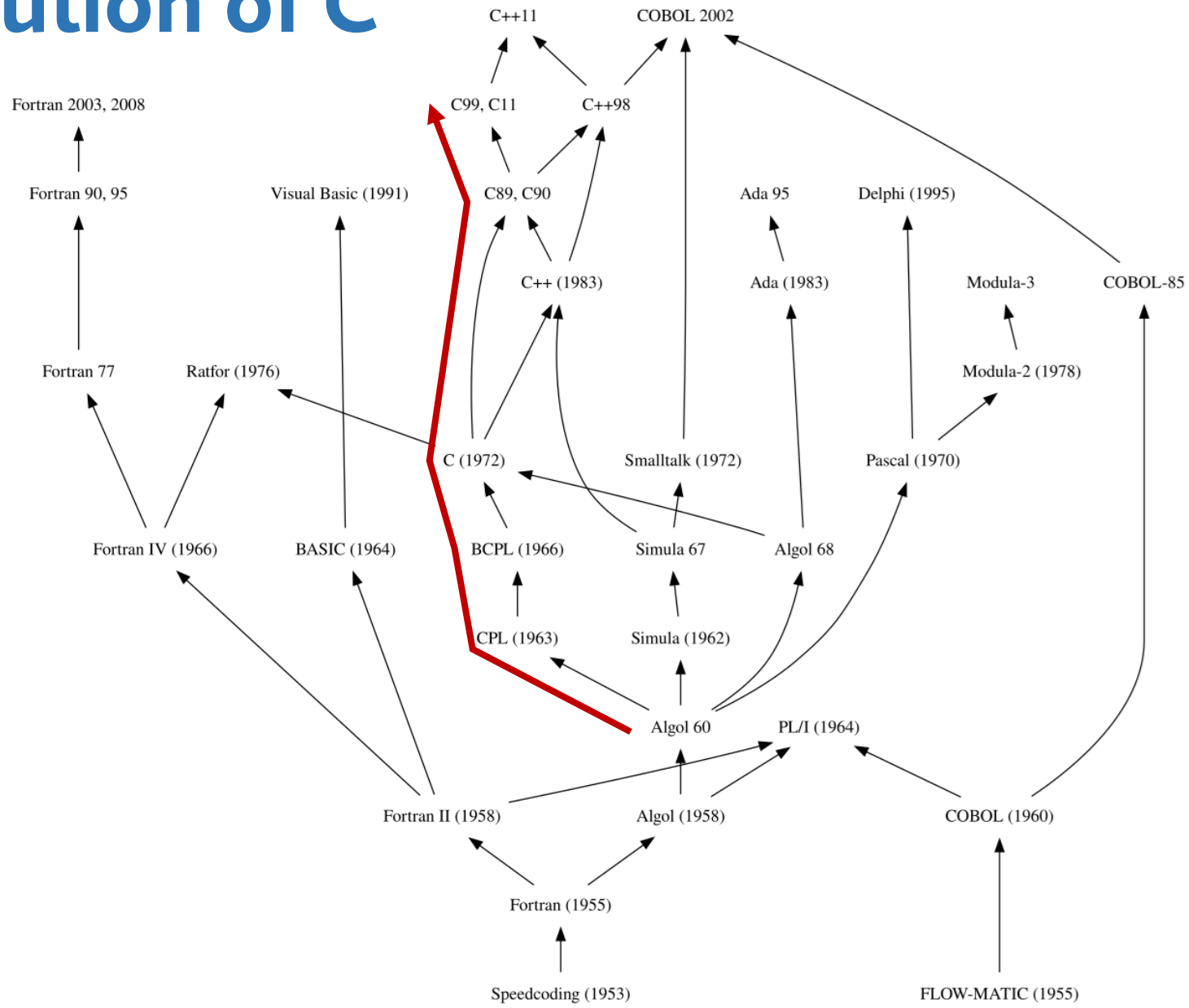
```
0000_0000_0000_0111_1011_0111_0000_0011
0000_0000_1000_0111_1011_0110_1000_0011
1111_1111_1000_0111_1000_0111_1001_0011
0000_0000_1101_0111_0000_0111_0011_0011
0000_0000_1110_0111_1011_0100_0010_0011
1111_1110_1111_0101_0001_0110_1110_0011
```

Instruction Set Architecture (ISA)

Microarchitecture

Logic Circuits

Evolution of C



- 1966: BCPL
- 1969: B
- 1972: C
- 1978: K&R C 1/e
- 1988 K&R C 2/e
- 1989: ANSI C
- 1999: C99
- 2011: C11 **This course**
- 2017: C17
- 2024+: C23

https://upload.wikimedia.org/wikipedia/commons/8/84/Algol%26Fortran_family-by-Borkowski.svg

IEEE Spectrum's Popularity Analysis

The Top Programming Languages 2024 › Typescript and Rust are among the rising stars

BY STEPHEN CASS | 22 AUG 2024 | 3 MIN READ |

Stephen Cass is the special projects editor at IEEE Spectrum.

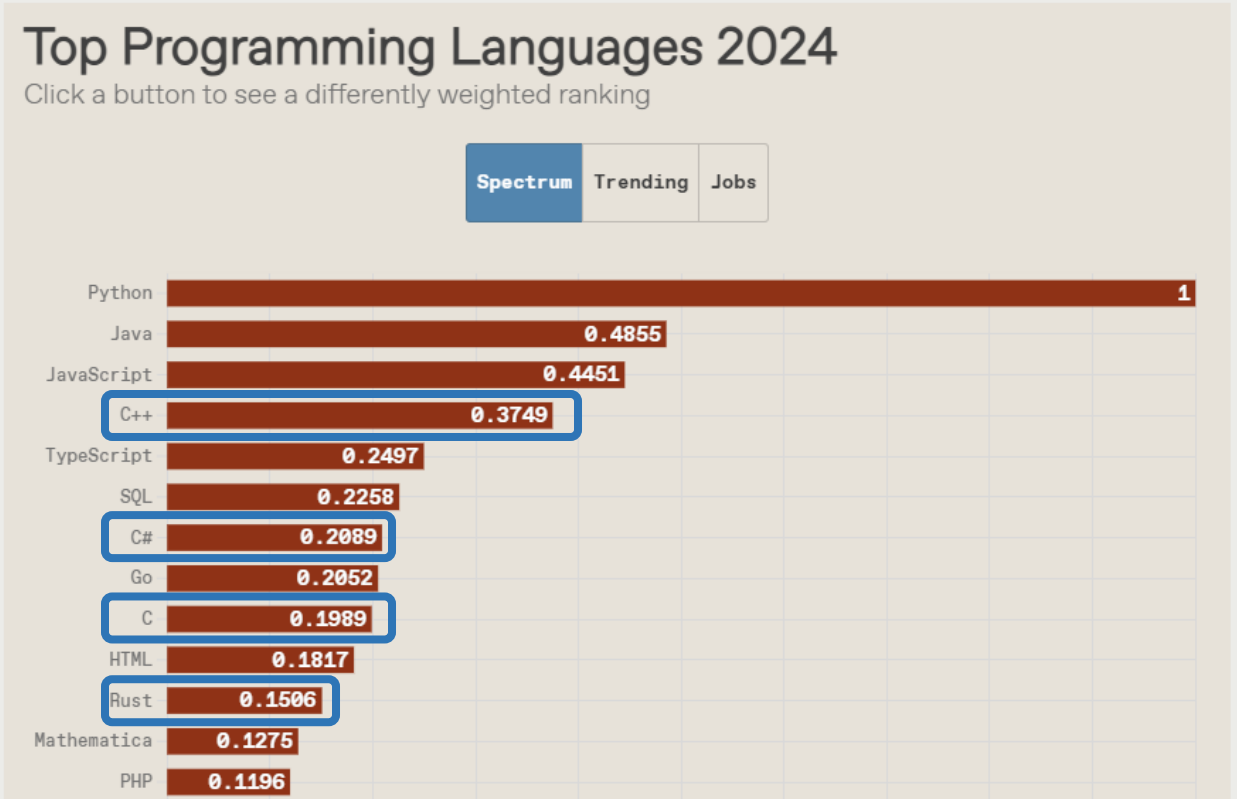
SHARE THIS STORY

TAGS

TOP PROGRAMMING LANGUAG…

PYTHON | SQL | RUST

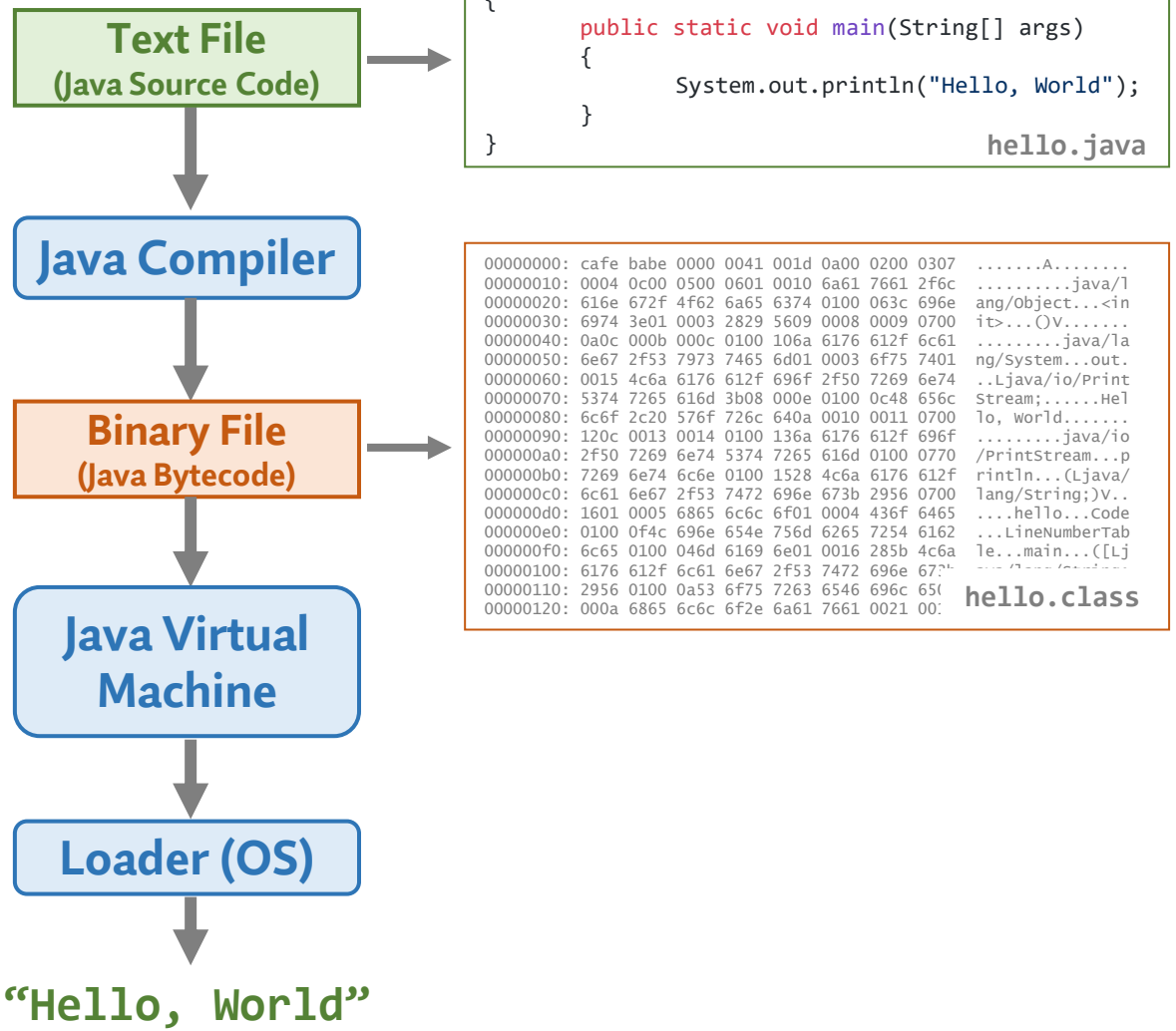
TYPESCRIPT



https://spectrum.ieee.org/top-programming-languages-2024

From Source Code to Executable

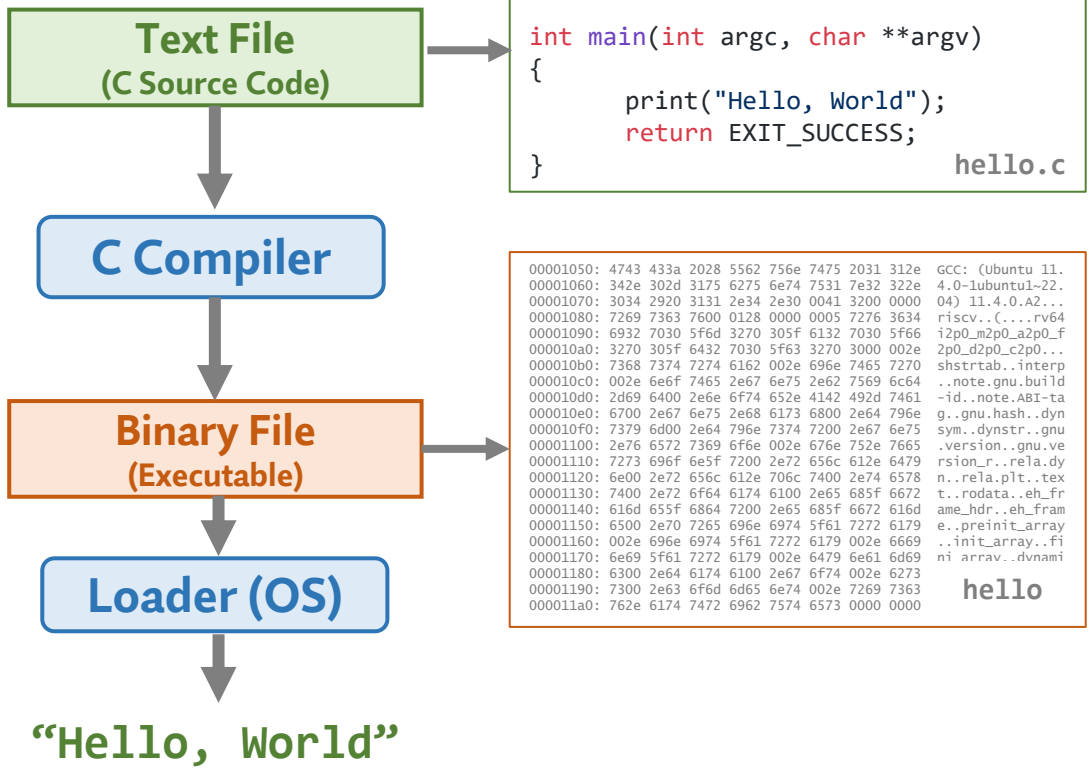
Java




```
public class hello
{
        public static void main(String[] args)
        {
                System.out.println("Hello, World");
        }
}
```
hello.java

```
00000000: cafe babe 0000 0041 001d 0a00 0200 0307  .......A........
00000010: 0004 0c00 0500 0601 0010 6a61 7661 2f6c  ..........java/l
00000020: 616e 672f 4f62 6a65 6374 0100 063c 696e  ang/Object...<in
00000030: 6974 3e01 0003 2829 5609 0008 0009 0700  it>...()V.......
00000040: 0a0c 000b 000c 0100 106a 6176 612f 6c61  .........java/la
00000050: 6e67 2f53 7973 7465 6d01 0003 6f75 7401  ng/System...out.
00000060: 0015 4c6a 6176 612f 696f 2f50 7269 6e74  ..Ljava/io/Print
00000070: 5374 7265 616d 3b08 000e 0100 0c48 656c  Stream;......Hel
00000080: 6c6f 2c20 576f 726c 640a 0010 0011 0700  lo, World.......
00000090: 120c 0013 0014 0100 136a 6176 612f 696f  .........java/io
000000a0: 2f50 7269 6e74 5374 7265 616d 0100 0770  /PrintStream...p
000000b0: 7269 6e74 6c6e 0100 1528 4c6a 6176 612f  rintln...(Ljava/
000000c0: 6c61 6e67 2f53 7472 696e 673b 2956 0700  lang/String;)V..
000000d0: 1601 0005 6865 6c6c 6f01 0004 436f 6465  ....hello...Code
000000e0: 0100 0f4c 696e 654e 756d 6265 7254 6162  ...LineNumberTab
000000f0: 6c65 0100 046d 6169 6e01 0016 285b 4c6a  le...main...([Lj
00000100: 6176 612f 6c61 6e67 2f53 7472 696e 673b  ava/lang/String;
00000110: 2956 0100 0a53 6f75 7263 6546 696c 650   )V...SourceFile.
00000120: 000a 6865 6c6c 6f2e 6a61 7661 0021 0001  ..hello.java.!..
```
hello.class

C

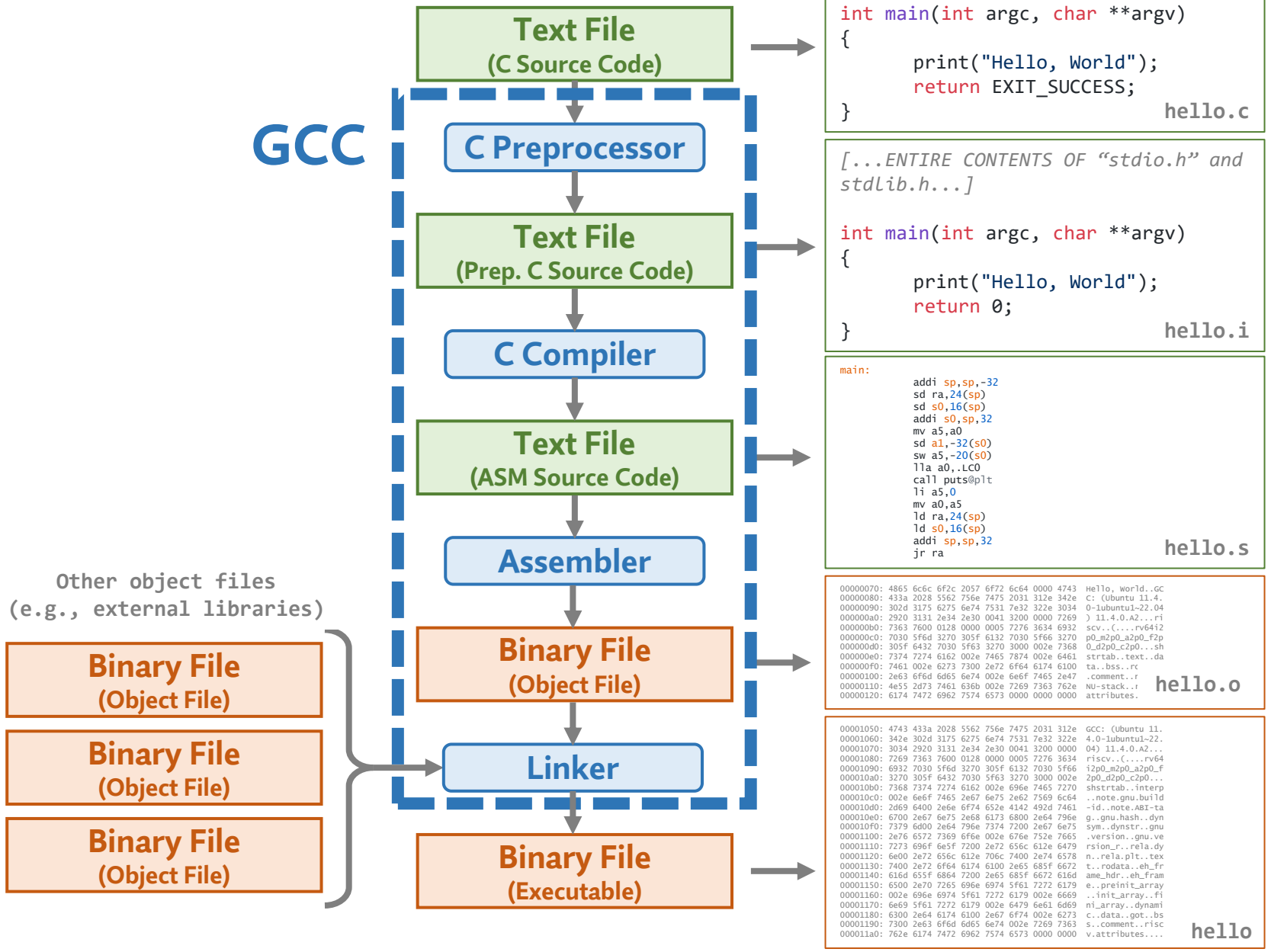



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
        print("Hello, World");
        return EXIT_SUCCESS;
}
```
hello.c

```
00001050: 4743 433a 2028 5562 756e 7475 2031 312e  GCC: (Ubuntu 11.
00001060: 342e 302d 3175 6275 6e74 7531 7e32 322e  4.0-1ubuntu1~22.
00001070: 3034 2920 3131 2e34 2e30 0041 3200 0000  04) 11.4.0.A2...
00001080: 7269 7363 7600 0128 0000 0005 7276 3634  riscv..(....rv64
00001090: 6932 7030 5f6d 3270 305f 6132 7030 5f66  i2p0_m2p0_a2p0_f
000010a0: 3270 305f 6432 7030 5f63 3270 3000 002e  2p0_d2p0_c2p0...
000010b0: 7368 7374 7274 6162 002e 696e 7465 7270  shstrtab..interp
000010c0: 002e 6e6f 7465 2e67 6e75 2e62 7569 6c64  ..note.gnu.build
000010d0: 2d69 6400 2e6e 6f74 652e 4142 492d 7461  -id..note.ABI-ta
000010e0: 6700 2e67 6e75 2e68 6173 6800 2e64 796e  g..gnu.hash..dyn
000010f0: 7379 6d00 2e64 796e 7374 7200 2e67 6e75  sym..dynstr..gnu
00001100: 2e76 6572 7369 6f6e 002e 676e 752e 7665  .version..gnu.ve
00001110: 7273 696f 6e5f 7200 2e72 656c 612e 6479  rsion_r..rela.dy
00001120: 6e00 2e72 656c 612e 706c 7400 2e74 6578  n..rela.plt..tex
00001130: 7400 2e72 6f64 6174 6100 2e65 685f 6672  t..rodata..eh_fr
00001140: 616d 655f 6864 7200 2e65 685f 6672 616d  ame_hdr..eh_fram
00001150: 6500 2e70 7265 696e 6974 5f61 7272 6179  e..preinit_array
00001160: 002e 696e 6974 5f61 7272 6179 002e 6669  ..init_array..fi
00001170: 6e69 5f61 7272 6179 002e 6479 6e61 6d69  ni_array..dynami
00001180: 6300 2e64 6174 6100 2e67 6f74 002e 6273  c..data..got..bs
00001190: 7300 2e63 6f6d 6d65 6e74 002e 7269 7363  s..comment..risc
000011a0: 762e 6174 7472 6962 7574 6573 0000 0000  v.attributes....
```
hello

Disclaimer: This is an oversimplification

C Compilation



Exploring the HW-SW Interface

- C is a great learning tool because it can **break the program abstraction**

1

C has precise **hardware representations** for every **language construct**

```
uint32_t scalar_sum(
        uint64_t a, uint64_t b[N])
{
    uint32_t i;
    for(i = 0; i < 100; i++)
        b[i] += a;
}
```

2

C **exposes those representations** to the programmer

Data

`uint64_t a`	64 bits	$\{ b_{63}\text{-}b_0 \}$ e.g., 00000000101101111000011110110011
`uint64_t b[N]`	N * 64 bits	$\{ (b_{63}\text{-}b_0)^{N-1} \ldots (b_{63}\text{-}b_0)^0 \}$
`uint32_t i`	32 bits	$\{ b_{31}\text{-}b_0 \}$
`a[i] += b;`	3 * 32 bits	00000000000001010011011110000011 00000000101101111000011110110011 11111110111101010011110000100011
`scalar_sum` (entire function)	7 * 32 bits	00110010000001010000011100010011 00000000000001010011011110000011 00000000100001010000010100010011 00000000101101111000011110110011 11111110111101010011110000100011 11111110111001010001100011100011 00000000000000001000000001100111

Code

Exploring the HW-SW Interface

- C is a great learning tool because it can **break the program abstraction**

1

C has precise **hardware representations** for every **language construct**

2

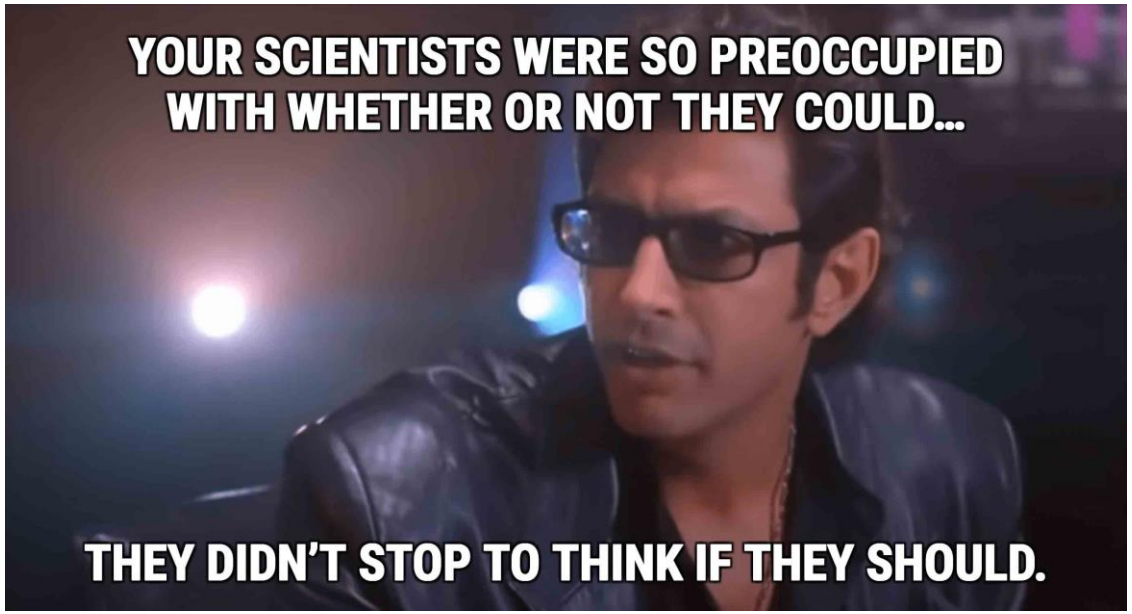
C **exposes those representations** to the programmer

C enables <u>thinking</u> and <u>acting</u> how the computer does

Strictly for learning purposes ☺

Efficiency-Security Tradeoff

- Manipulating hardware representations of data can be **highly efficient**
  - No "unnecessary" **safety checks** or **data format conversions**
  - **Nothing in the way** between your code and the hardware



- This can be **a very dangerous game**
  - Poor C code can contain sinister bugs and security vulnerabilities
  - Take *CS 419: Computer Security* for more on that

Goal of C in CS 211

- This class will **not** make you a proficient C programmer
  - We will learn **just enough C** to study how programs map to hardware
  - Pointers, arrays, structures, explicit memory management
- Then we will move on to the **next abstraction level** (asm + machine code)

We want to understand this

What we see

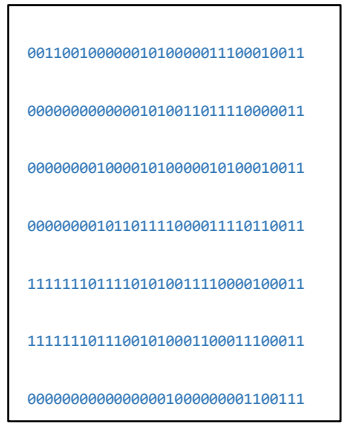
```
void scalar_sum(long a[], long b)
{
    int i;
    for(i = 0; i < 100; i++)
        a[i] += b;
}
```

Text File:
scalar_sum.c

"The Compiler"

GCC, LLVM
- C preprocessor
- C compiler
- C assembler
- C linker

What the computer sees




```
// addi     a4,a0,800
// ld       a5,0(a0)
// addi     a0,a0,8
// add      a5,a5,a1
// sd       a5,-8(a0)
// bne      a0,a4,4
// ret
```

Binary File:
scalar_sum.o

(Lesser) Benefits of C

- **Main benefit:** C is a great learning tool
  - Established 1972 and still in wide use
  - It enabled UNIX: the first OS not written directly using assembly

- It is also:
  - **Performance-oriented:** very little gets in between you and the hardware
    - Performance predictability!
  - **Everywhere:** at the lowest-level of (most) systems software
  - **Influential:** inspired C++, C#, Obj-C, Java, Python, etc.
  - **Simple:** short, concise specification

Agenda

- **More Logistics**
- **Part 1:** C programming
- **Part 2:** Linux
- **Part 3:** iLab Demo

Linux is an Operating System

...

Linux **MacOS** **Windows** **Android**

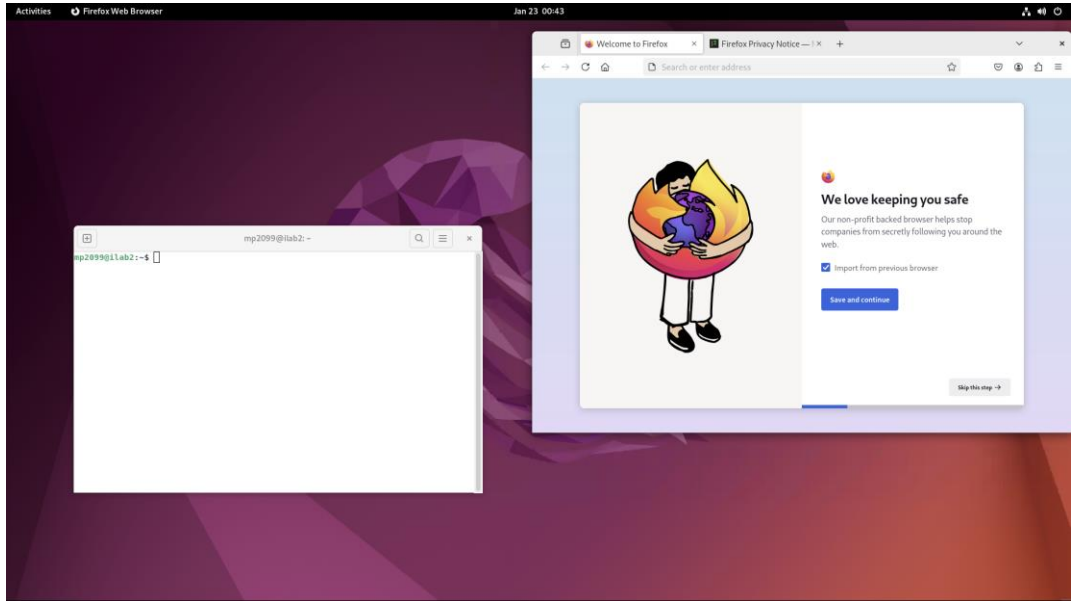
- You can do the basics on any of them
  - Create, edit, and modify files
  - Build applications (C, Java, etc.)
  - Install, run, and debug applications
  - ...

https://images.logicalincrements.com/gallery/1920/822/macos-vs-windows-vs-linux-operating-system-os-guide-display-image.webp

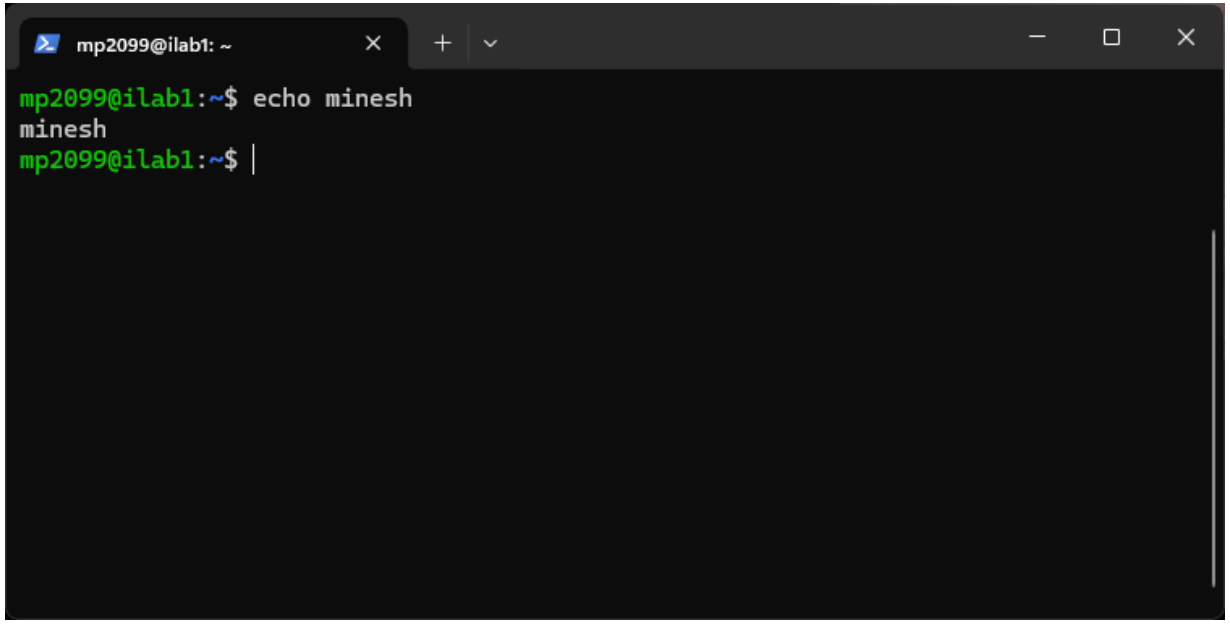
https://upload.wikimedia.org/wikipedia/commons/4/4b/Android14Home.png

Why Linux?

- Once again, **it's a great learning tool**
  - Full transparency into the entire software stack
  - Decades of open-source tools for exploring just about anything

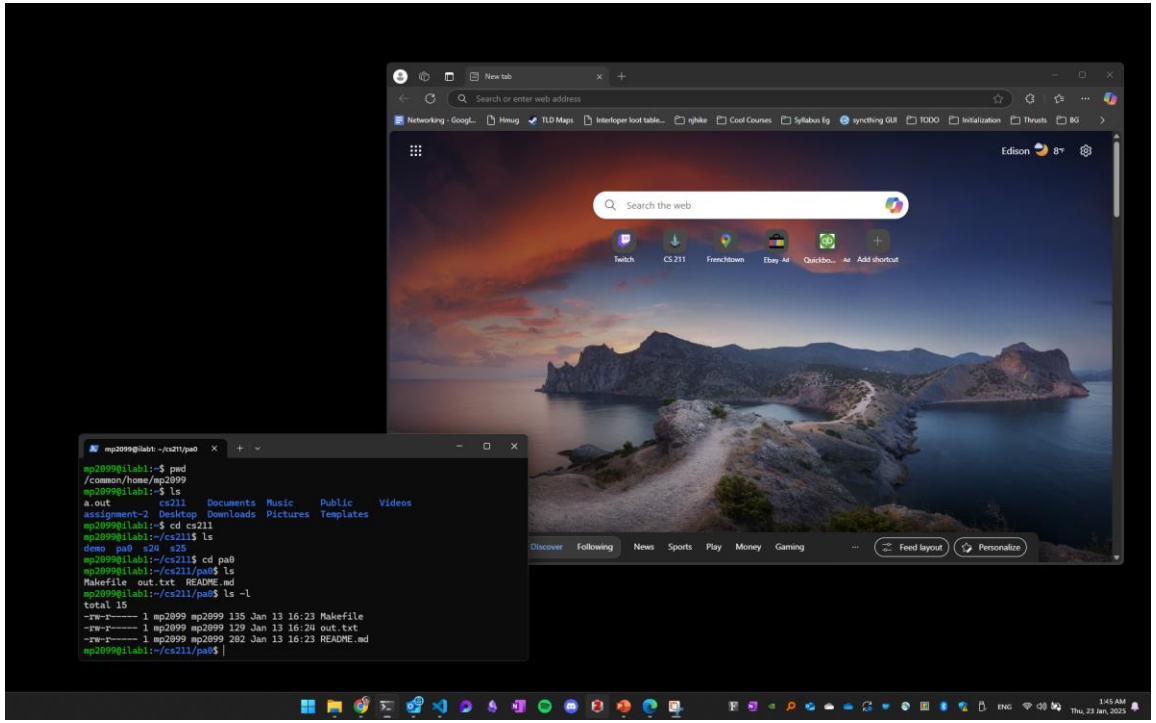


Using Linux with
a Graphical User Interface (GUI)

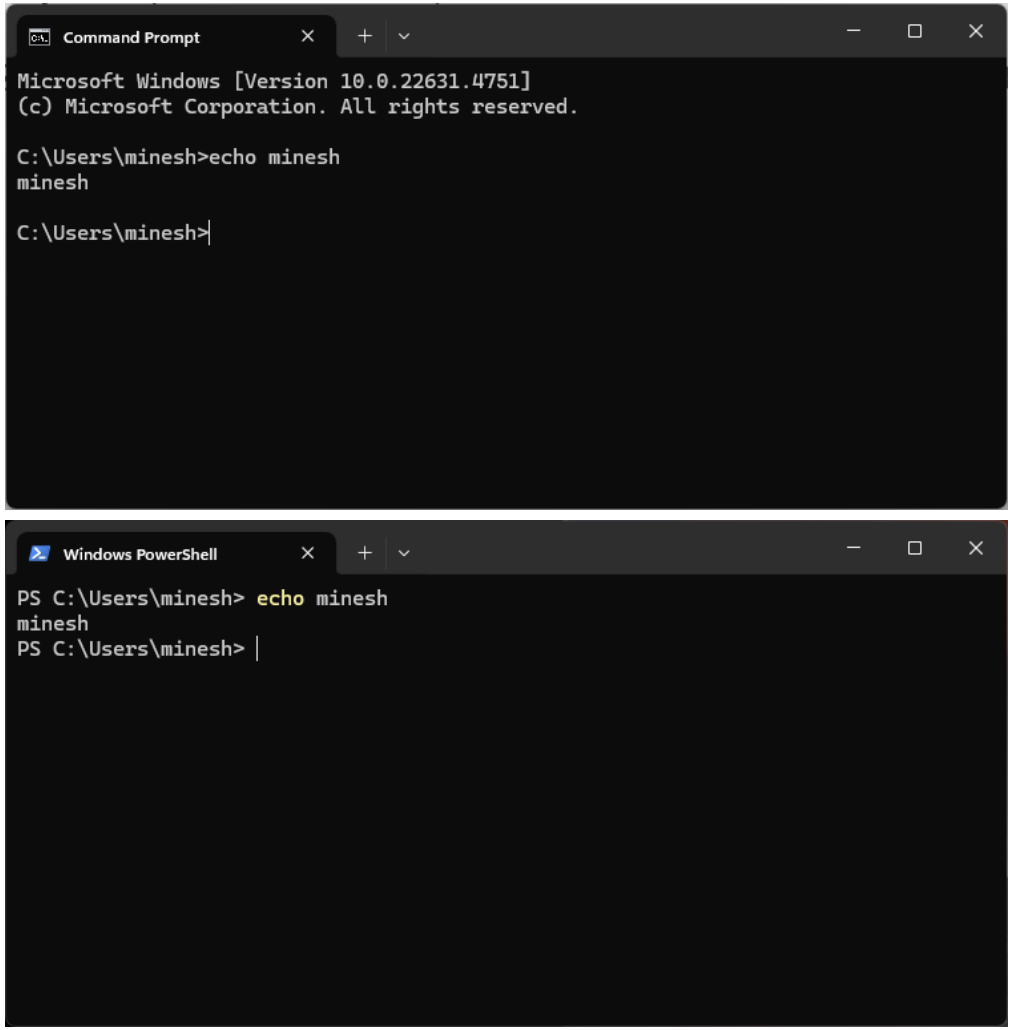


Using Linux with
a Command-Line Interface (CLI)

Compare: Ways of Using Windows 11



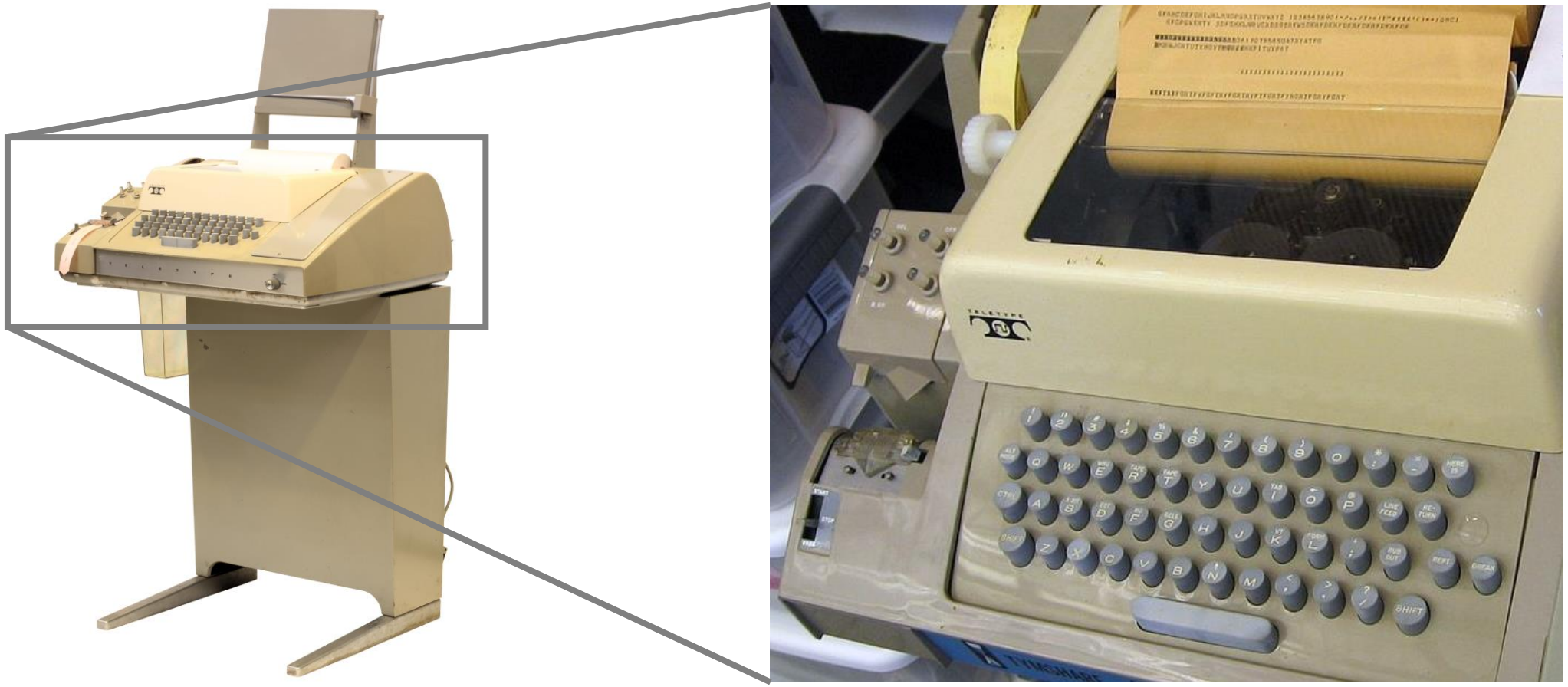
Using Windows 11 with
a Graphical User Interface (GUI)



Using Windows 11 with
a Command-Line Interface (CLI)

Linux Through the Command Line

- Many Linux applications were developed before graphics existed



Teletype Terminal (1963)

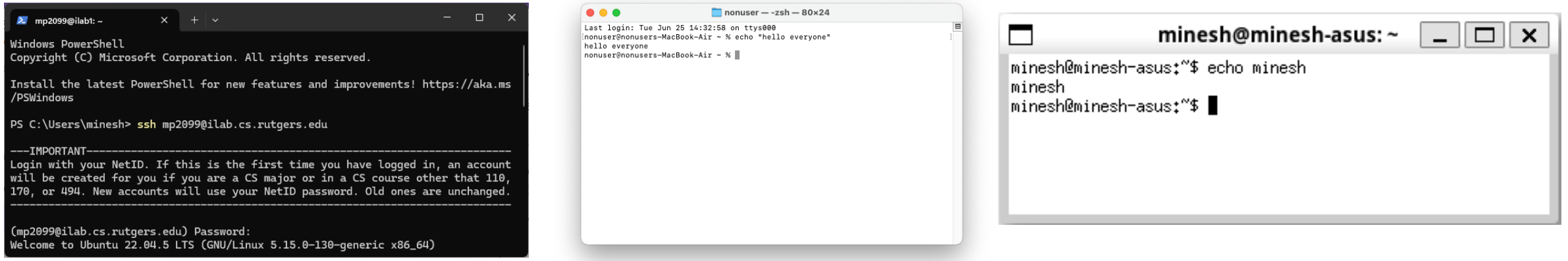
https://upload.wikimedia.org/wikipedia/commons/3/33/Teletype-IMG_7287.jpg

https://upload.wikimedia.org/wikipedia/commons/2/23/TTY33ASR.jpg

Programming in Linux

- Many Linux applications are **designed** to be used **text-only**

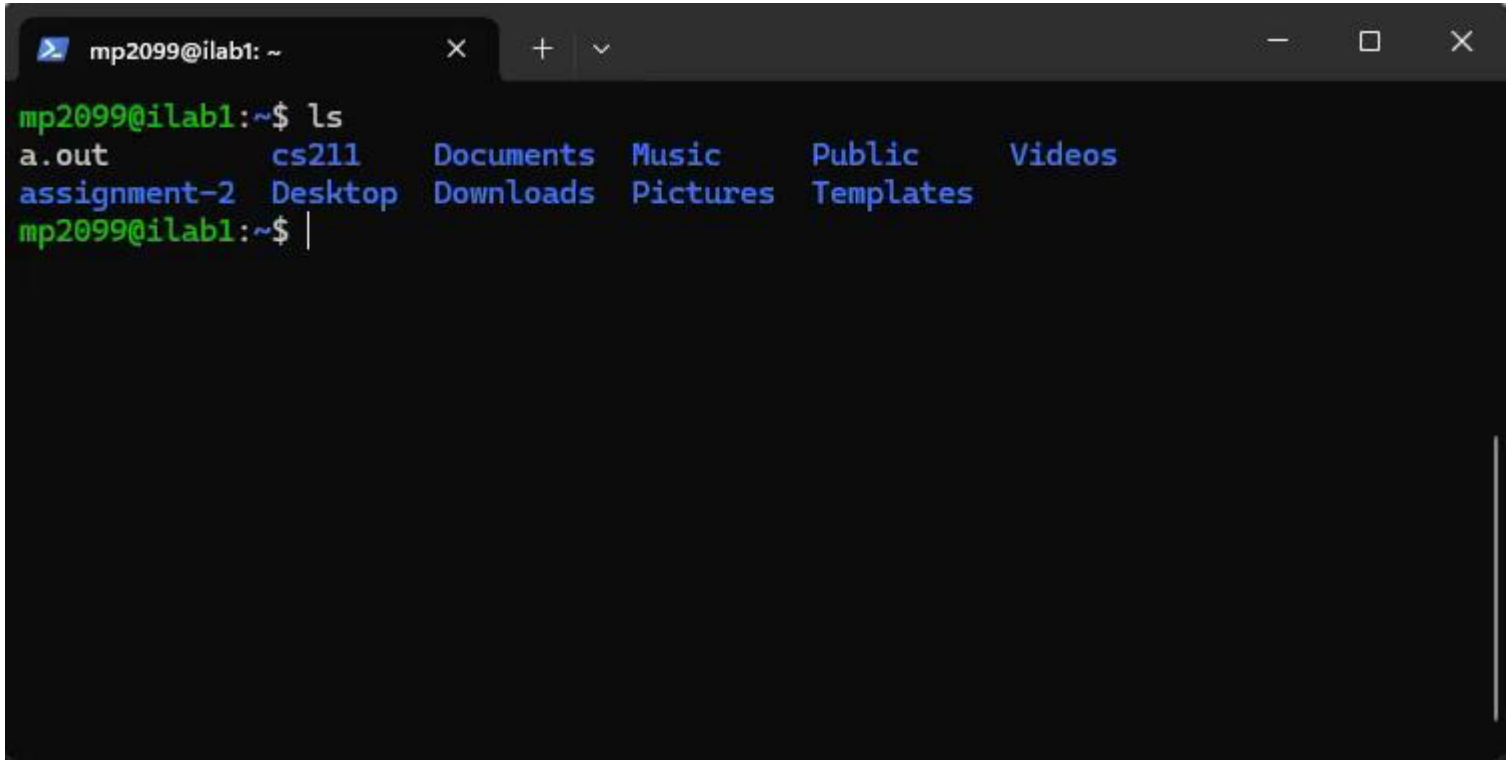
- You will need to use the command line in this class
  - We don't have Teletype Terminals, so you will use a **Terminal Emulator**
  - A program that emulates the functionality of an old Teletype Printer



Using native Terminal Emulators on Windows, Mac, and Linux (left-to-right).

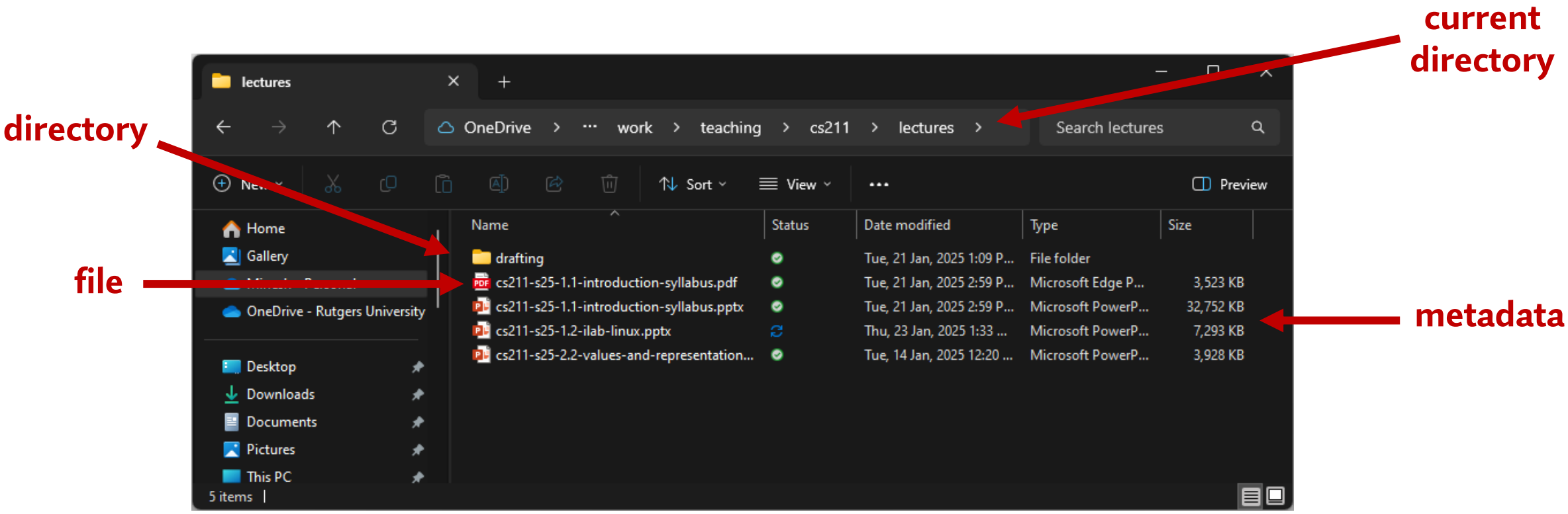
Using Terminal Emulators

- The “terminal prompt” waits for you to enter a command
  - E.g., “**clear**” cleans up the terminal
  - E.g., “**echo**” prints out whatever follows
  - E.g., “**gcc**” is the C compiler we will use in this class



Files and Directories: Windows GUI

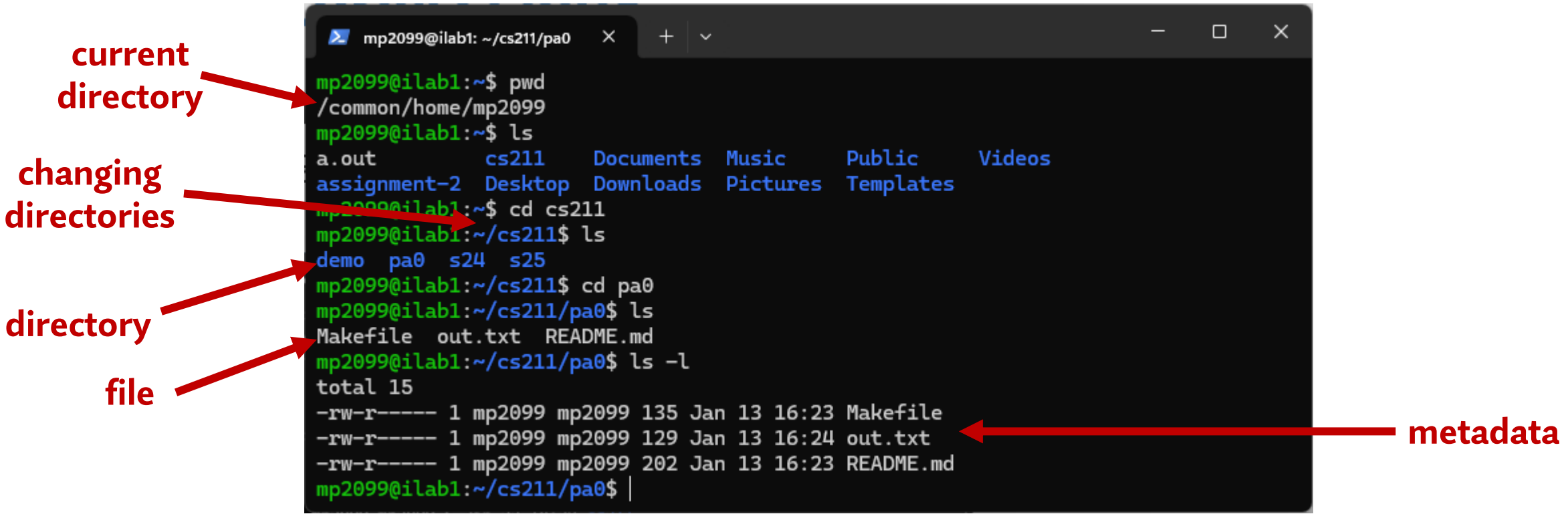
- Works roughly the same way on Windows, Mac, and Linux
  - "**Files**" contain data (e.g., text, image data, binary values)
  - "**Directories**" contain files or other directories



Windows Explorer

Files and Directories: Linux CLI

- Works roughly the same way on Windows, Mac, and Linux
  - "**Files**" contain data (e.g., text, image data, binary values)
  - "**Directories**" contain files or other directories

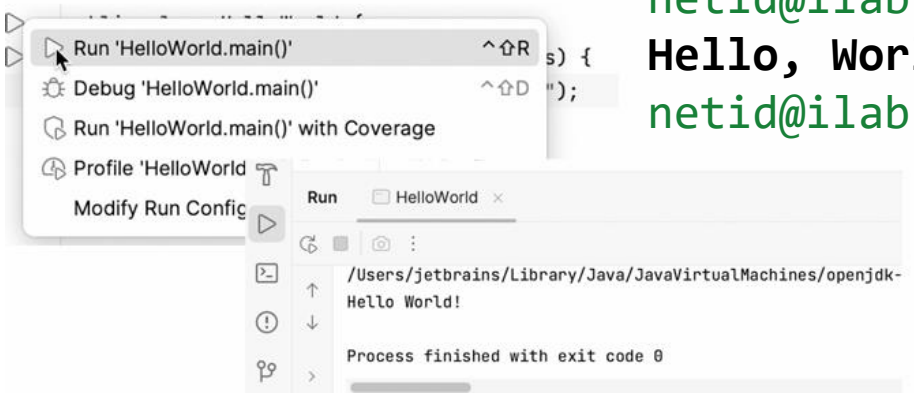


ilab (via Windows Terminal Emulator)

C vs. Java On the Command Line

Source file: "hello.java"

```
public class HelloWorld
{
     public static void main(String[] args)
     {
          System.out.println("Hello, World");
     }
}
```




```
netid@ilab:~$ javac hello.java
netid@ilab:~$ java hello.class
Hello, World
netid@ilab:~$
```

Source file: "hello.c"

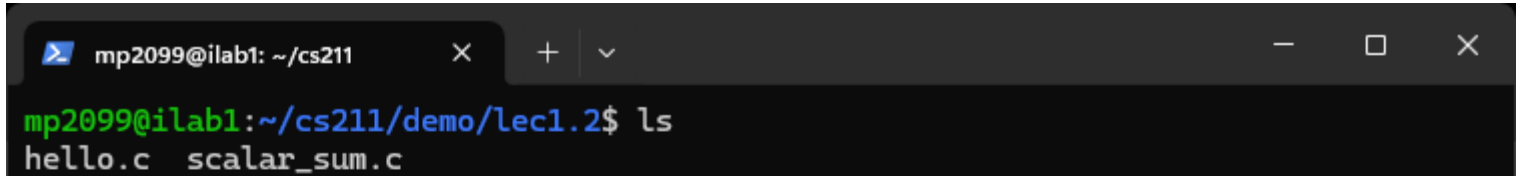
```
#include <stdio.h>

int main(int argc, char **argv)
{
     print("Hello, World");
     return EXIT_SUCCESS;
}
```

```
netid@ilab:~$ gcc -o hello hello.c
netid@ilab:~$ ./hello world
Hello, World
netid@ilab:~$
```

https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html#package

Running Applications in Linux

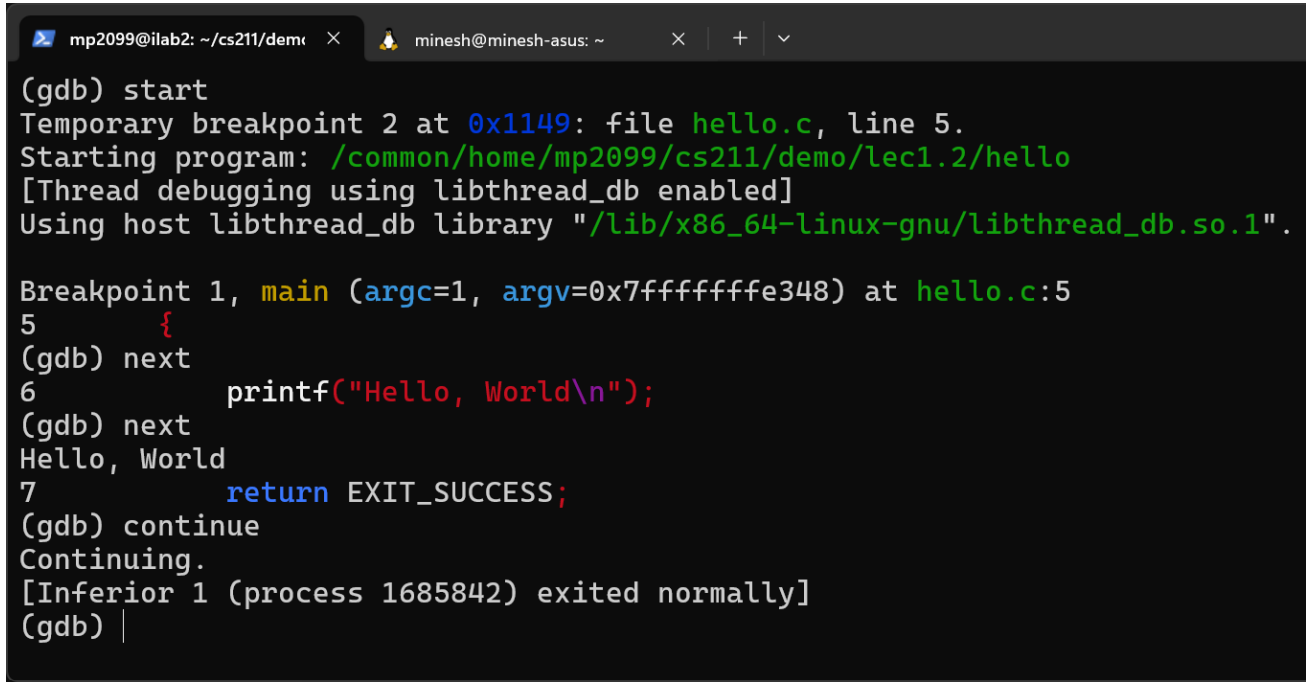
- Applications are run by issuing the full path to the executable file
  - **Relative paths** are given starting with **“./”**
    - **“.”** = current dir
    - **“..”** = parent dir
  - **Absolute paths** simply start with **“/”** (the “root” directory)



Debugging C Applications in Linux

- **Method 1:** print out variables (`printf()`)
  - Tried-and-tested method
- **Method 2:** GNU Debugger (GDB)
  - Step through code, set breakpoints, print values, evaluate expressions, and more!

```
mp2099@ilab2:~/cs211/demo/lec1.2$ gcc hello.c -g -o hello
mp2099@ilab2:~/cs211/demo/lec1.2$ gdb hello
```



Agenda

- **More Logistics**
- **Part 1:** C programming
- **Part 2:** Linux
- **Part 3:** iLab Demo

Rutgers CS Instructional Lab

- Hill 248
- Hill 252
- Hill 254

4 x

https://dlcdnwebimgs.asus.com/gain/d47ac467-7878-47b7-a29a-c9adfb80b9bd/

▼ Specifications

rLab{1,3,4}.cs: OS: Ubuntu 22.04 LTS | **CPU:**2 x AMD EPYC 7413 24-Core Processor @ 2.7GHz - 96 vCores
Memory: [rLab1.cs: 1.5TB] [rLab3.cs: 1TB] [rLab4.cs: 2TB]
Make/Model:ASUSTeK COMPUTER INC. ESC8000A-E11 | April 2022 | **CPUMark:** 74417 | **Storage:** 7TB SSD
rLab1.cs GPU Specs: 3 x GeForce RTX A4500 Ada | **CUDA Core:** 3 x 3840 **Memory:** 3 x 24GB| **Cuda capability:** 8.9 | **Version**: 12.2
rLab3 and rLab4 GPU Specs:8 x RTX A4000 | **Memory:** 8 x (16 GB, 6144 CUDA cores, 19 Tensor Cores, 48 RT Cores) | **CUDA capability** : 8.6 | **Version**:11.3

---

rLab2.cs: OS: Ubuntu 22.04 LTS | **CPU:**2 x AMD EPYC 9224 24-Core Processor @ 2.7GHz - 96 vCores
Memory: 1TB **Make/Model:**ASUSTeK COMPUTER INC. ESC8000A-E12 | Aug 2023 | **CPUMark:** 93314 | **Storage:** 2 x 7TB SSD
rlab2.cs GPU Specs:8 x RTX A4000 | **Memory:** 8 x (16 GB, 6144 CUDA cores, 19 Tensor Cores, 48 RT Cores) | **CUDA capability** : 8.6 | **Version**:12.2

https://report.cs.rutgers.edu/nagiosnotes/iLab-machines.html

Three Ways to Connect

- The ilab webpage discusses ways to connect in detail

- We recommend three ways:

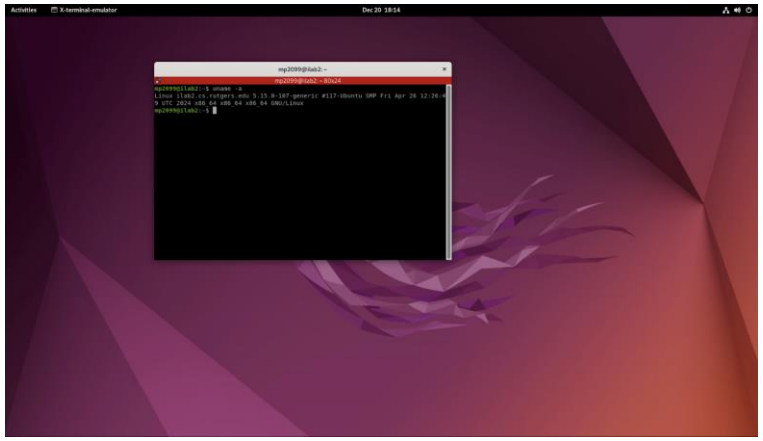
1. **Command Line:** SSH
2. **Visual Studio Code** + SSH Extension
3. **Graphical User Interface:** WebLogin

Programming Assignment 1

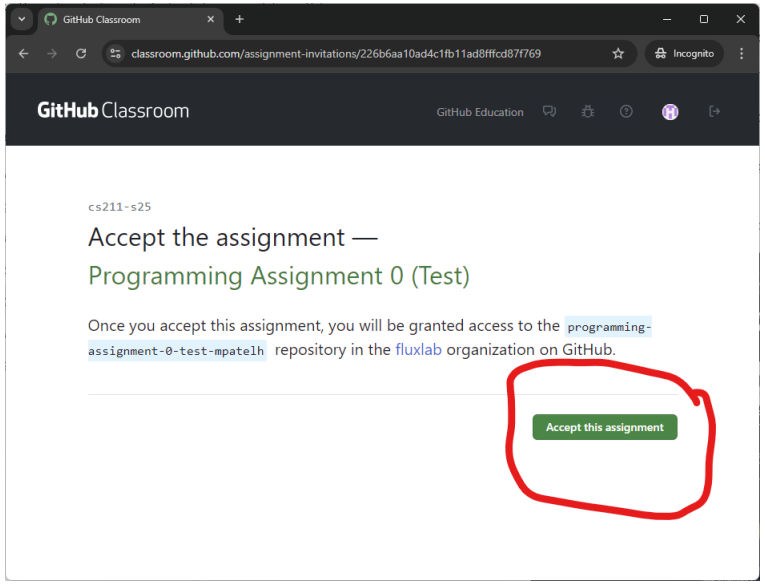
- No programming; just the submission process

1. **Log in to iLab with a command line interface**
   - Set up your CS account
2. **Get the code through GitHub Classroom**
   - Set up a GitHub account
3. **Run the code**
   - Trivial: run “Make” on the command line
4. **Upload the generated output file using Git**
5. **Submit using GradeScope**
   - Make sure you can access the course’s GradeScope

1

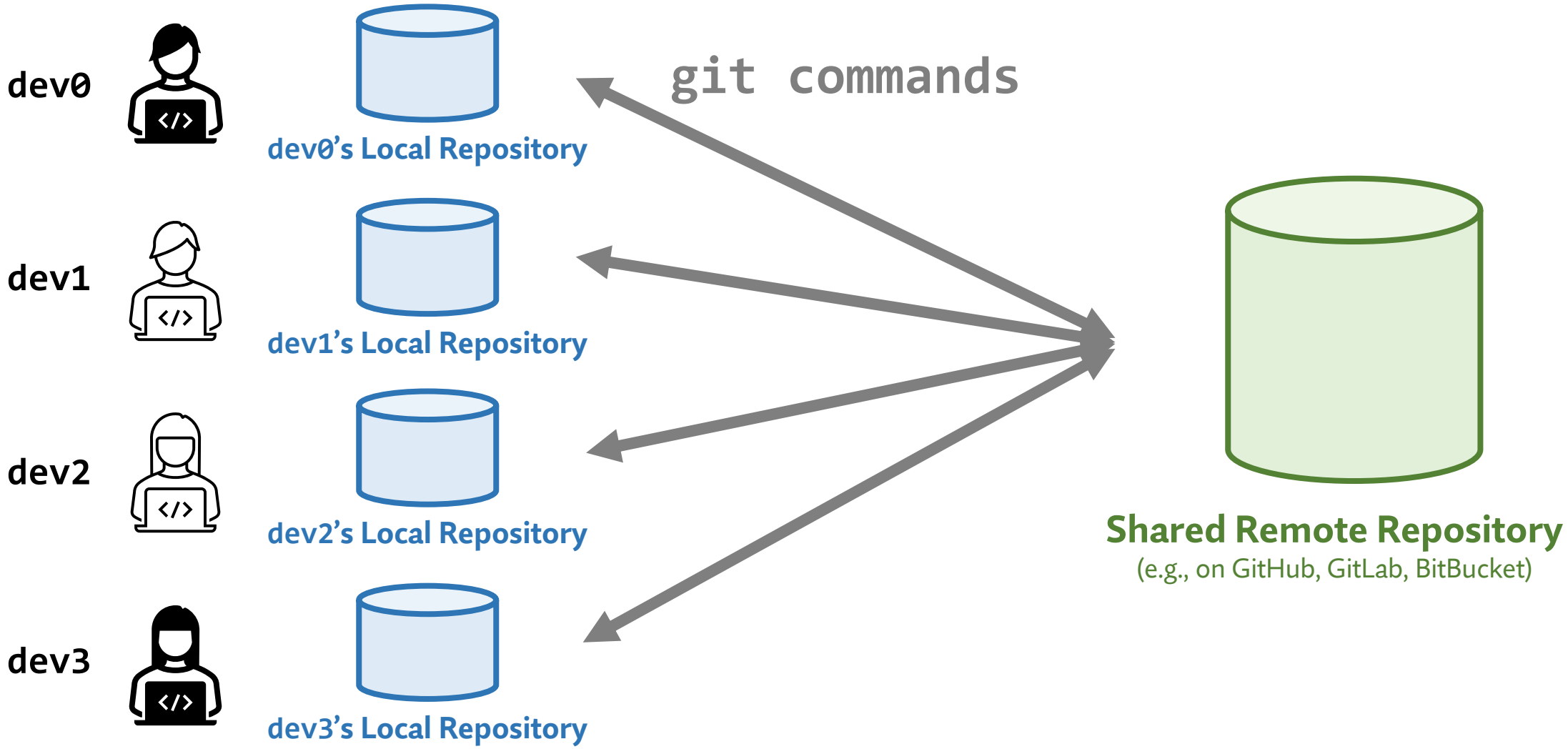


2



Git

- Git syncs files between “local” and shared “remote” **repositories**



Using Git

- **Widely used** throughout hardware/software engineering
  - Enables very clean sharing between multiple contributors
  - Keeps track of file histories
  - … and much, much more

- I will not cover Git in class – PA1 explains the necessary basics
  - There are many great resources out there on using Git
    - E.g., Git documentation
    - E.g., Youtube: How Git Works: Explained in 4 Minutes

"PA0" Demo

- I will demo:
  1. **Log in to iLab with a command line interface**
     - From Windows 11 (PowerShell)
     - From a Linux virtual machine on my laptop (WSL)
     - Using WebLogin
     - Using Visual Studio Code + SSH Extension
  2. **Get the code through GitHub Classroom**
     - PAo GitHub Classroom link
  3. **Run the code**
     - Trivial: run "Make" on the command line
  4. **Upload the generated output file using Git**
  5. **Submit using GradeScope**

Survey #1 is Out!

- Please help us help you ☺

Survey #1 Link

CS 211: Intro to Computer Architecture

1.2: C, Linux, and ilab Demo

Minesh Patel

Spring 2025 – Thursday 23 January