INTRODUCTION TO SOFTWARE DEVELOPMENT

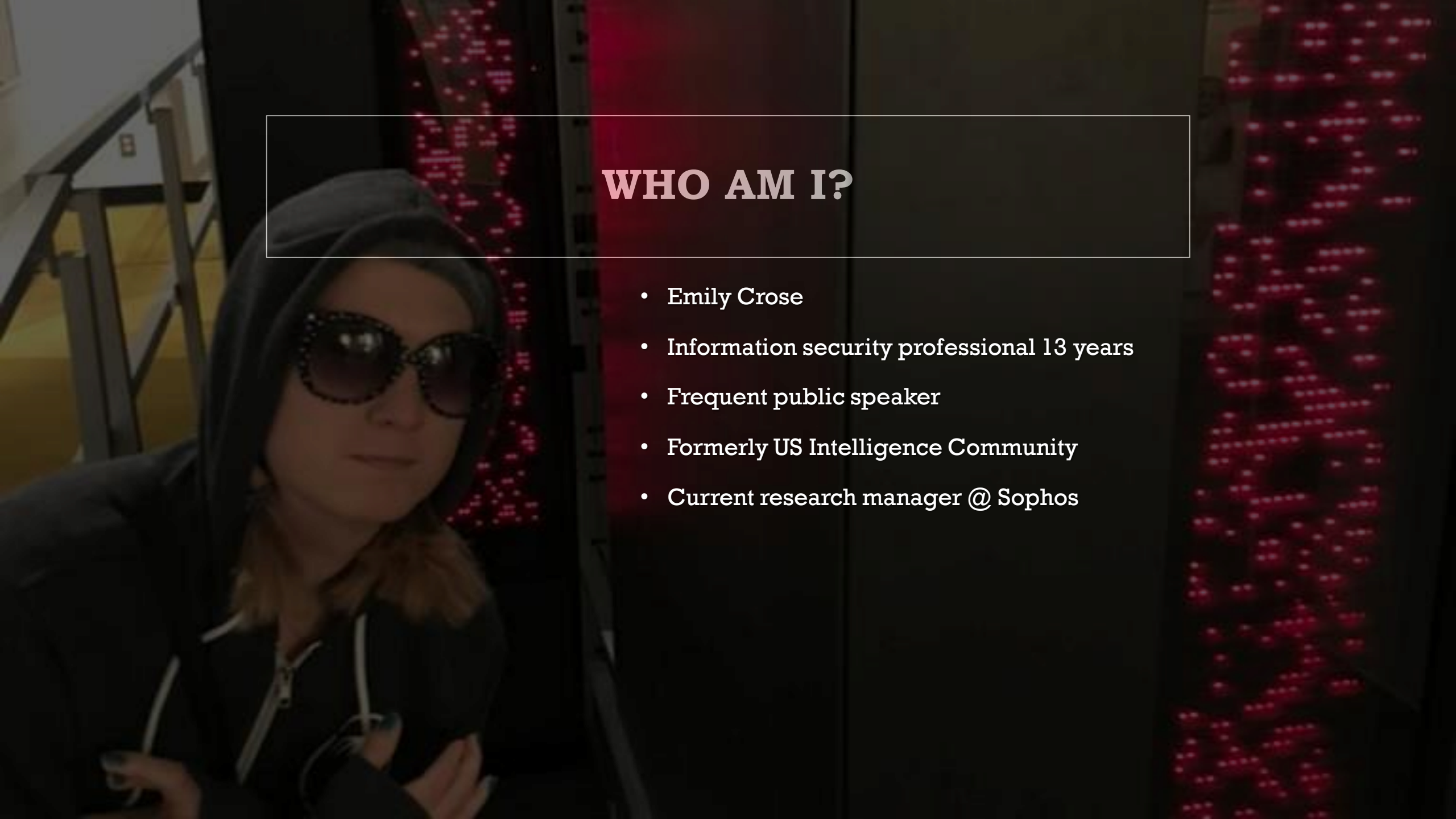
Emily Crose

For

Oakland University

FOR TODAY

- Introductions
- Tools for the class
- Overview of class material
- Hardware & Software introduction

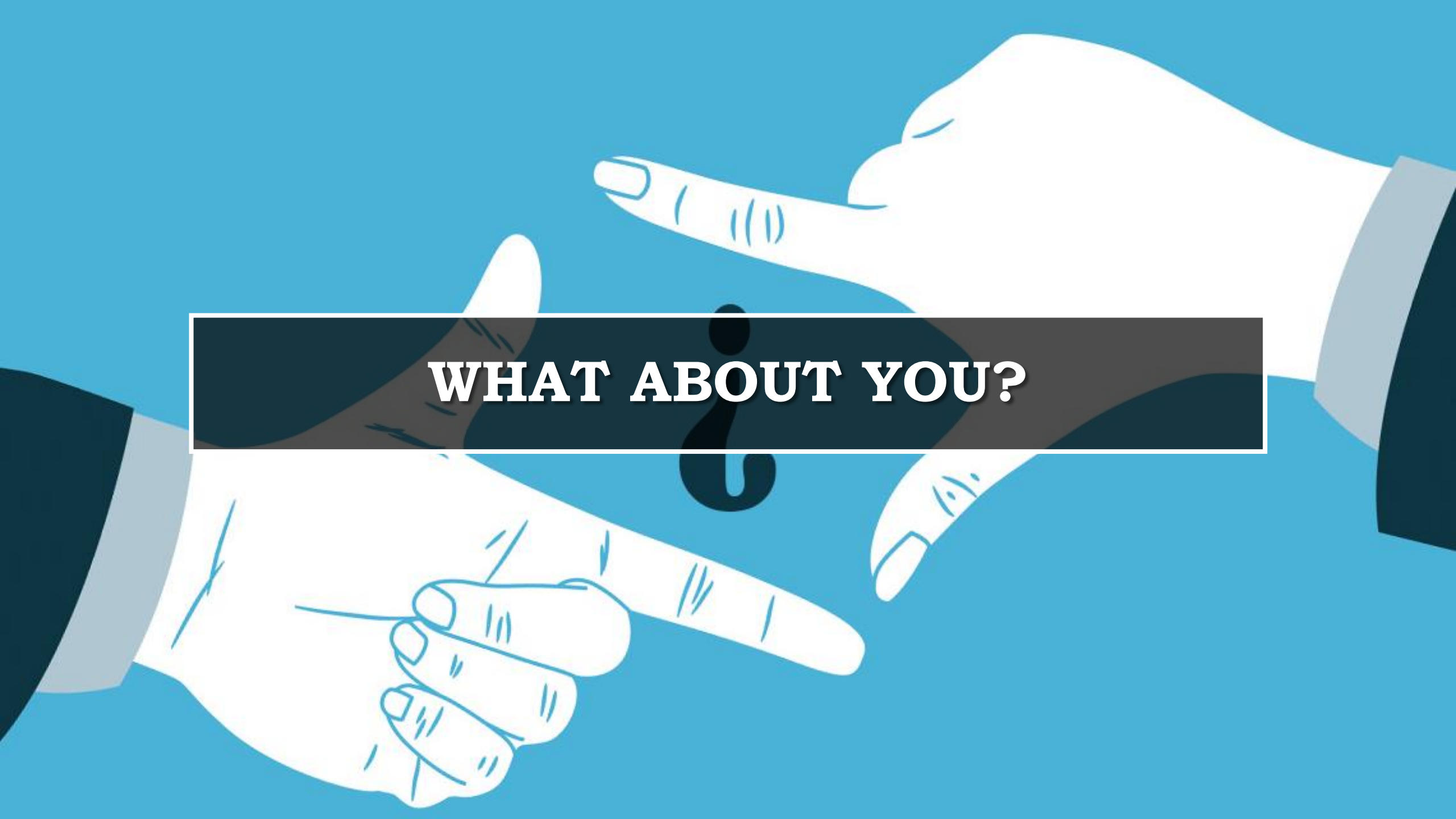


WHO AM I?

- Emily Crose
- Information security professional 13 years
- Frequent public speaker
- Formerly US Intelligence Community
- Current research manager @ Sophos

CONTACT INFO

ecrose@oakland.edu


WHAT ABOUT YOU?

WHEN WE MEET

- Meet 4 times/week
- 3 hours/day
- 4 weeks
- Mondays, Tuesdays, Wednesdays & Thursdays
    - 7AM Eastern – 10AM Eastern

TOOLS WE WILL USE

- Moodle
  - Class materials
- Github
  - Code repositories
  - Change management
- Jira
  - ticketing
- Confluence
  - For wikis
- More?

COURSE OVERVIEW

What will we be talking about in this course?

What do you want to learn?

MATERIAL OVERVIEW

Understanding modern software

How it works

How it is maintained

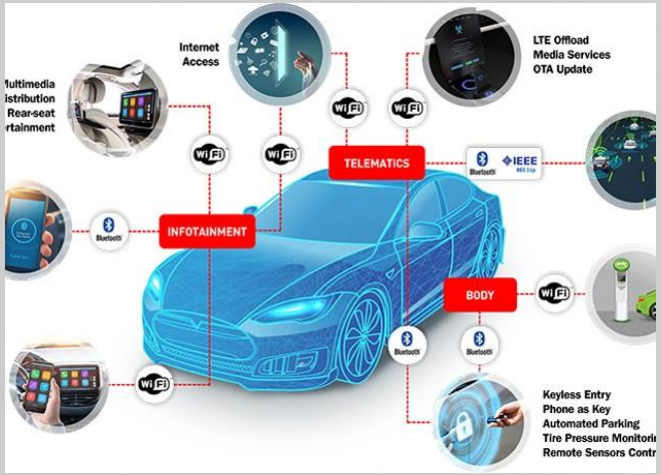
The Software Development Life Cycle (SDLC)

Tools for managing software development

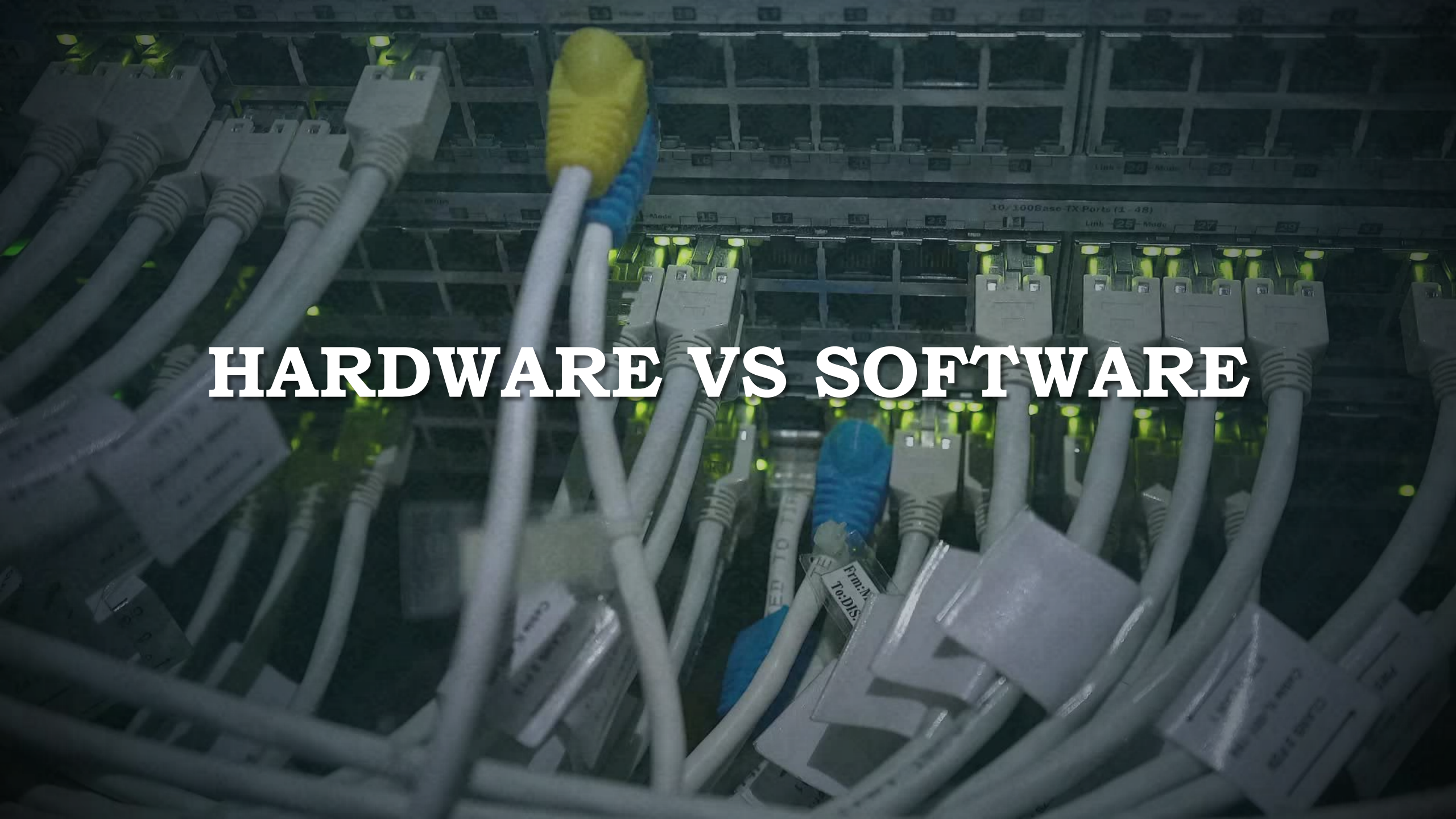
Management techniques for maintaining codebases

Standards that apply to these topics

SOFTWARE WE CAN SEE



HIDDEN SOFTWARE

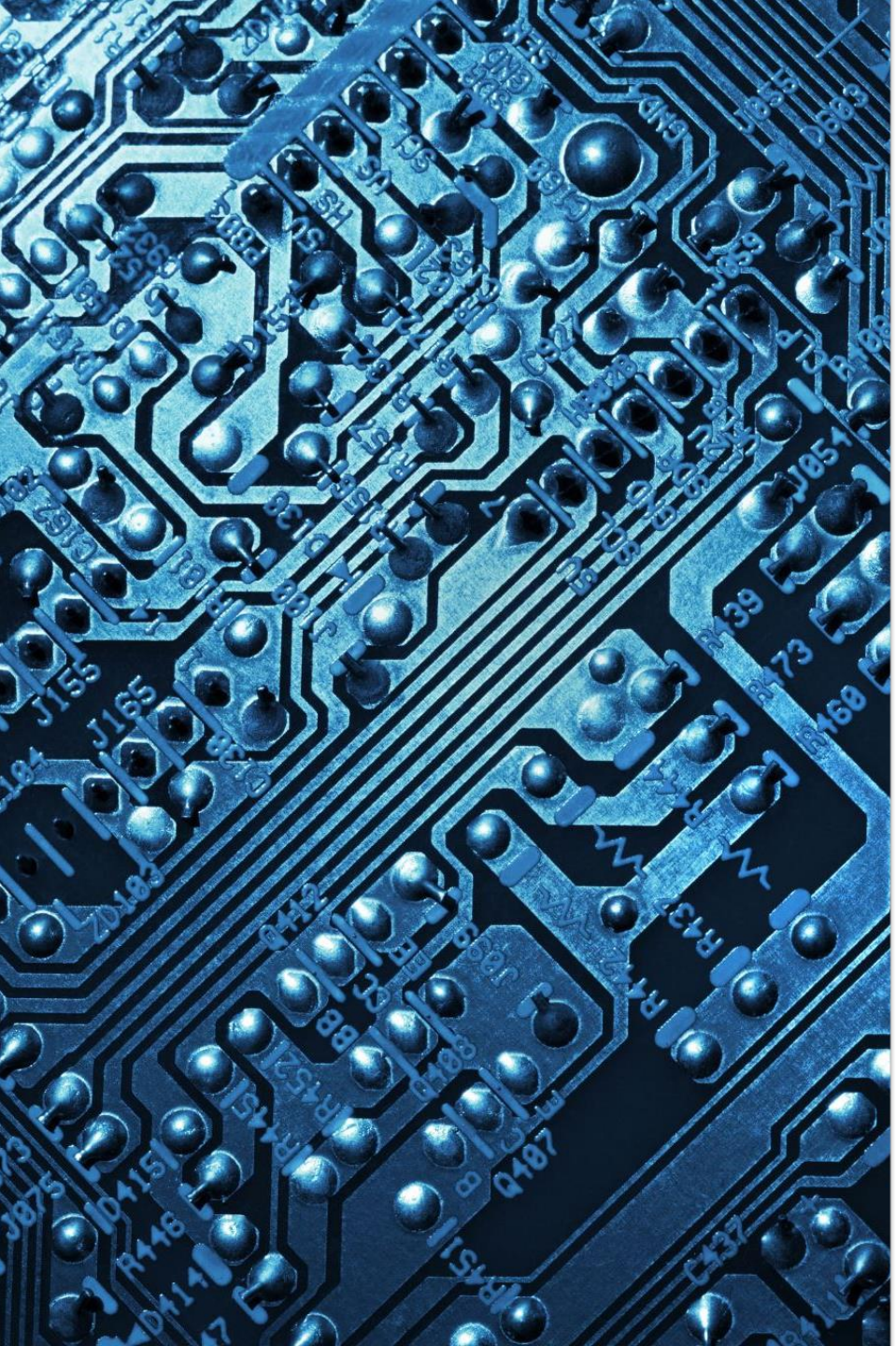

HARDWARE VS SOFTWARE

HARDWARE COMPONENTS

- Central Processing Unit (CPU)
- Graphics Processing Unit (GPU)
- Random Access Memory (RAM)
- I/O devices
  - Keyboard
  - Mouse
  - VR Headset

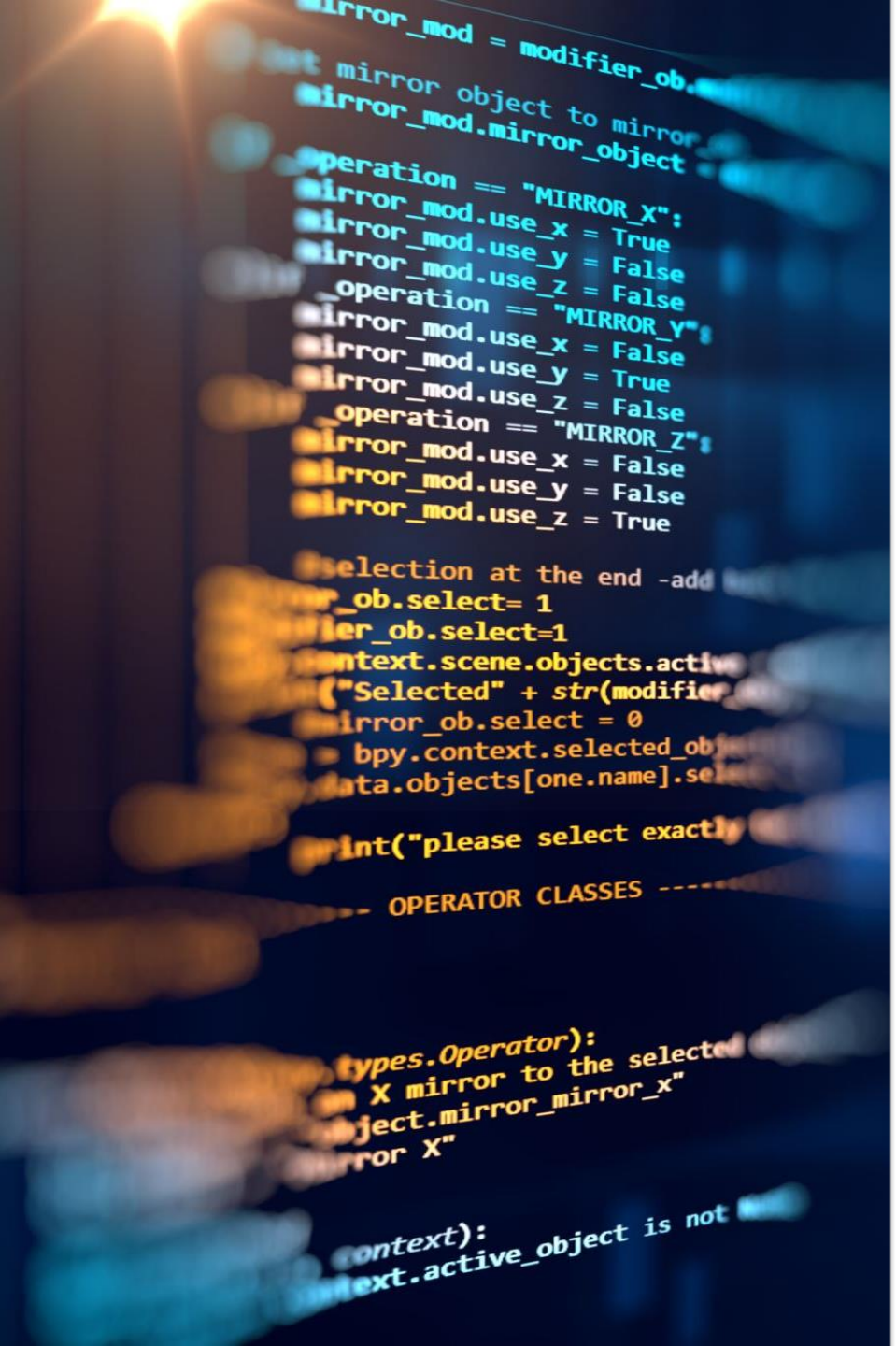
HARDWARE COMPONENTS CONT’

- Sensors
  - Temperature
  - Pressure
  - gyroscope



PURPOSE OF HARDWARE

- The physical half of computing (As opposed to logical)
- Does physical calculations
- Produces signals which are interpreted by software



PURPOSE OF SOFTWARE

- Provides interface between hardware and the user
- Software provides a “logical” layer of interpretation


MODERN SOFTWARE

OLD THINGS

- Old languages
  - Javascript
  - C
  - C++
  - Assembly
- Desktop software
- Mainframes?

OPERATING SYSTEMS

Windows

- 10
- 11
- Embedded

Mac OS

iOS

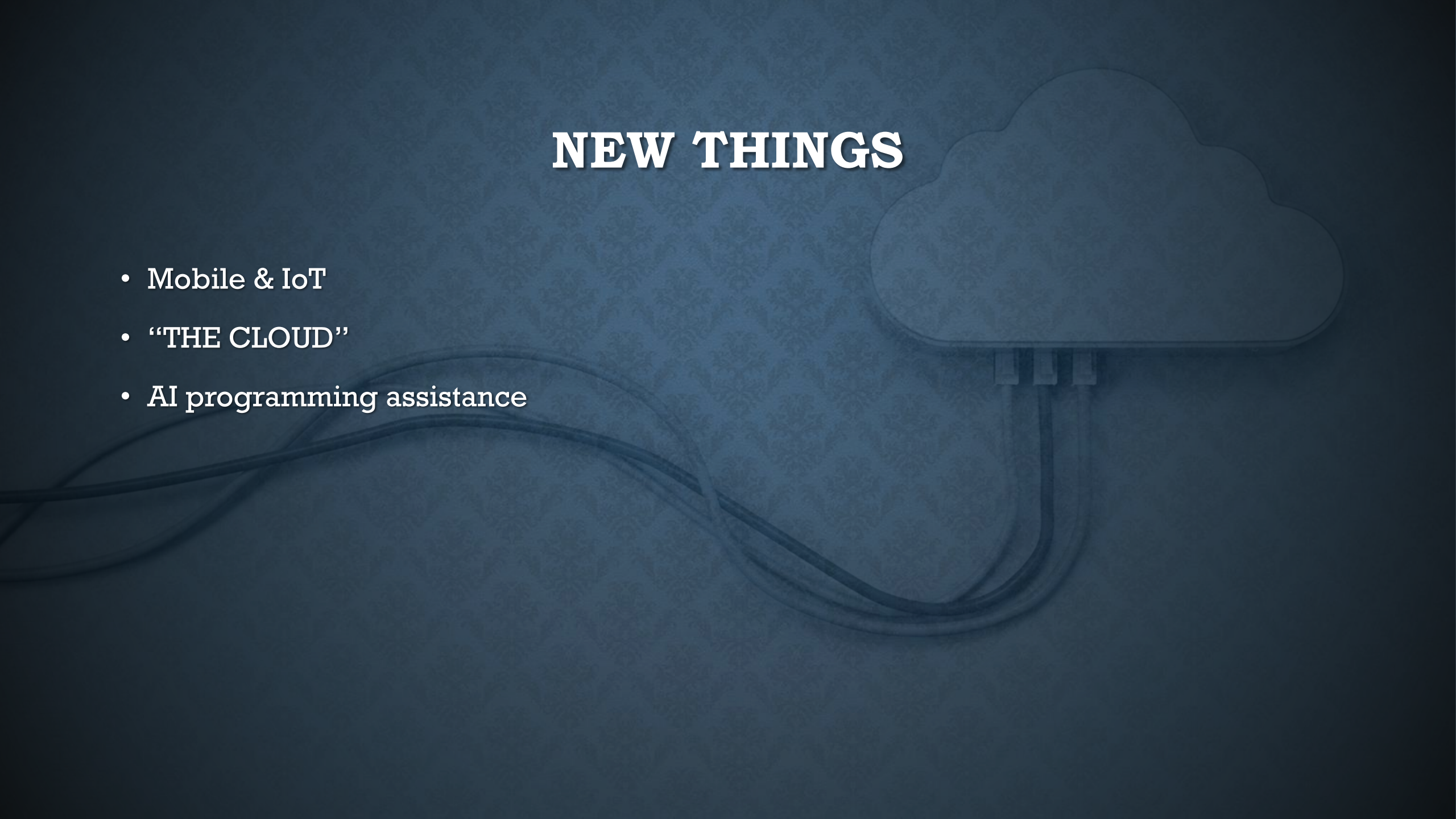
Linux

- Various flavors


10 MINUTE BREAK!

NEW THINGS

- Mobile & IoT
- “THE CLOUD”
- AI programming assistance




WHAT IS THE CLOUD?
Have we seen this before?

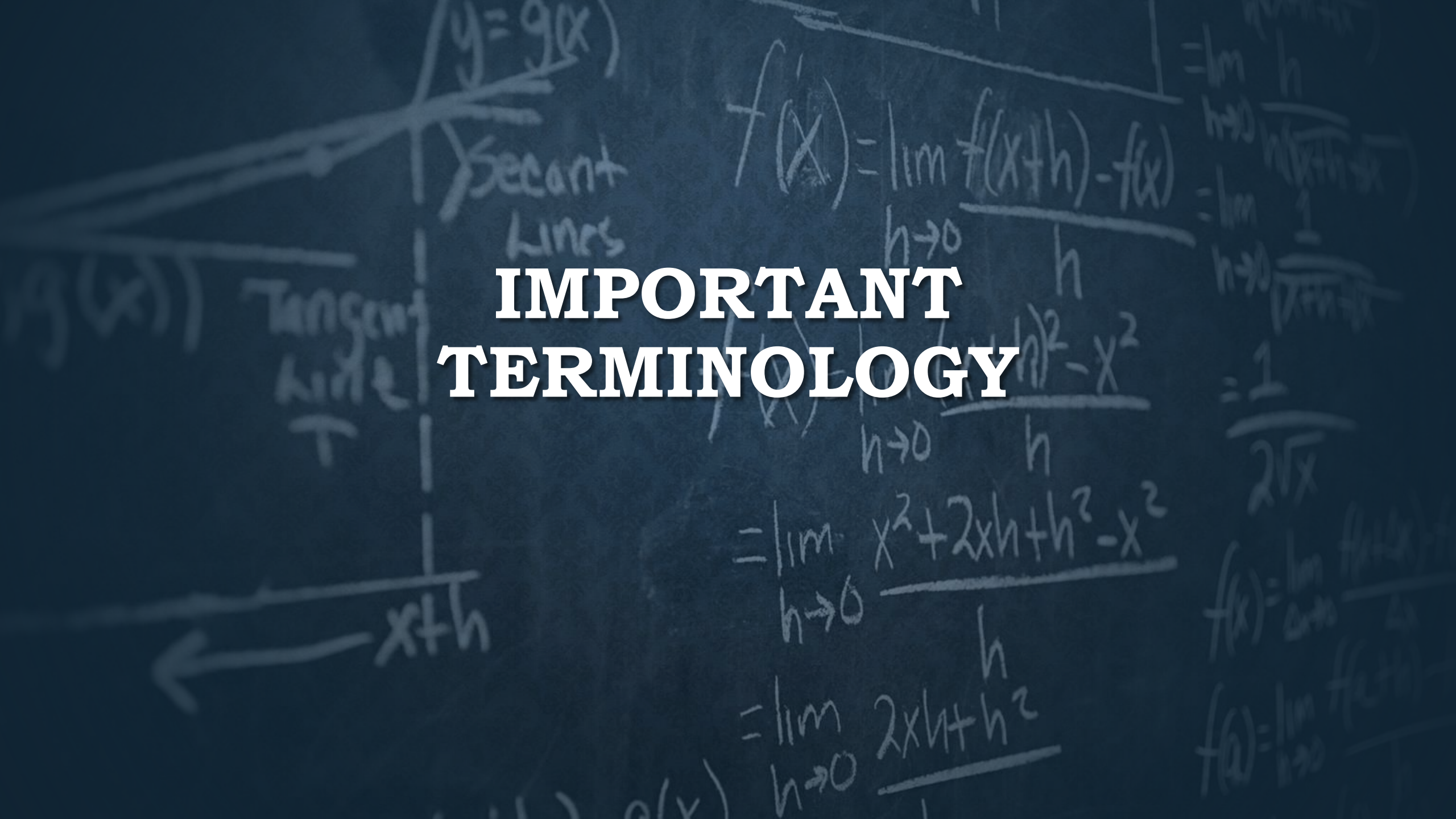
POPULAR CLOUD VENDORS

- Azure (Microsoft)
- Amazon Web Services (AWS)
- Google cloud computing
- Resellers

WHAT IS MOBILE/IOT?

FACTORY TECH

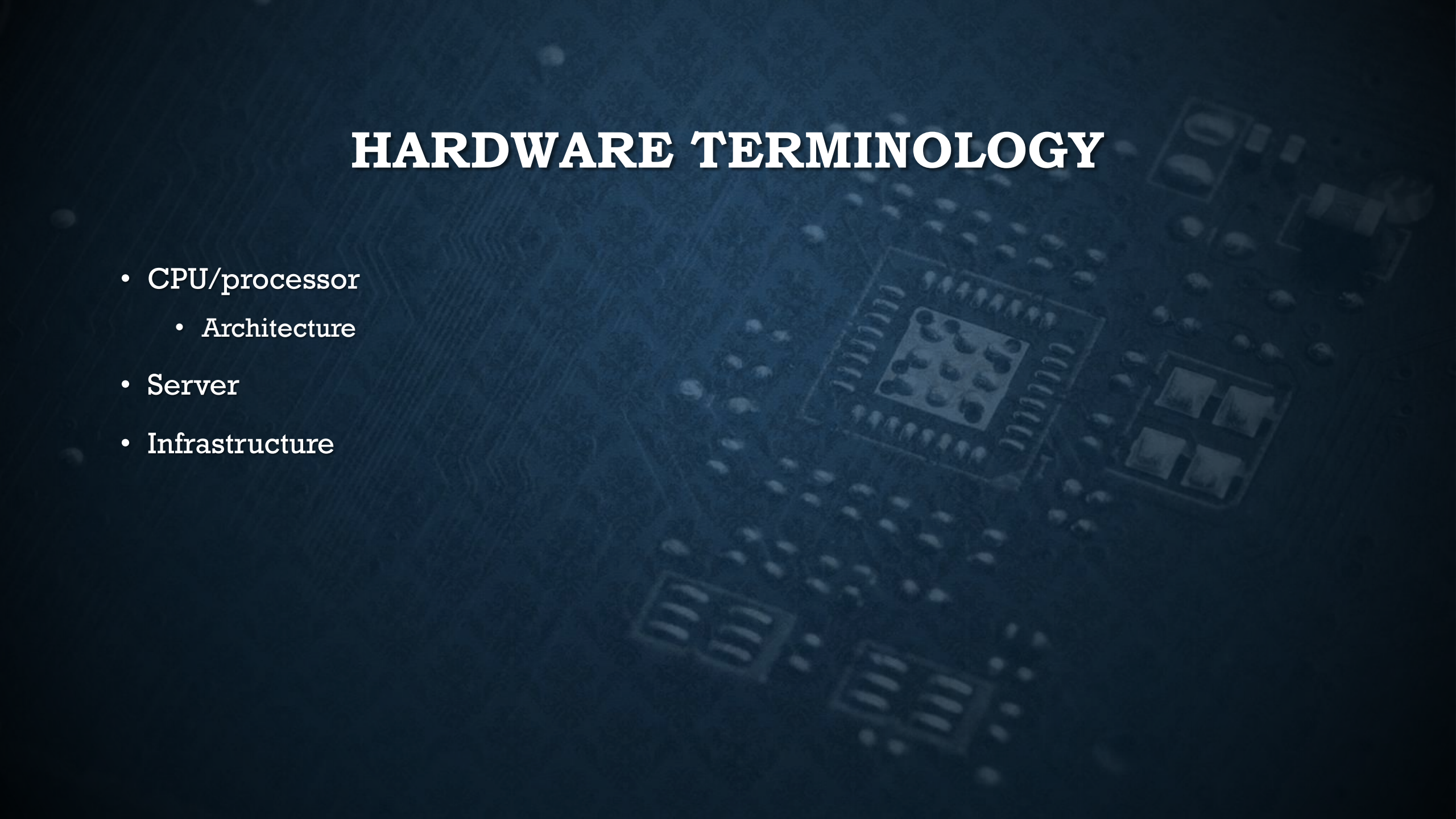
hardware + software!



IMPORTANT TERMINOLOGY

HARDWARE TERMINOLOGY

- CPU/processor
  - Architecture
- Server
- Infrastructure



SOFTWARE TERMINOLOGY

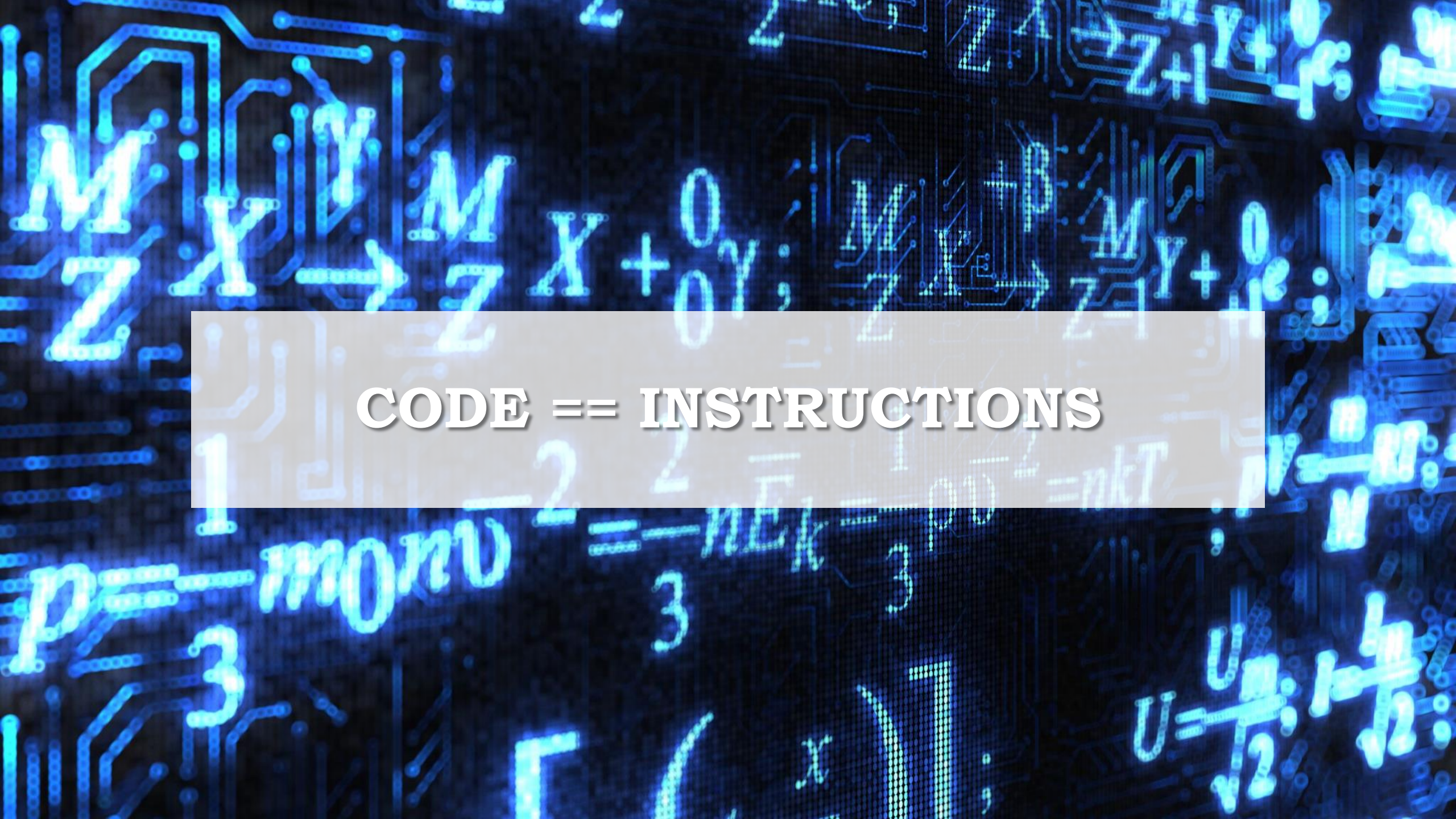
- Programming/coding
- Updates vs. Upgrades
- Code commenting
- Parsing
- Execution
- Environment
- Application Programming Interface (API)

SECURITY TERMINOLOGY

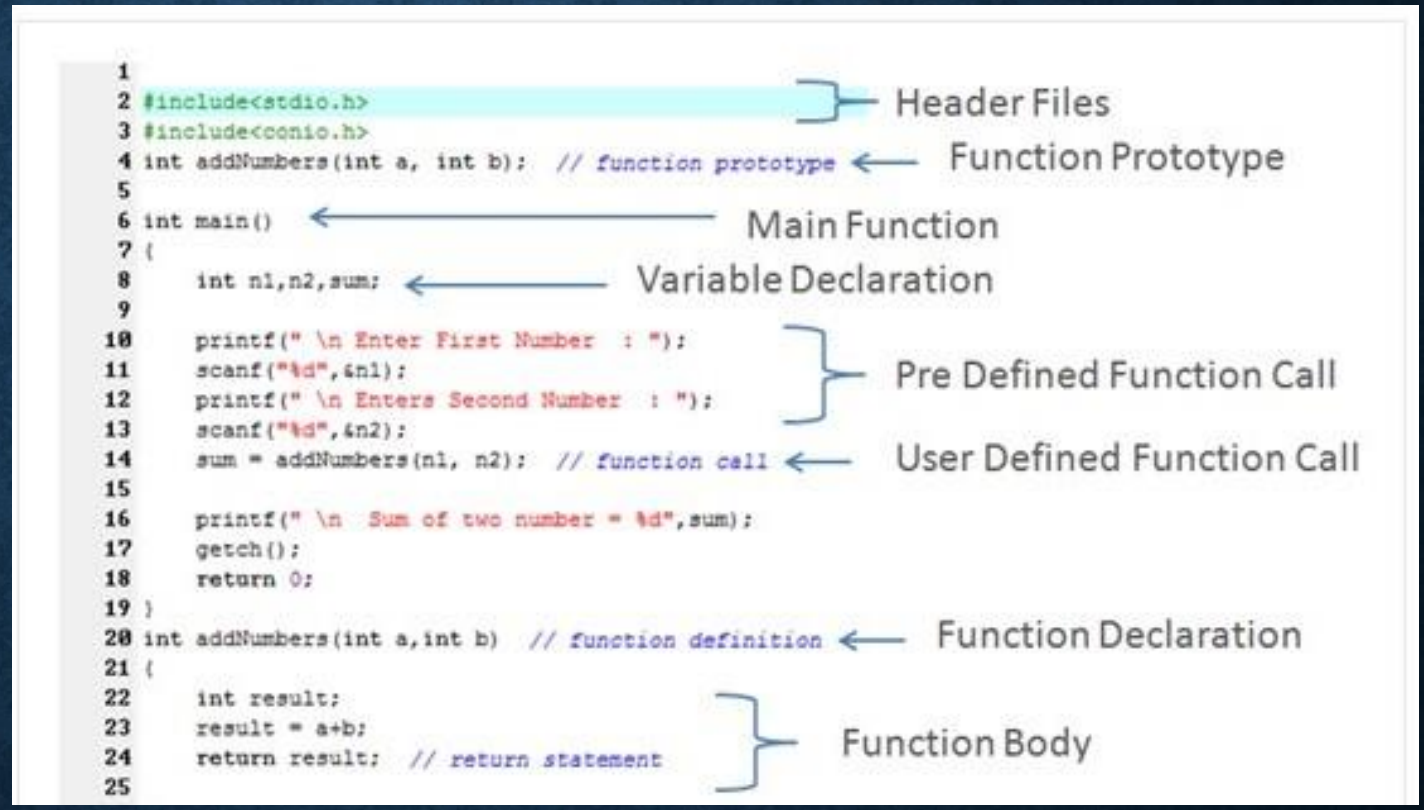
- Vulnerability
- Exploit
- Payload

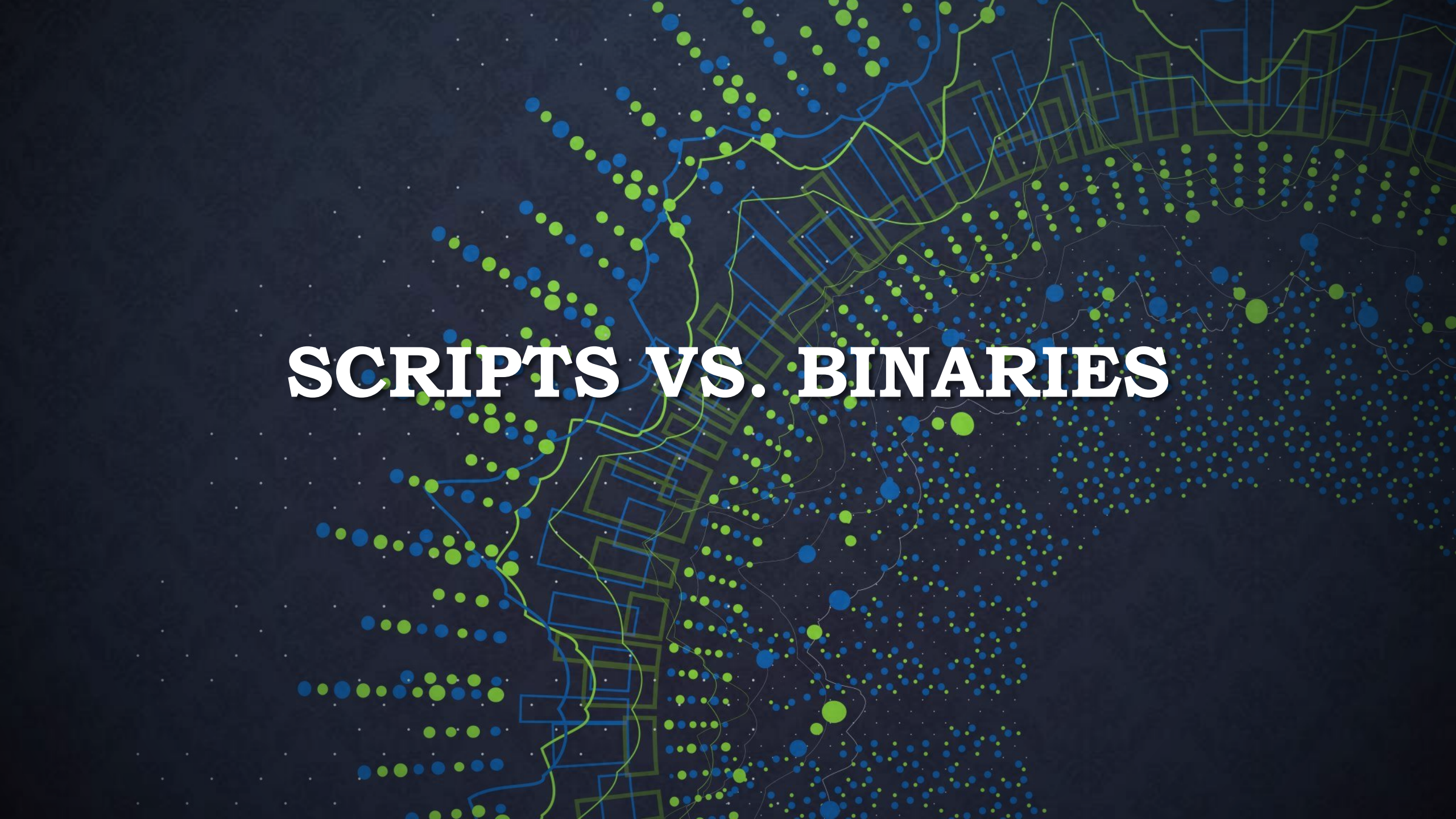
WHAT IS A PROGRAM *REALLY*?


PROGRAM == CODE


CODE == INSTRUCTIONS

PARTS OF A PROGRAM




SCRIPTS VS. BINARIES

COMPILING

What is compiling?

A set of instructions that has been compiled becomes an executable ‘binary’

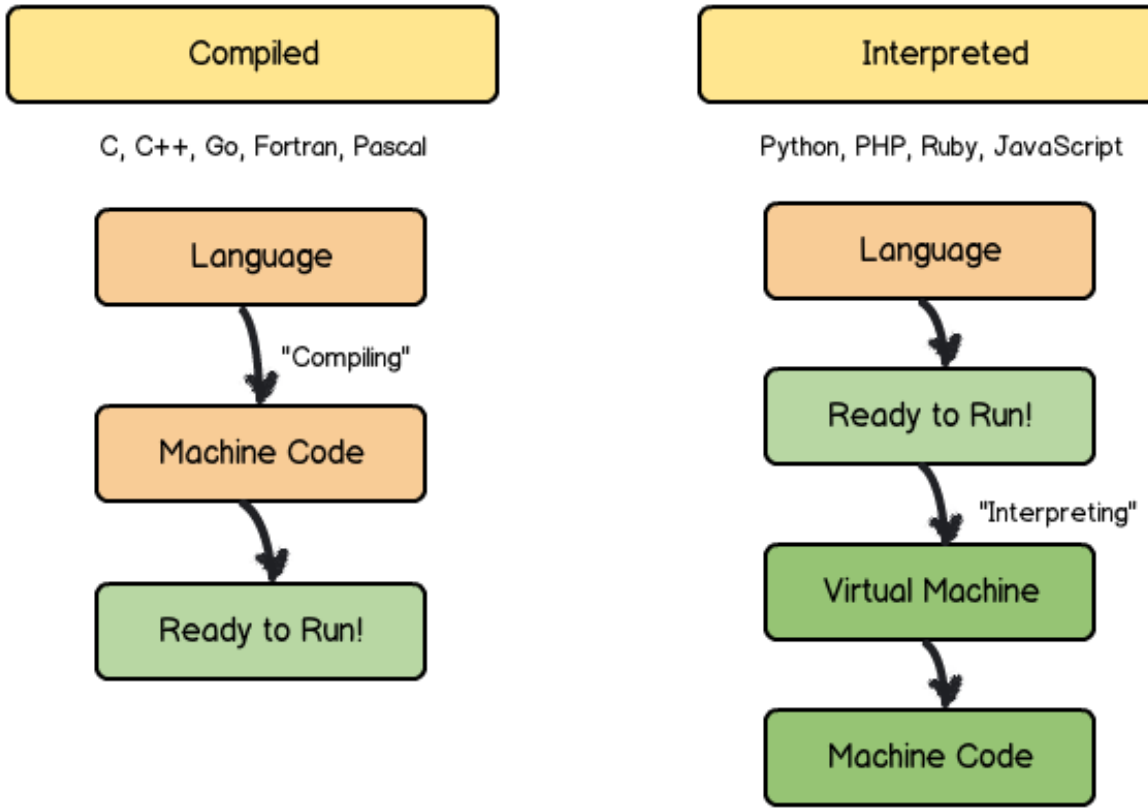
We may just call this an ‘executable’

Scripts do not compile!

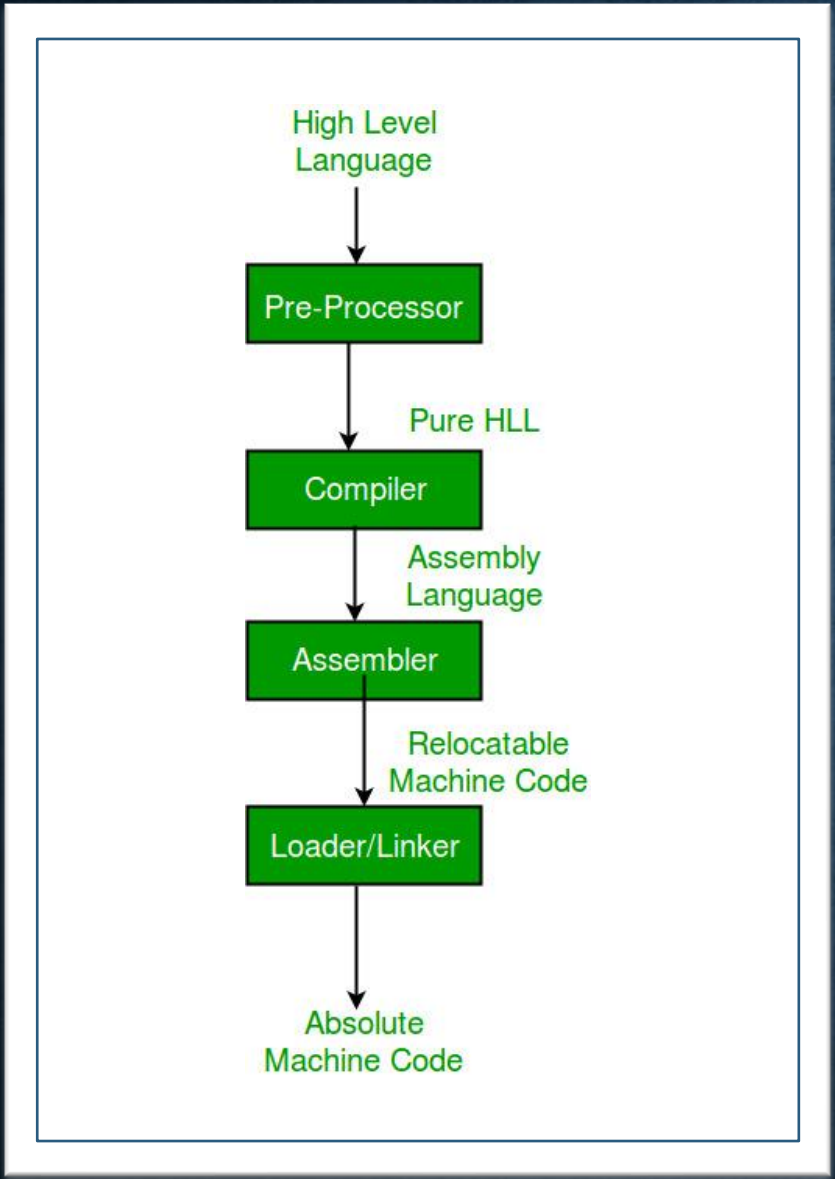
Compiled Language
VS
Interpreted Language

Comparison Chart

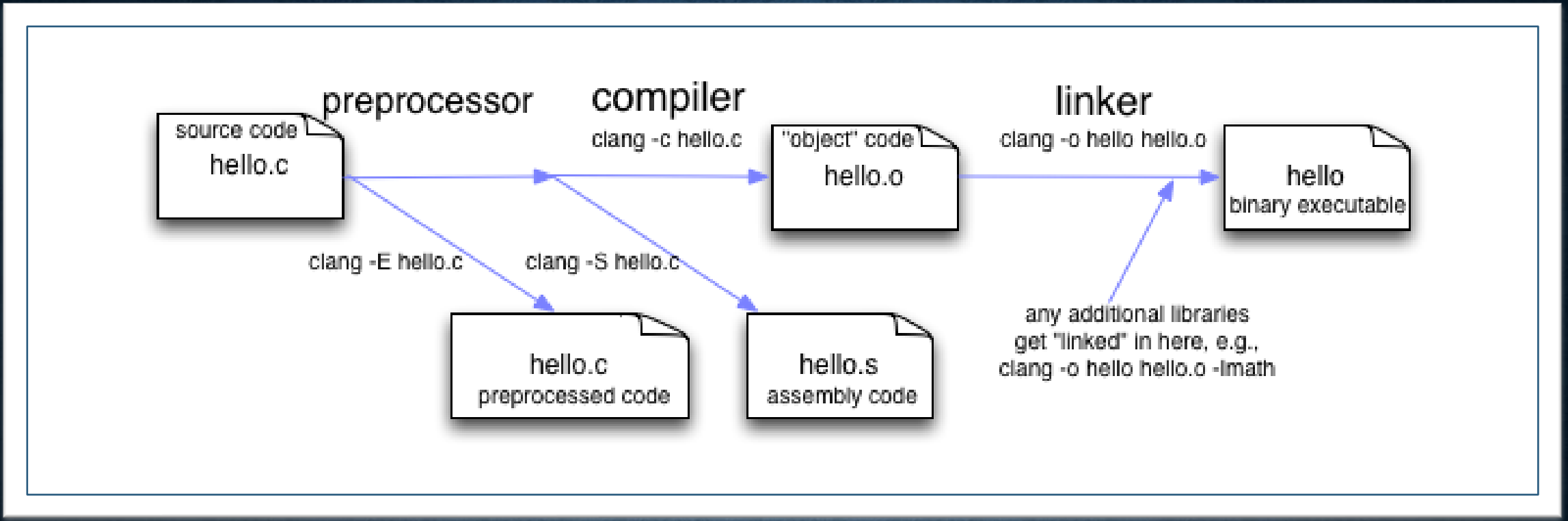
Compiled Language	Interpreted Language
The code of compiled languages can be executed directly by the computer's CPU.	A program written in an interpreted language is not compiled, it is interpreted.
The source code must be transformed into machine readable instructions prior to execution.	It does not compile the source code into machine language prior to running the program.
Compiled programs run faster than interpreted programs.	Interpreted programs can be modified while the program is running.
Delivers better performance.	Delivers relatively slower performance.
C, Fortran, and COBOL are languages used to produce compiled programs.	Java and C# are compiled into bytecode, the virtual interpreted language.



COMPILED VS INTERPRETED SIMPLIFIED



COMPILE PROCESS



COMPILING C IN PRACTICE


10 MINUTE BREAK!

SOFTWARE SNAFUS

- What can go wrong if we don't get software right?
  - Program crashes
  - System stability compromised
  - Exploitation
    - Network breach
    - Theft of IP/customer data
    - Loss of reputation
  - Risk of injury?


ISAT GeoStar 45

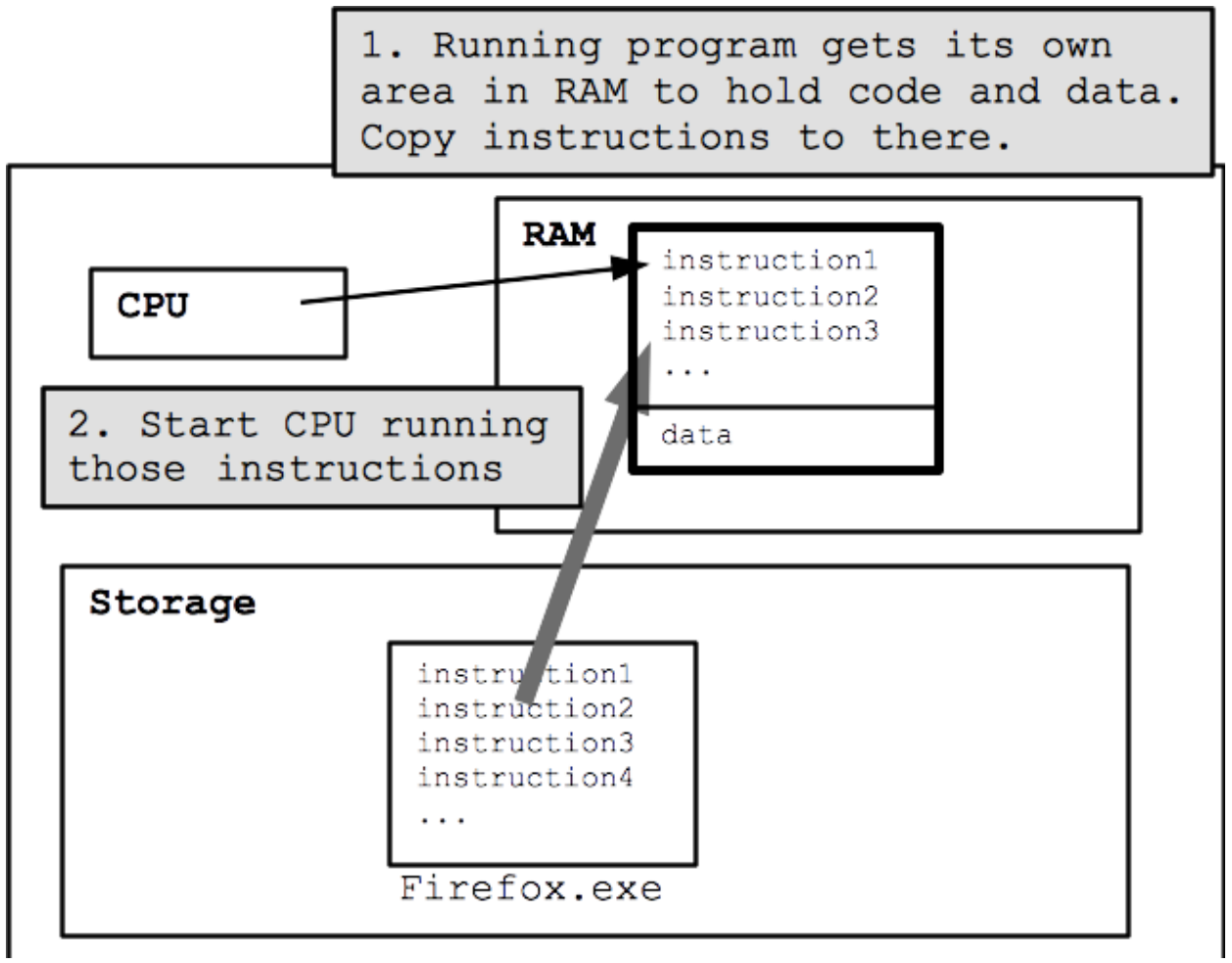
MORE & MORE FAILS

- “Spaghetti” code
  - Difficult to add new features
- Poor documentation/code commenting
  - “Legacy” code maintenance

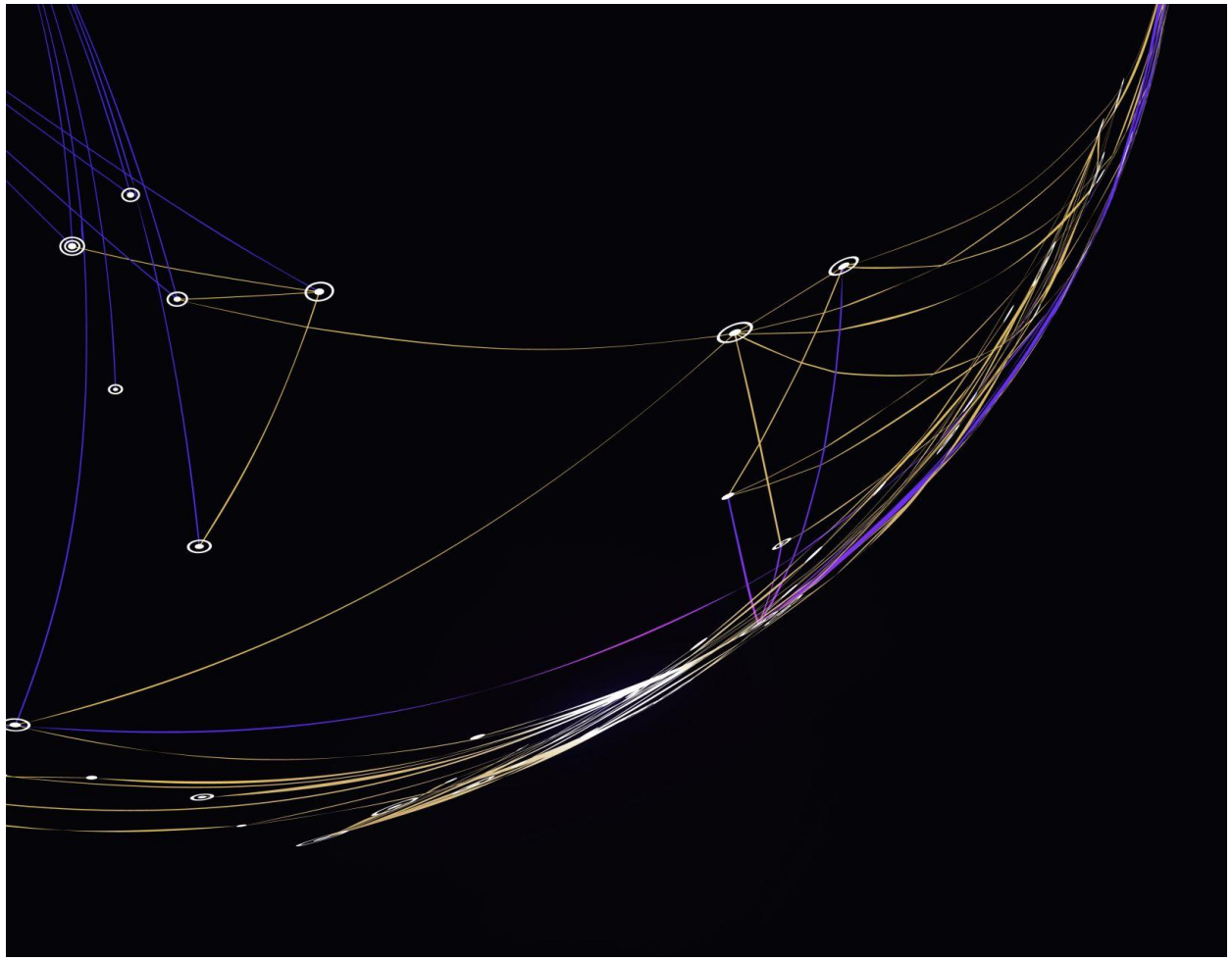
LET'S LOOK AT SOME CODE

https://github.com/hexa-decim8/girltalk/blob/master/girltalk.sh

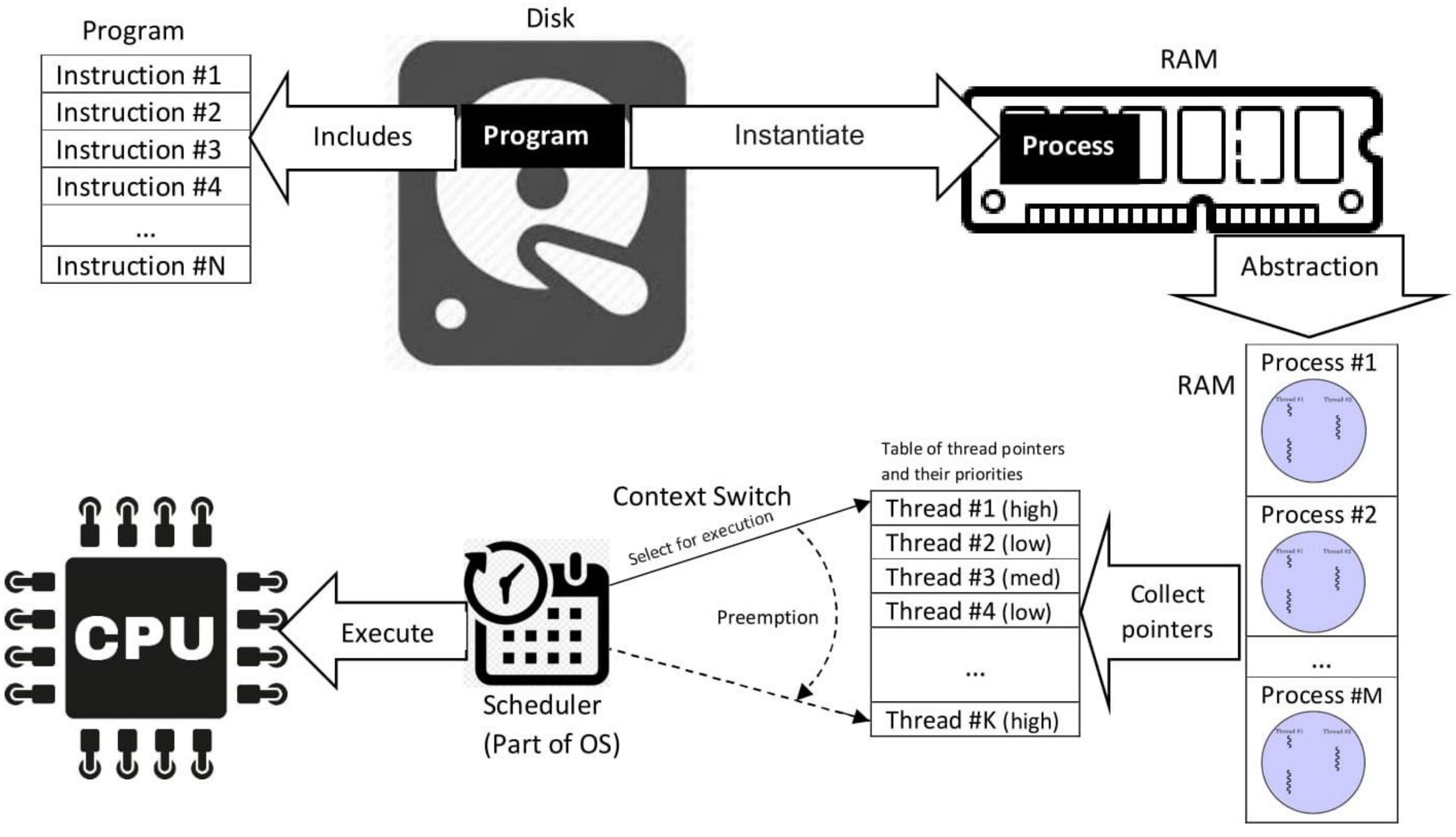
HOW A PROGRAM WORKS

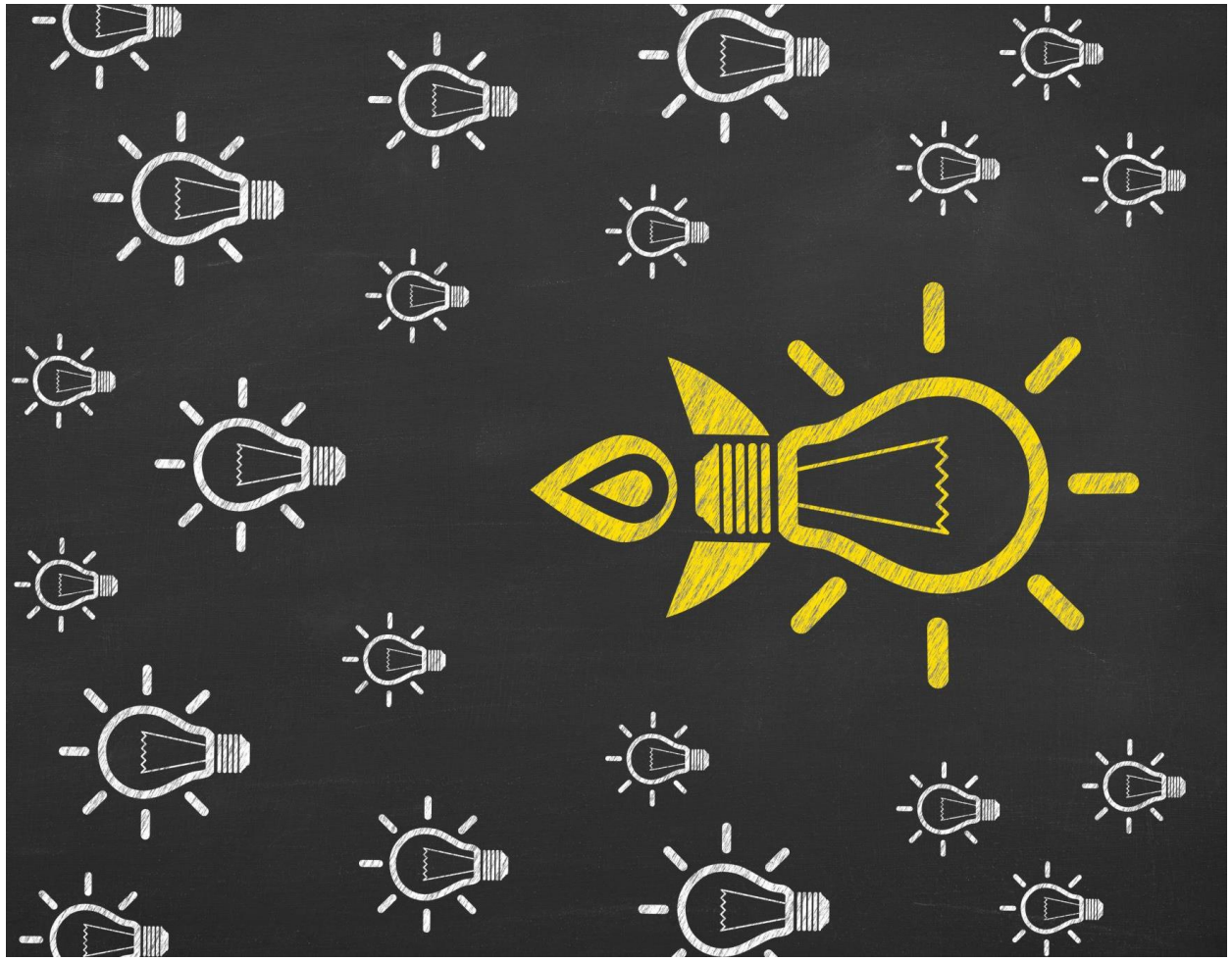


HOW FIREFOX WORKS



DETAILED EXECUTION


Program
Instruction #1
Instruction #2
Instruction #3
Instruction #4
...
Instruction #N
Disk
Includes
Program
Instantiate
RAM
Process
Abstraction
RAM
Process #1
Process #2
...
Process #M
Table of thread pointers
and their priorities
Thread #1 (high)
Thread #2 (low)
Thread #3 (med)
Thread #4 (low)
...
Thread #K (high)
Collect
pointers
Context Switch
Select for execution
Preemption
Scheduler
(Part of OS)
Execute
CPU



IMAGINE AN OUTSTANDING APP

TYING IT ALL TOGETHER

- Hardware is becoming much more diverse these days
  - Hardware can go places it didn't used to be able to
    - Software follows hardware
- Making good software is complicated!
  - Managing these projects is complicated


QUESTIONS?

DAY 1 REVIEW

DAY 2 PREVIEW

CONTACT INFO

ecrose@oakland.edu