Dec 5th

Computers Overview

Software Vs. Hardware

A hardware device is a device that allows you to interact with the computer, or that it is physical.

Note:

- Byte → 8 bits (1 bit is either 0 or 1)
- KiloBytes (KB) → 10^3 = 1,000 Bytes
- MegaBytes (MB) → 10^6 = 1,000,000 Bytes
- GigaBytes (GB) → 10^9 Bytes
- TeraBytes (TB) → 10^{12} Bytes
- PetaBytes (PB) → 10^{15} Bytes

Examples:

- CPU → Central Processing Unit aka chip aka processor
    - The fastest piece of the computer (literally what defines “speed”)
    - This is actually where all the information comes from → CPU tells RAM what to store, and tells the hard drive what to keep and what to delete
    - Instruction set
    - Cpu communicates with everyone
    - There is one cpu in the computer, but each cpu can have what’s known as multiple cores:
        - Each core acts as a standalone cpu
        - Examples:
            - Duo-core: 2 cores
            - Quad-core: 4 cores
            - Hex-core: 6 cores
            - Octa-core: 8 cores
            - 24 cores
    - Usually in the range of 2.1 GHz - 2.5GHz at 4-6 cores (mostly 4)
- Hard drive / internal volume aka disk
    - Think of this as long term memory/storage.
    - When the computer is on / off, the information is still stored and remains
    - This is waaaaaaay slower than RAM → it’s used to store information, not to necessarily retrieve it / have it ready all the time.
    - Usually in the range of 512GB - 10TB
    - Old school, out of date version of hard drives: HDD
    - SSD → solid-state drive (this is normally now in most machines)
- RAM → Random Access Memory aka memory

    - Comes in RAM sticks
    - Think of this as short-term memory
    - When the computer is on, there is information stored on RAM that can be quickly accessed.
    - When the computer is off, the information in RAM gets cleared
    - RAM allows your computer to multitask better, and to store more information at once
    - Usually in the range of 8GB - 128GB
- GPU → Graphics Processing Unit aka graphics card aka rendering cards
    - This is mainly for graphics rendering and operations
    - Nowadays, it can be used for machine learning and artificial intelligence
    - Say you wanna play a game in 4K 60 fps -- you need a good graphics card to allow the rendering to be possible
    - NVIDIA and AMD are the two main manufacturers
    - Gpus have memory on the actual cards to allow for performance
- Peripheral devices
    - Joysticks
    - Monitor
    - Mouse
    - Keyboard

Software is internal instructions for the computer to run on the particular hardware.
The instruction is code.

Examples:

- Microsoft Word → 16 million lines of code
- Google Chrome
- Safari
- Any video game
    - Executables that need to be copied
- Any application that you use

⇒ code (what you're gonna write) → compilation step aka your code is being compiled (turned into machine-readable code) assembled → machine-translated → 1s and 0s (machine instructions) → the output of the program/executable/your application

Note: run-time is an important concept in the pipeline

Designing hardware requires a strong understanding of circuits, electronics, and embedded systems.

Designing software requires a strong understanding of coding/programming, algorithms, and a mixture of both problem solving and analytical skills.

Hardware is also designed to solve problems -- but the problems are much more obvious

Example: keyboard → the requirements are much simpler and more apparent (mechanical vs butterfly, ergonomics, etc)

Software

If the hardware like the cpu is managing everything, what is giving all the instructions to it?

Operating System → OS

Examples of OS:

- macOS (previously called OS X)
    - Catalina,
    - El Capitain
    - Yosemite
    - …
    - Snow Leopard
- Windows
    - XP
    - Vista
    - 7
    - 8
    - “9” was skipped because it interfered with a lot of the code/logic written for Office and other MS products
        - JOKES
    - 10
- Linux
    - RedHat
    - CentOS
    - Ubuntu
    - Lubuntu (aka lightweight ubuntu)
- iOS
- iPadOS
- Android
- Horizon (Nintendo Switch)

DETOUR:

The greatest OS → Unix (created by Dennis Ritchie)

Before 1970, every time you built a computer, you'd need a special OS for it.
Unix came along and standardized everything -- it is an OS that has system requirements and if they are met, you can have it running on any acceptable hardware device.

macOS → proprietary flavor of Unix called Darwin
Windows → idk who cares / also idk and i forgot
Linux → very closely related to Unix but overall different (created by Linus Torvalds)

⇒ almost all engineers use linux and mac to write/develop software

ANYWAYS:

The OS looks pretty from the outside (which is what you're looking at), but in reality, it's very complicated and gross-looking.

You never directly interact with the underlying OS, you interact with modules that let you interact with it (GUI aka graphical user interface).

The "kerne" of the OS is the single most important (and central) part of the OS. It controls everything.

The only way to directly (or semi-directly) interact with the kernel is to use a shell.

A shell is an application that allows you to give it instructions on performing certain tasks.

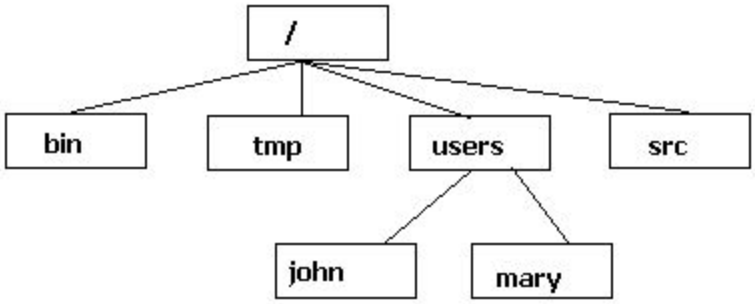
NOTE:

Command prompt on Windows machines is NOT a terminal window, but a console where some of the functionality of a terminal window holds.

Filesystem Tree

Windows → you have a bunch of drives, called C, D, etc.
Linux/Mac (Unix) → you have a single file structure

NOTE: directory = folder



/ → root directory
/Users → where all the users live
~ → home directory, which is usually /Users/username

Shell Commands

Command prompt:

[computer-name]: [path-to-current-directory] [username] $
Example:

1. mos-macbookpro: ~ mo $

NOTE:

1. Commands ALWAYS follow a $
2. There are hidden files and directories that won't appear in finder, but will appear in the terminal
    a. Hidden files and directories all start with "."

pwd

Presently working directory

This command is to know exactly where you are in the filesystem.

cd

Change directory

This command changes your position from one directory to another in the filesystem.

ls

List directories given a specific directory

mkdir

Make directory

rm (VERY DANGEROUS)

Remove → deletes a file or directory

Programming

In order to be able to communicate with the OS, and build large-scale applications, you need to program/code.

Programming = coding

Think of your code as a set of instructions that tell the OS what to do → you execute these instructions when your code “runs”
This can be done using:

- PyCharm
- Terminal

- Docker/containerization
    - Modern software development
- Proper orchestration (Kubernetes)
    - Modern software development

You write code in a specific language, called a programming language

We will be using Python as our programming language.
Reasons:

1. One of the most popular programming languages
2. Very VERY easy to pick up and understand
3. Very helpful when it comes to prototyping and scientific purposes
4. Very modular

Other languages:

1. C
    a. Awful language made in 1970s (what Unix was built-in)
2. C++
    a. An improvement over C
    b. Still very low-level
        i. It is very fast, but also very difficult
        ii. It's useful for embedded systems and circuits and gpu programming
        iii. Most games are written in C++
        iv. So are most applications at their core
3. Java
    a. The original language for Minecraft
    b. Started off being a slow-to-execute language, but has really kicked off and become extremely popular.
    c. JVM → Java Virtual Machine
        i. Several languages that are a part of the JVM
4. Python
    a. Mo's favorite
    b. Used a lot in industry, especially by
        i. Instagram
        ii. Snapchat
5. Javascript
    a. Completely unrelated to Java
    b. Often times a frontend language, but can also be a backend language
6. Scala
    a. Used for everything from server-side code to large scale data applications
    b. Very "safe"
    c. Functional (as opposed to imperative)
    d. Also on the JVM
7. Kotlin

    a. Created by Google for Android development
    b. Now it's much more multipurpose
8. HTML
    a. Mark-up language (not really programming), but we add it here anyways
9. Go
    a. Created by Google
    b. Very common nowadays for backend
10. PHP
    a. The worst thing ever
    b. What all of Facebook is written in

Generally, the easier the language, the more "baggage" it has
Baggage:

- The underlying features of the language have to do a lot of heavy-lifting for you
- The language is easier, but you're limited in a lot of ways
- The implication is the speed at which your application is built/works

NOTE:
With modern hardware, no language is "too slow," and most can do with the slight performance hiccups.

The goal of learning how to program and code is NOT to learn a language.

Software engineering is not about a language → it's about DESIGN and ANALYTICAL SKILLS

Languages are different from each other (this is called syntax), but the overall mindset is what matters.

Always depth over breadth!

Python

There are technically 2 main versions of python: python 2.7 and python 3.
→ python 2.7 is deprecated (discontinued support)

We will be using python 3.x

It can be both interactive and scripted (script-based → our main focus)

We will be using PyCharm

Python files will have the extension of “.py”
NOTE:

- Microsoft word documents are “.doc” or “.docx”
- Text files are “.txt”
- C++ are “.cpp”
- Python files are “.py”