As we begin this class, I would like to acknowledge and pay respect to the traditional owners of the land: the Gadigal people of the Eora Nation. It is upon their ancestral lands that the University of Sydney is built. I pay respect to their Elders past and present, and to any First Nations people in the attendance.

As we share our own knowledge, teaching, learning and research practices within this university may we also pay respect to the knowledge embedded forever within the Aboriginal Custodianship of Country.

THE UNIVERSITY OF SYDNEY

INFO1112
Computing 1B OS and Network Platforms

Dr Hazem El-Alfy

Week 1 - 31/7/2023

Photo by Morgan Richardson on Unsplash

Welcome to INFO1112

- Introduction
- Administrative info
  - Covid University policies
  - timetable
  - assessment
  - special consideration
- Intro to the course
- Python demo

Lecturer:

Dr Hazem El-Alfy

PhD in Computer Science
University of Maryland, USA

Research Area: Machine Learning and Computer Vision

I have been teaching since 1997

Started teaching at Uni Sydney five minutes ago!

THE UNIVERSITY OF SYDNEY

Teaching Assistants

Tiancheng (Michael) Mai

Chris Polak

Preethom Pal

Daniel Kovalenko

THE UNIVERSITY OF SYDNEY

Tutors

David (Hongyi) Shi

Ali Aljaafreh

Gabriel Timothy

Pratham Purohit

Mohammad Saad Azam

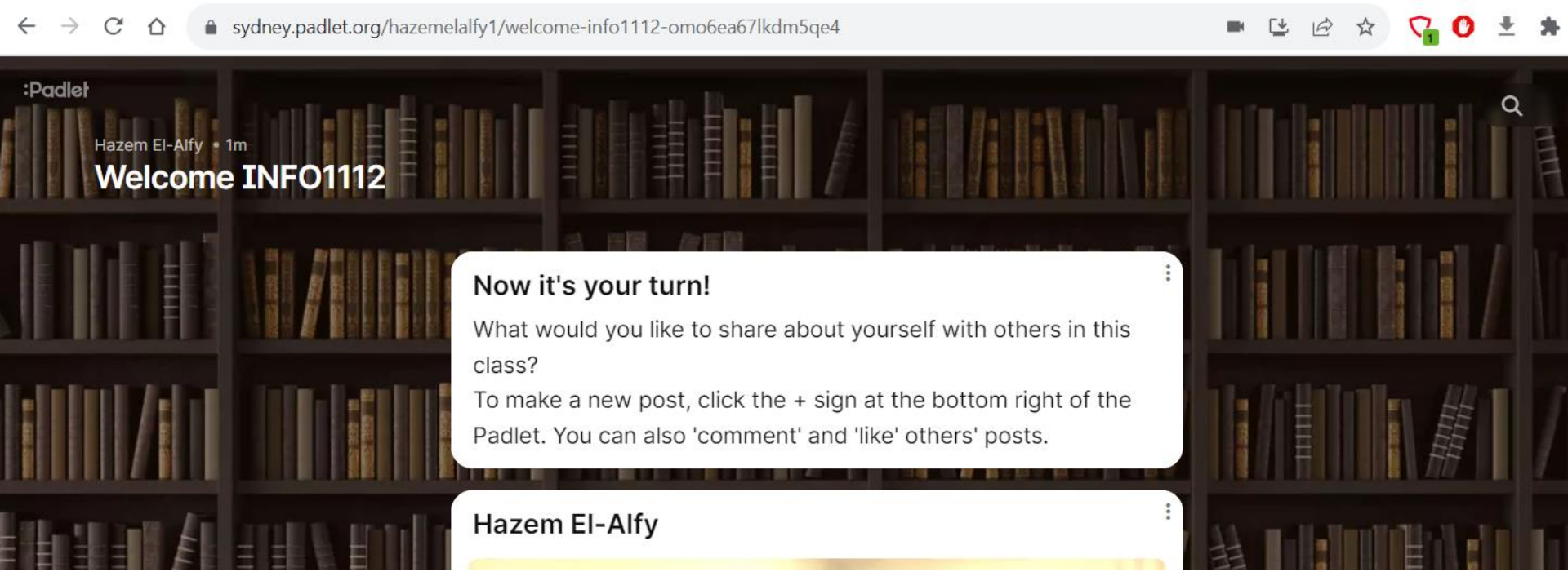
George Wang, Benjamin Braham, Ashek Ahmmed, Rubaina Tausif, Ashton Liu, Mostafa Shahin

THE UNIVERSITY OF SYDNEY

Activity – Padlet Meet and Greet

What would you like to share about yourself with others in this class?

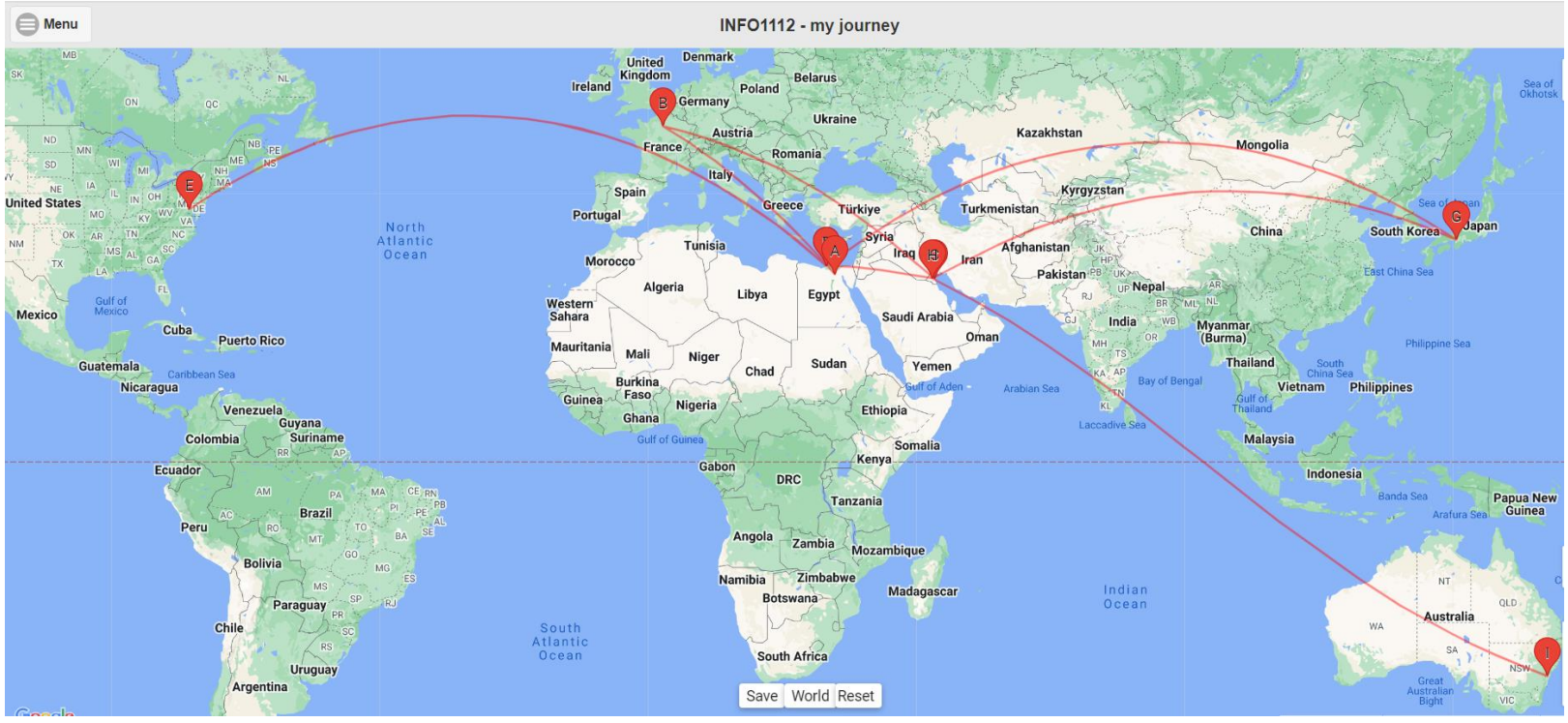
Padlet Page



Activity – World Map Meet and Greet

Where are you from? Where did you go around the world?

World Map



INFO1112: Places

- Lecture: Monday 12-2pm online via Zoom
  - Zoom link is everywhere on Canvas
  - If you are on campus, you can use J07.02.S213 during lecture time but bring your own device.

- Lab:
  - Face-to-Face. No online option this semester.
  - Please attend the lab you are scheduled for.

Expectations

- Students attend scheduled classes, and devote an *extra* 6-9 hours per week
  - doing homework and assignments
  - preparing for classes, watching pre-record video before class
  - revising and integrating the ideas
  - practice and self-assess
- Students are responsible learners
  - Participate in classes, constructively
    - Respect for one another (criticize ideas, not people)
    - Humility: none of us knows it all; each of us knows valuable things
  - Check Canvas site regularly!
  - Notify me or your tutor if there are difficulties - use the Ed forum

INFO1112 Assessment

Assessment Information on Canvas

Description	Marks	Type	Due
6 Homework Exercises	20	shell/Ed	fortnightly starting week 2
Assignment 1	10	intro python/Ed	week 5
Assignment 2	20	python/Ed	week 11
Mid Semester Quiz	10	online	lecture of Week 7
Final Exam	40	written	lecture of wk 13
Total	100		

Late submissions in INFO1112

- **No late submissions accepted for homeworks**
- Late submission policy for assignments:
  - If you have not been granted special consideration, a penalty of 5% of the maximum marks applies for each late day (or part of). After ten days, you will be awarded a mark of zero.
  - *e.g. If an assignment is worth 40% of the final mark and you are **one hour** late submitting, then the maximum marks possible would be 38%.*
  - *e.g. If an assignment is worth 40% of the final mark and you are 28 hours late submitting, then the maximum marks possible marks would be 36%.*
- Warning: submission systems get very slow near deadlines
- Submit early, you can resubmit if there is time before the deadline

Special Consideration (University policy)

- If your performance on assessments is affected by illness or mishap
- Follow the proper procedure:
  - Have professional practitioner sign special USyd form
  - Submit application for special consideration online
  - Note you have only a quite short deadline for applying
  - http://sydney.edu.au/current_students/special_consideration/

Academic Integrity (University policy)

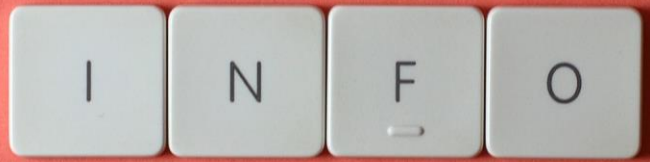
- The University of Sydney is unequivocally opposed to, and intolerant of, plagiarism and academic dishonesty.
  - Academic dishonesty means seeking to obtain, or obtaining academic advantage for oneself or for others (including in the assessment or publication of work) by dishonest or unfair means.
  - Plagiarism means presenting another person's work as one's own work by presenting, copying or reproducing it without appropriate acknowledgement of the source. [from site below]
- http://sydney.edu.au/elearning/student/EI/index.shtml
- Submitted work is compared against other work (from students, the internet, etc)
  - **Ed will do this for all assessments in INFO1112**
- Penalties for academic dishonesty or plagiarism can be severe
- You must complete the self-education module AHEM1001 (required to pass INFO1112)

Academic Integrity (Use of AI Tools)

- The new Academic Integrity Policy 2022, which came into force on 20 February 2023, mentions Generative Artificial Intelligence technologies in three key places.
  - Clause 4(9)(2)(j)(i) states that it is an academic integrity breach to inappropriately generate content using artificial intelligence to complete an assessment task.
  - Clause 4(13)(1)(i) states that submitting an assessment generated by AI may be considered contract cheating.
  - Clause 5(16)(5)(b) mentions that students must acknowledge assistance provided when preparing submitted work, including the use of automated writing tools.

- Please familiarise yourself with the University's policy on the use of AI tools in assessments. Check the 'Frequently asked questions about generative AI at Sydney'

Course



The Course (finally!)

- "bits to applications"
- How does it work?
- a *spiral* approach - almost all the topics covered in the course will be covered in much greater depth in other courses
- INFO1112 is designed to introduce you to the underlying technical ideas

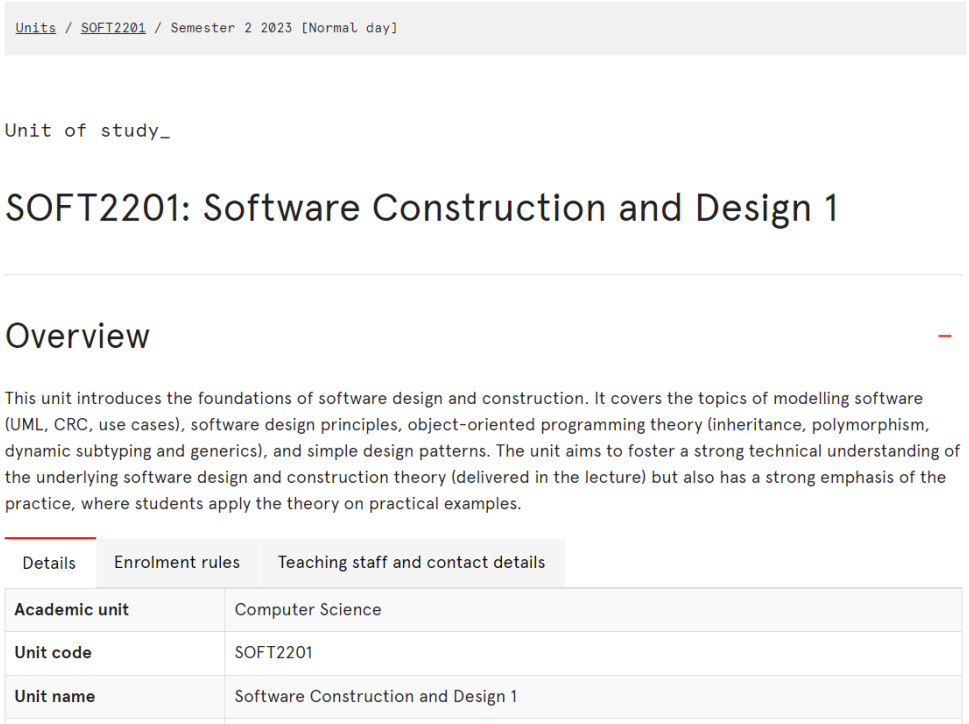
The Course Components

- A lecture session every Monday 12-2pm
- Lecture recording will be available the next day on Canvas, “Recorded Lectures“ tab.
- A weekly F2F lab with exercises due in lab.
- Homework due fortnightly in weeks 2, 4, 6, 8, 10, 12
- Two assignments during semester
- Mid-semester quiz (online)
- Final paper exam.

INFO1112 Weekly Outline

The topics for each week in 2023 are tentatively:

- W1: introduction, binary representation, operating system
- W2: processes
- W3: file system, name space
- W4: the boot sequence
- W5: virtual machines
- W6: networks
- W7: internet protocol
- mid-semester break
- W8: domain name service, application layer
- W9: public holiday
- W10: computer and network security
- W11: cloud computing (~)
- W12: window systems and Android (~)
- W13: Micro-services + course review



Units / SOFT2201 / Semester 2 2023 [Normal day]

Unit of study_

SOFT2201: Software Construction and Design 1

Overview

This unit introduces the foundations of software design and construction. It covers the topics of modelling software (UML, CRC, use cases), software design principles, object-oriented programming theory (inheritance, polymorphism, dynamic subtyping and generics), and simple design patterns. The unit aims to foster a strong technical understanding of the underlying software design and construction theory (delivered in the lecture) but also has a strong emphasis of the practice, where students apply the theory on practical examples.

Details | Enrolment rules | Teaching staff and contact details

Academic unit	Computer Science
Unit code	SOFT2201
Unit name	Software Construction and Design 1

Week 1

Short Segments this week:

- binary numbers
- data representation inside computers
- operating systems
- a review of Python

Lab session **this week**: revision of shell commands.

No homework this week but there is homework due end of Week 2.

Canvas will be updated as soon as the material is ready.

Assignment 1

Description will be released next week in Ed

The assignment is Unix / Python based

Scaffold will be available in Ed.

Due date: Week 5 Sunday, Sep 3rd 11:59 PM.

Whole class Survey

In Mentimeter

A bit about you and your technology

3 minutes survey

THE UNIVERSITY OF SYDNEY

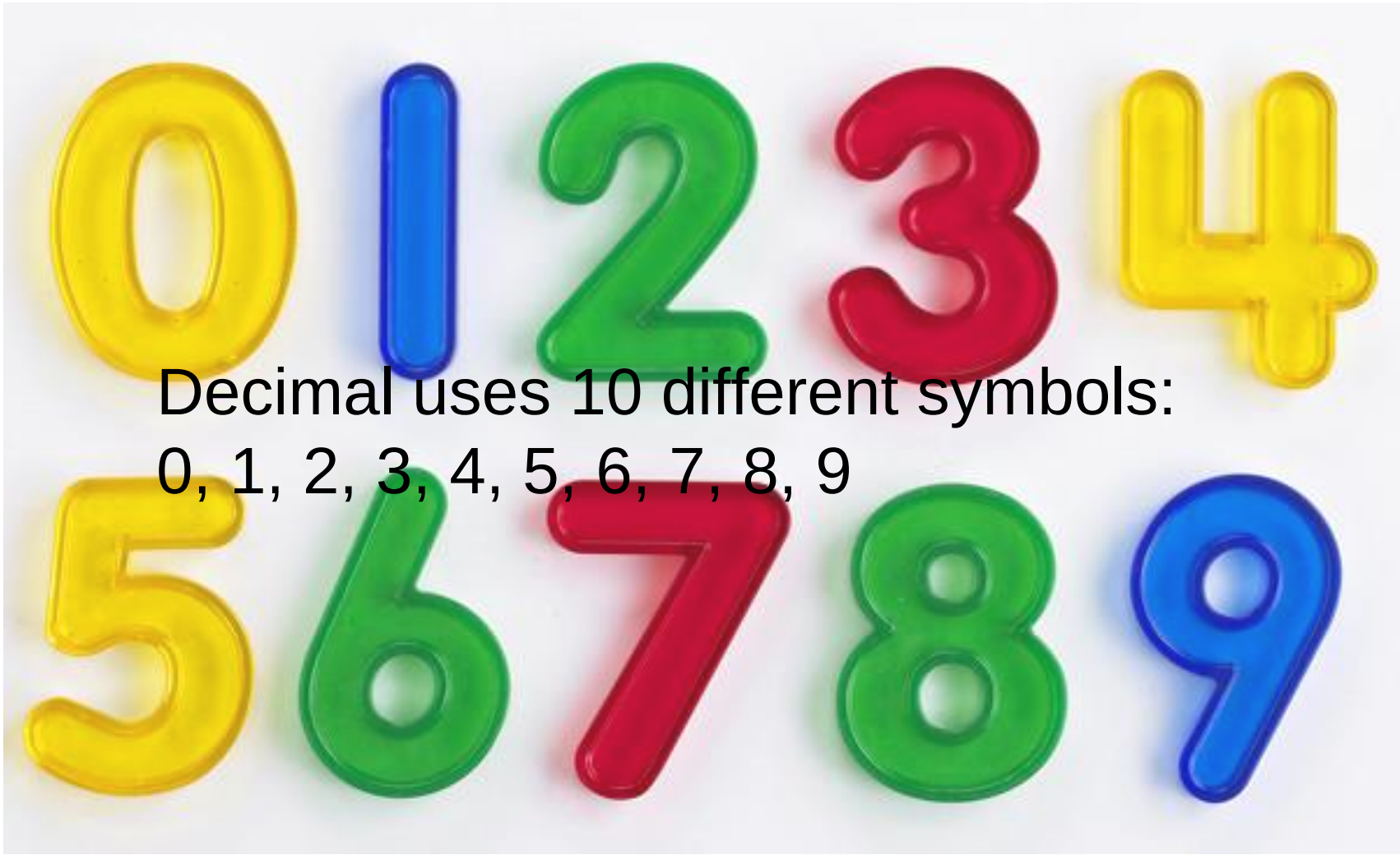
BREAK

Binary Numbers

by

Bob Kummerfeld

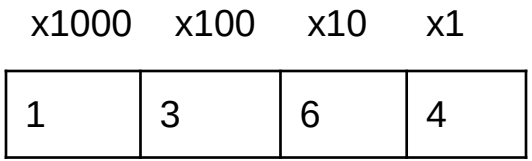
Let's start with the DECIMAL number system that we use every day.


Decimal uses 10 different symbols:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ten symbols allows ten different values. To represent higher values we could make up more symbols.

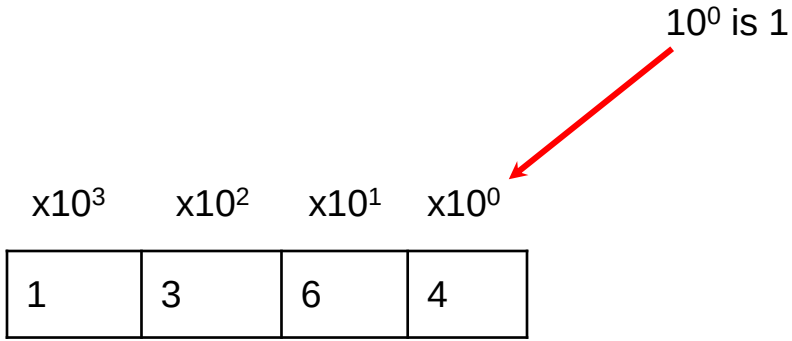
or we could use a sequence of the symbols and attach meaning to the ***position*** of the symbol in the sequence

Example



1 x 1000 + 3 x 100 + 6 x 10 + 4 x 1 = 1364

Each position is a power of ten



$1 \times 10^3 + 3 \times 10^2 + 6 \times 10^1 + 4 \times 10^0 = 1364$

Each position in the number has a weight.

The rightmost has a weight of one.
The second to the right has a weight of a hundred.
The third to the right has a weight of a thousand.
...and so on.

The weights are a power of 10.

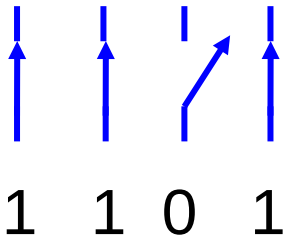
The BINARY number system only uses two symbols:

0 and 1

Why Binary?

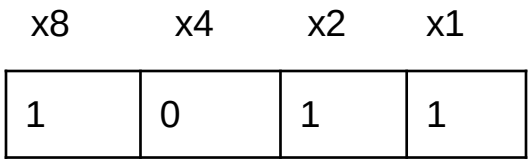
Computer memory is created from memory elements that are either **on** or **OFF**, representing 1 or zero.

These binary digits, or **BITS** are usually arranged in groups of 8 called **BYTES**.



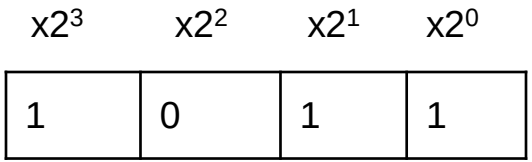
For BINARY the positions represent powers of 2 rather than powers of 10 used in decimal.

Example



1 x 8 + 0 x 4 + 1 x 2 + 1 x 1 = 11

Each position is a power of two



$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

Binary numbers take up more space than decimal!

For example, the number 9384 in decimal is 10010010101000 in binary.

One way to save space is to use another number base, other than 2 or 10, to represent the number.

If we choose a number for the base that is also a power of two it makes it easy to convert each "digit" into binary and so convert the whole number.

The most common base used in computing is base 16, usually called HEXADECIMAL.

The word comes from the Greek "hex" for six and Latin "deci" for ten.

Originally the British called base 16 "Sexadecimal" from the Latin word "sex" for six. It didn't catch on.

For hexadecimal or base 16 we need 16 different symbols, so we use 0 to 9 and A to F.

0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

For example, the number:

10010010101000 in binary can be written

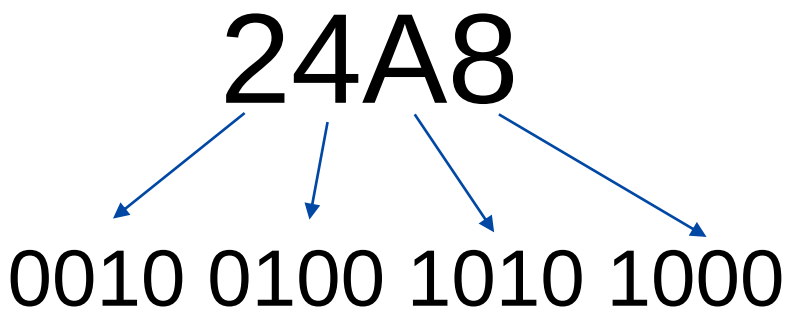
24A8

much shorter!

It is easy to convert to and from binary because each digit in the hexadecimal number represents a number in the range 0-15. If we write the digit in binary form it is:

0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

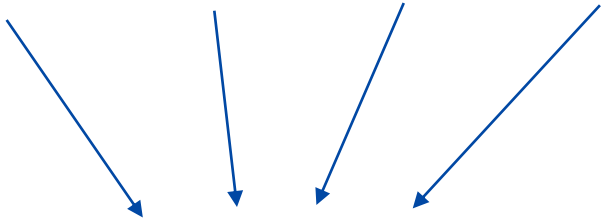
If we take each digit in the hex number and convert it we get:



and that is our original number.

We can also do the reverse - convert a binary number to Hexadecimal easily:

0010 0100 1010 1000



24A8

You soon get used to converting 4 binary digits into the single hex digit.

Since bytes consist of 2 groups of 4 bits, it is very convenient to write their values as 2 hex digits.

Data Representation

by

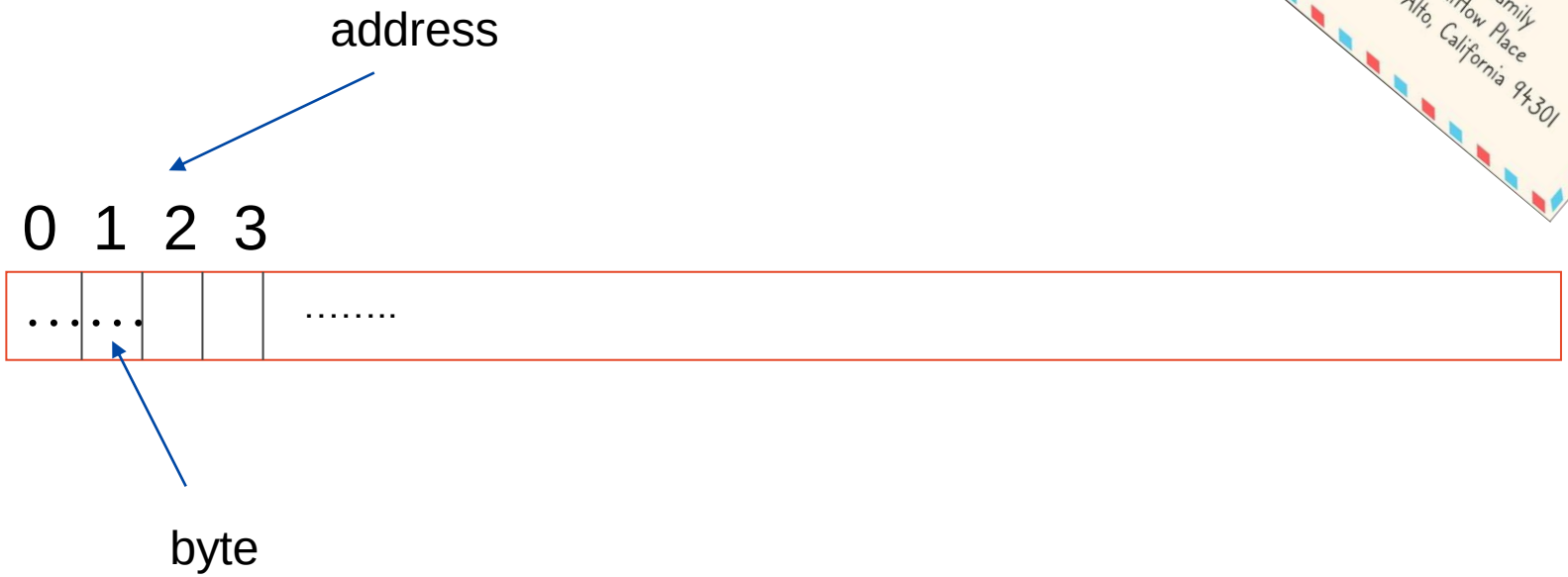
Bob Kummerfeld

Memory

- The memory of a computer is made up of binary digits or ***bits***
  - But not just a simple stream of bits
- Memory is divided into 8 bit pieces called ***bytes***
- These pieces are each given a number from 0 to the number of bytes in the memory
- Each of these numbers is called a memory ***address***

Memory


Dr. Michael Smith
123 ABC St
Los Angeles, CA 94301
The Jones Family
1234 wikiHow Place
Palo Alto, California 94301
stamp


address
0 1 2 3
byte


Address Book

We represent **everything** in the computer using patterns of binary digits.

Integers, fractions, characters, images, sound, video….

Computers can store numbers (integers)

1 = 1

2 = 10

3 = 11

4 = 100

5 = 101

...etc...

We can store these bit patterns in bytes.

But each byte is only 8 bits long – this puts a limit on the range of values that can be represented. (How big?)

Numbers in the computer are not the same as integers in mathematics!

Computers don't have infinite memory!

- Number representations are finite: we choose a certain number of bits to represent a number
- Representations matter: we agree on what the bits mean
- Memory is finite; numbers are infinite

Computers can also store characters

We can choose some patterns of bits to represent characters.

Characters are often represented in a single byte, but this restricts the range of characters (how many?).

Characters can also be represented with multiple bytes, either a fixed number of bytes (eg 2) or a variable number of bytes (how do we know how many?).

Character Representation

Example: We can represent characters using a code that associates an 7-bit number with a character

A = 1000001 = 65 (decimal)

B = 1000010 = 66

…

a = 1100001

b = 1100010

space = 0100000

etc

American Standard Code for Information Interchange

These are examples in the ASCII character code

Maximum code value is 127 (7 bits) or 1111111

What is the implication?

Character Codes

Only 0-127 (128) characters can be represented in ASCII

Fine for English, what about other languages? Chinese? Or even French?

There are many character codes in use.

After many years, and many proposed standards, a character code called "**Unicode**" was developed by a consortium of companies and in cooperation with the International Organisation for Standardisation and first released in 1991.

Character Codes

Unicode uses 32 bits/character (4 bytes) and so could theoretically represent over 4 billion characters!

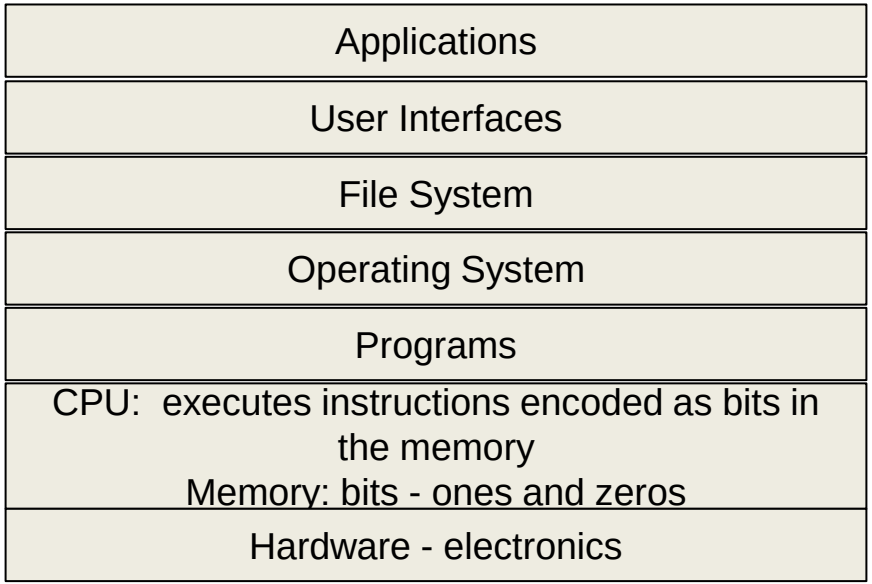
Currently (2020) there are over 140,000 characters represented in unicode that cover more than 150 languages.

There are also characters that are no longer used except in historical documents represented in Unicode. And there are "emojis" 👏😄👍

😄 has code 1F600 in hexadecimal

😀 has code 1F637 in hexadecimal

Machine Instructions

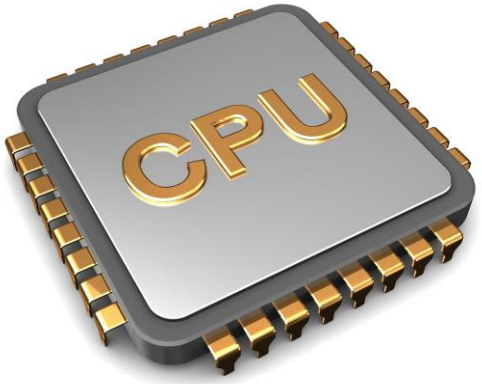


Instructions or "Code"

Computer programs are a set of instructions executed by the central processing unit.

- As well as data (numbers, characters etc), we also store the program instructions (or ***code***) in the memory of the computer
- A program written in a language like C is translated into machine instructions that are loaded into the memory and executed by the central processing unit (CPU)
- Machine instructions are also patterns of bits
- Programs can create machine instructions, store them in the memory and then get the CPU to execute them

Central Processing Unit - CPU



› Instructions are represented in bits and stored in a sequence of bytes, eg:

01100100 10001100

This might (in some hypothetical machine) mean “add the integer in address 4 to the integer in address 8 and store the result at address 12”

› the idea to store instructions in the same memory as values was **the** major breakthrough in the development of computers
  - “stored program computers”

Instructions

The example might be broken down as:

0110 code for “add” instruction

0100 address 4

1000 address 8

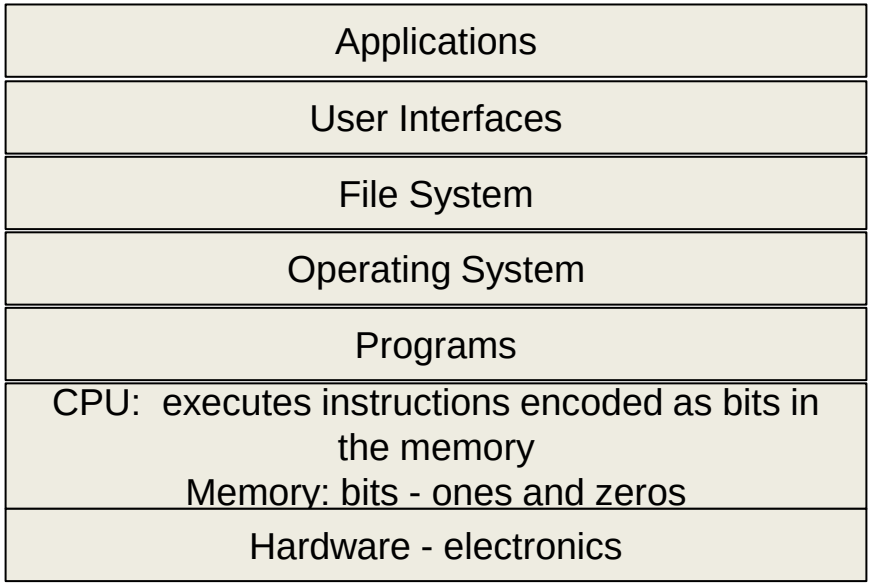
1100 address 12

Operating Systems

by

Bob Kummerfeld

Operating Systems



A "simple" system consists of memory+CPU+input/output devices.

Program instructions are stored in memory and executed by the CPU and may read/write data from external devices such as keyboard and screen.

(you will develop programs like this in ELEC1601)

But systems such as MS Windows or Apple MacOS provide far more:

- sophisticated graphical user interface
- other I/O devices such as a mouse
- a file system
- network connections
- concurrency (running more than one program at a time)
- security

These services are generally provided by an ***Operating System***

The Unix Operating System

Invented long long ago (1969) in a place called Bell Laboratories in the USA by Dennis Ritchie and Ken Thompson. Many others contributed.

Released to Universities in source code form in 1975.

Many versions of the original and many clones (eg Linux) have been produced since then. All are recognisable as a Unix based system.

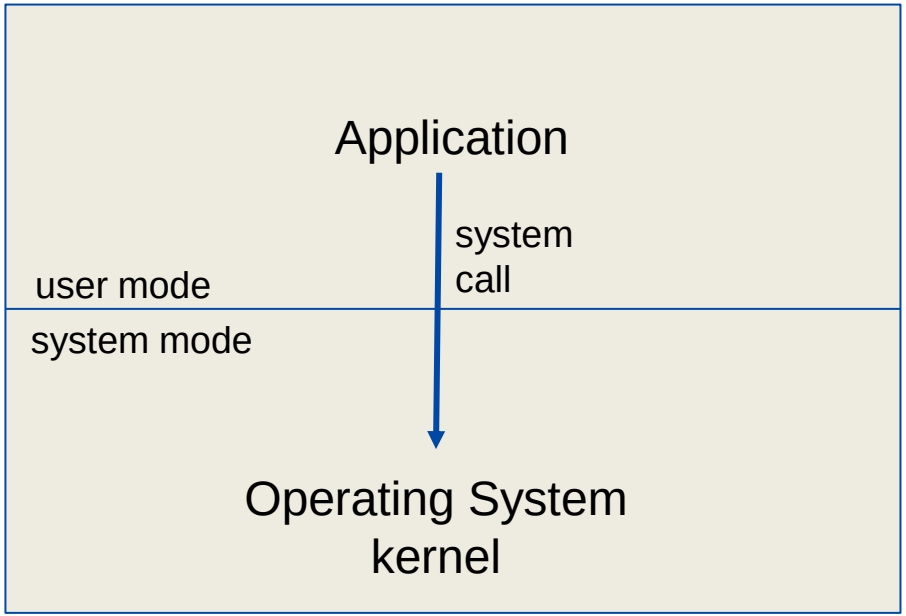
We will use Linux for all our examples and lab exercises.

Linux is a Unix clone developed originally by Linus Torvalds, a Finnish programmer. Linux is now probably the most widely used operating system. Unix based systems are the basis for Android, Apple (MacOS, IOS, ipadOS, watchOS etc) and many others.

An Operating System is a special program that is *privileged* (can access and control all of the hardware) and provides all of the basic services. For other services such as the graphical user interface, the operating system is supplemented by application programs.

To enforce the distinction between the operating system and user programs the CPU usually has (at least) two modes of operation: system and user.

To switch from user mode to system mode (eg when an application program requests a service such as reading a file) and special CPU instruction is executed - system call. This will cause execution to continue in the operating system program in system mode.



The File System

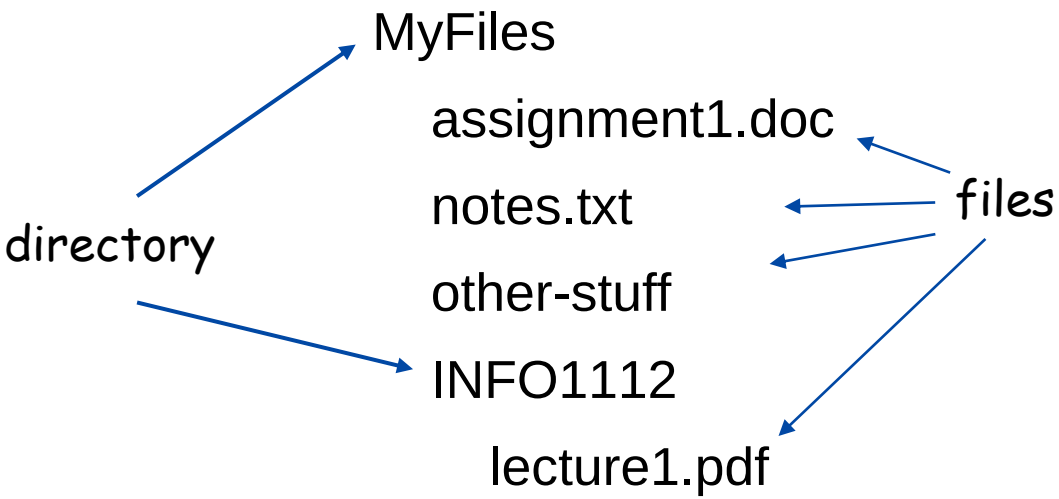
To store "persistent" data that doesn't disappear when we turn off the computer we need storage other than main memory. This can be on a separate device such as a disk drive connected to your computer or on another computer connected to the network.

"disk" drives are implemented with various technologies but all provide **persistent** storage.

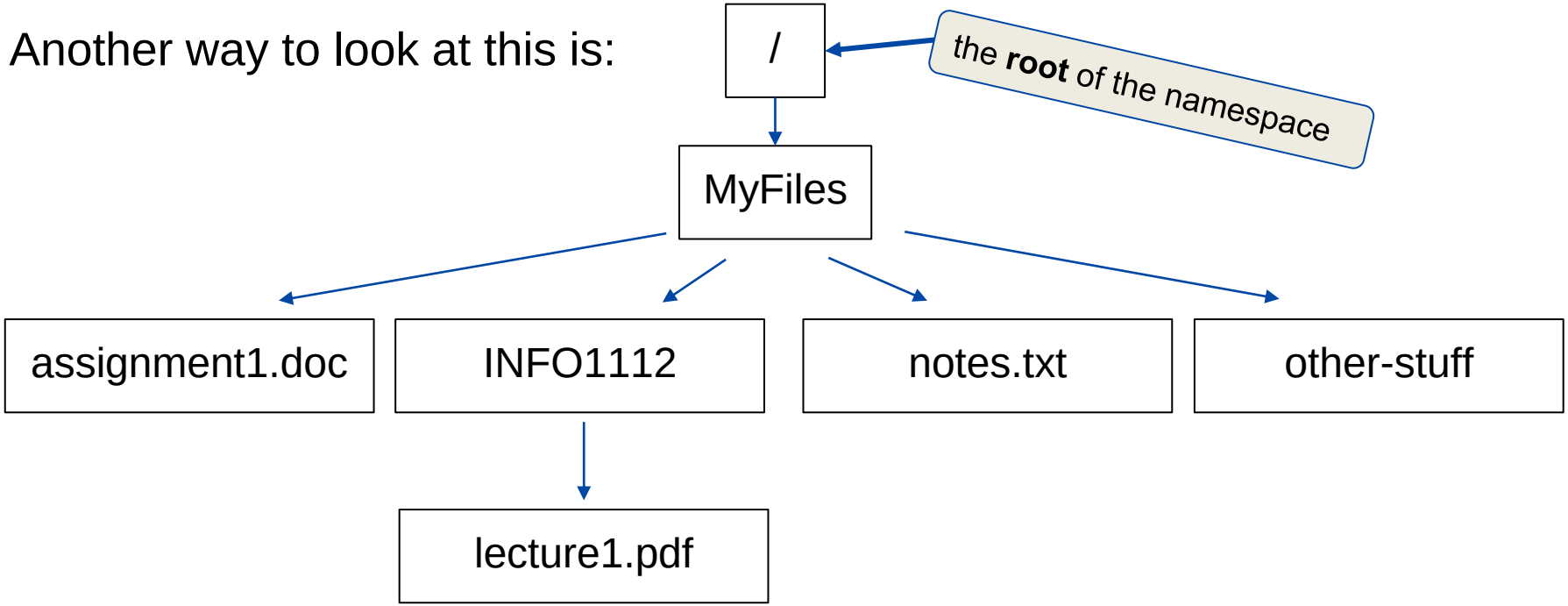
The raw storage provided by these mass storage devices is not so convenient to use, so in nearly all cases we structure the storage into ***files*** and ***directories***.

This is what we call the ***file system***.

The file system (in Unix) consists of a set of directories (or folders) that can contain files or sub-directories.



Another way to look at this is:



This is called a ***tree*** data structure

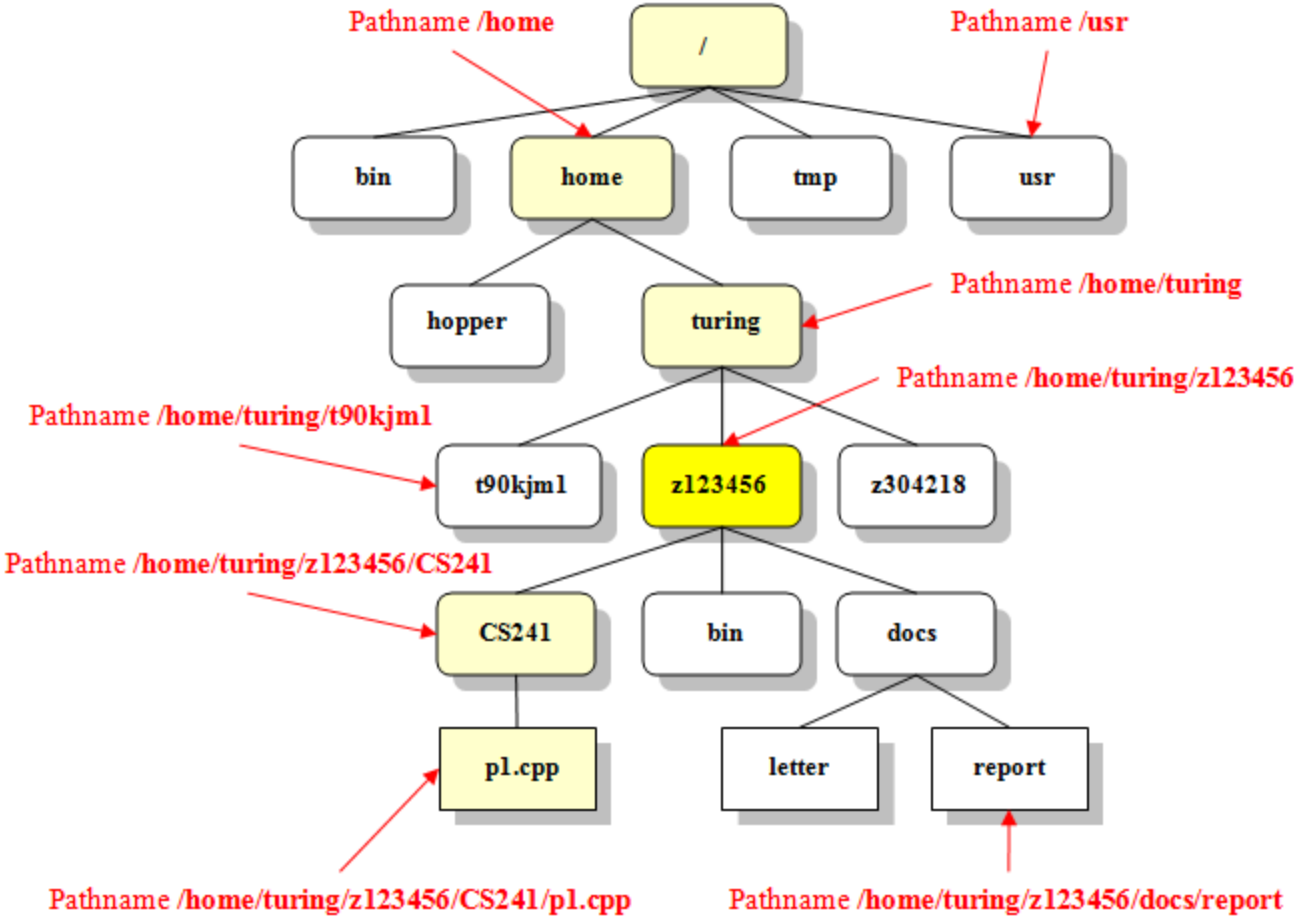
The names and the structure of the file system are called the ***name space***

To specify a given file or directory in the Unix file system *name space* we write the ***path name*** of the file or directory.

For example: to specify the lecture1.pdf file we might write:

/MyFiles/INFO1112/lecture1.pdf

This tells us that "MyFiles" is a directory at the ***root*** of the file system and that “lecture1.pdf" is a file contained in the "INFO1112" directory, which is contained, in turn in the "MyFiles".



https://faculty.cs.niu.edu/~mcmahon/CS241/Notes/Unix_Reference/file_structure.html

Image source: https://ourcodeworld.com/

Python

Python is an open source programming language developed in the early 1990s by Guido van Rossum

Knowledge of Python is assumed for this course.

I know that some of you are learning python in parallel with INFO1112. by taking INFO1110.

If you don't have knowledge of Python and want a quick start, try this https://www.w3schools.com/python/

If time permits, I may spend some time, occasionally, working through some Python examples.

Python

Python Developed in early 1990s by Guido van Rossum

Name comes from the British TV comedy “Monty Python’s Flying Circus” (early 1970's)

Today a large number of people contribute to the development and maintenance of Python

Python is *open source* meaning it can be downloaded and used for free and modified in any way

Python

Clean and simple syntax, easy to read and write.

Encourages good programming habits

Has a LOT of power… but doesn't get in the way of learnability

Has a large standard library ("comes with batteries included")

Python scripts are portable to anywhere the interpreter runs.
Python runs almost everywhere, even tiny machines such as arduino or the micro:bit

Python is available (for free) for: Windows, Apple, Unix/Linux

Activity

On a piece of paper write a Python function that takes an integer argument and prints a triangle of that many Xs?

eg given 3, print this:

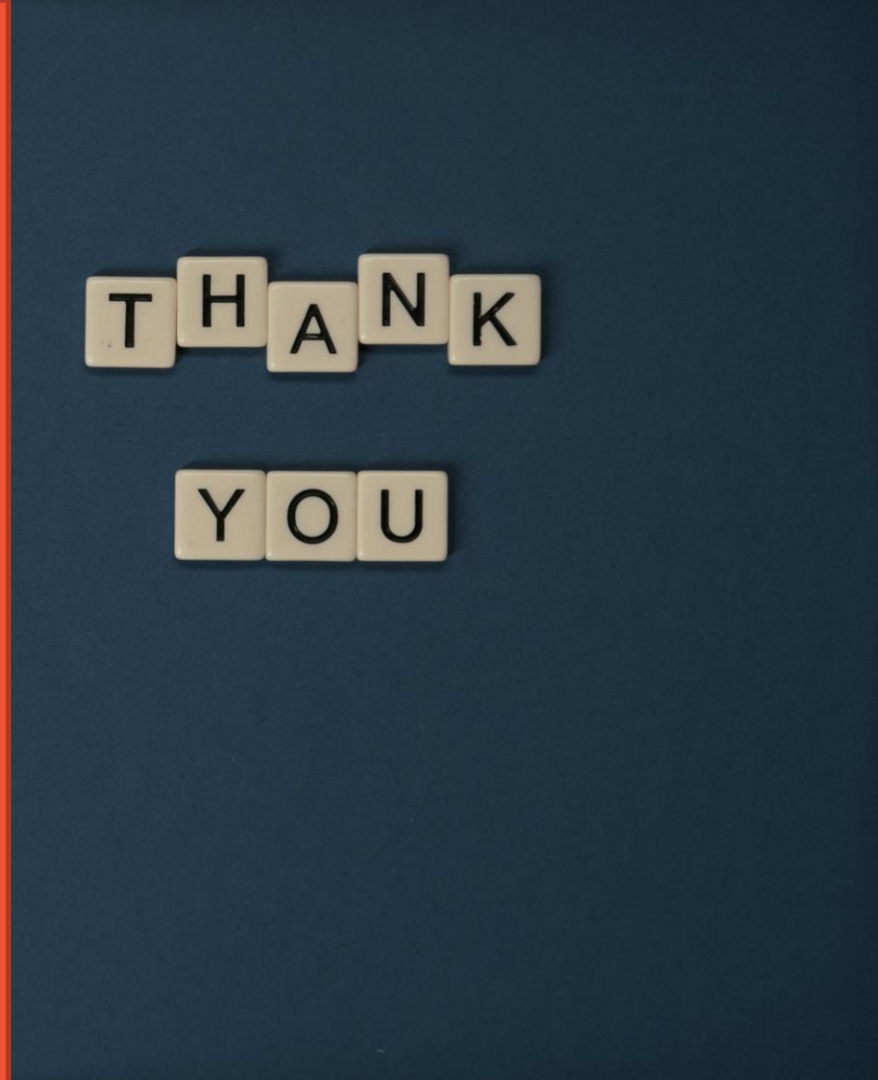
X
XX
XXX


Demo
Example code

Acknowledgement

This unit was designed by **Robert Kummerfeld** and modified by **Nazanin Borhan**. Shout out to them and their great contribution to INFO1112.

Also thank you to the TAs and Tutors without whom this unit would not run smoothly.




Questions?