Dr David A McMeekin

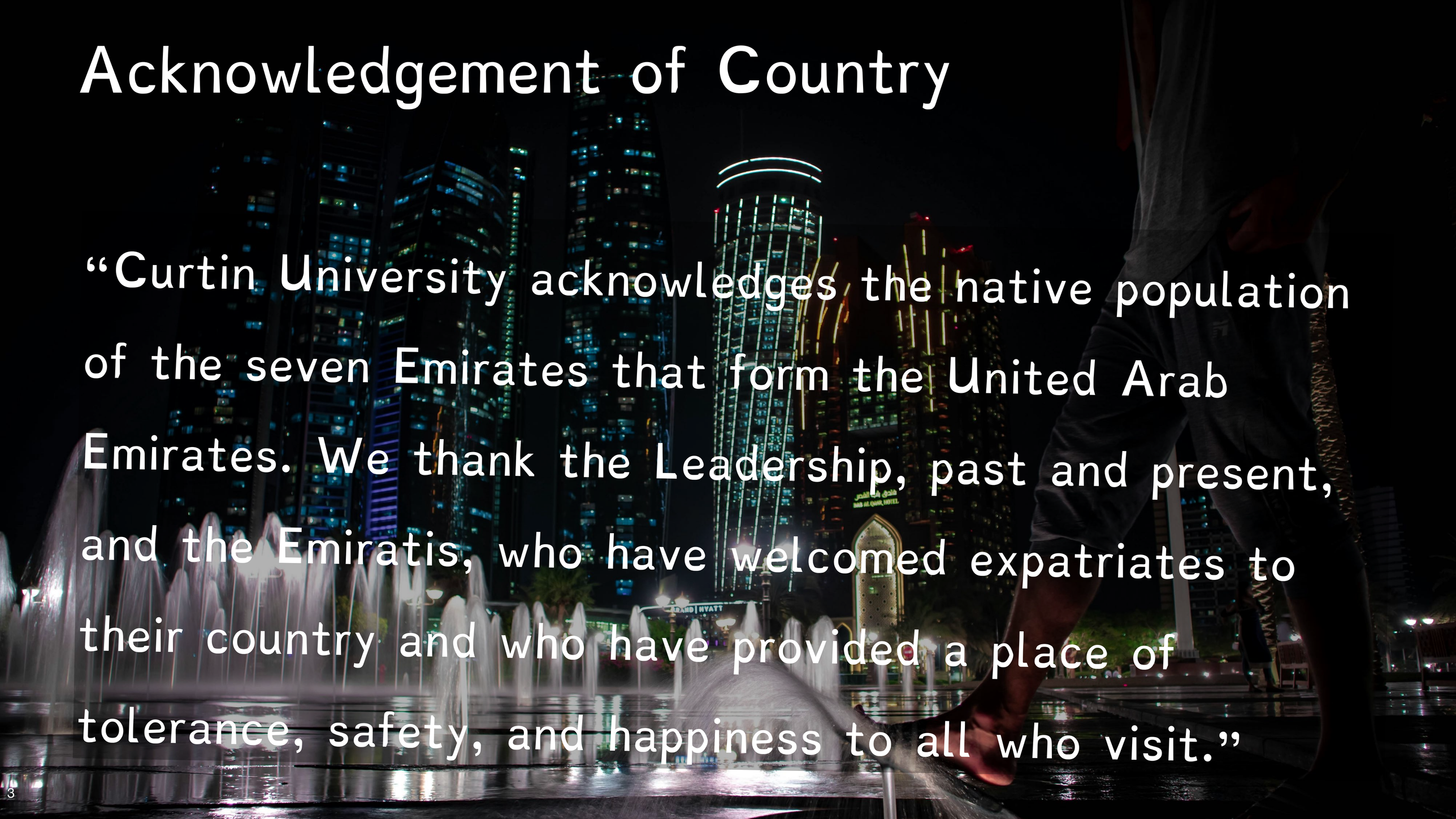
Introduction

Programming Design and Implementation

Acknowledgement of Country

“I acknowledge the Wadjuk people of the Noongar nation on which Curtin University’s Boorloo Campus sits, and the Wangkatja people where the Karlkurla Campus sits, as the traditional custodians of the lands and waterways, and pay my respect to elders past, present and emerging.”

Acknowledgement of Country



“Curtin University acknowledges the native population of the seven Emirates that form the United Arab Emirates. We thank the Leadership, past and present, and the Emiratis, who have welcomed expatriates to their country and who have provided a place of tolerance, safety, and happiness to all who visit.”

COMP1007 - Unit Learning Outcomes

- Identify appropriate primitive data types required for the translation of pseudocode algorithms into Java;
- Design in pseudocode simple classes and implement them in Java in aLinux command-line environment;
- Design in pseudocode and implement in Java structured procedural algorithms in a Linux command-line environment;
- Apply design and programming skills to implement known algorithms in real world applications; and
- Reflect on design choices and communicate design and design decisions in a manner appropriate to the audience.

Outline

- Computer Basics;
- UNIX;
- VIM;
- Binary;
- Octal & Hex;
- Signed Integers; and
- Real Numbers.

Computer Basics

What is a computer?

- Machine that accepts data, processes it and produces output;
- Computer consists of:
  - CPU – Central Processing Unit;
  - Input/Output devices;
  - Dynamic memory; and
  - Secondary storage.

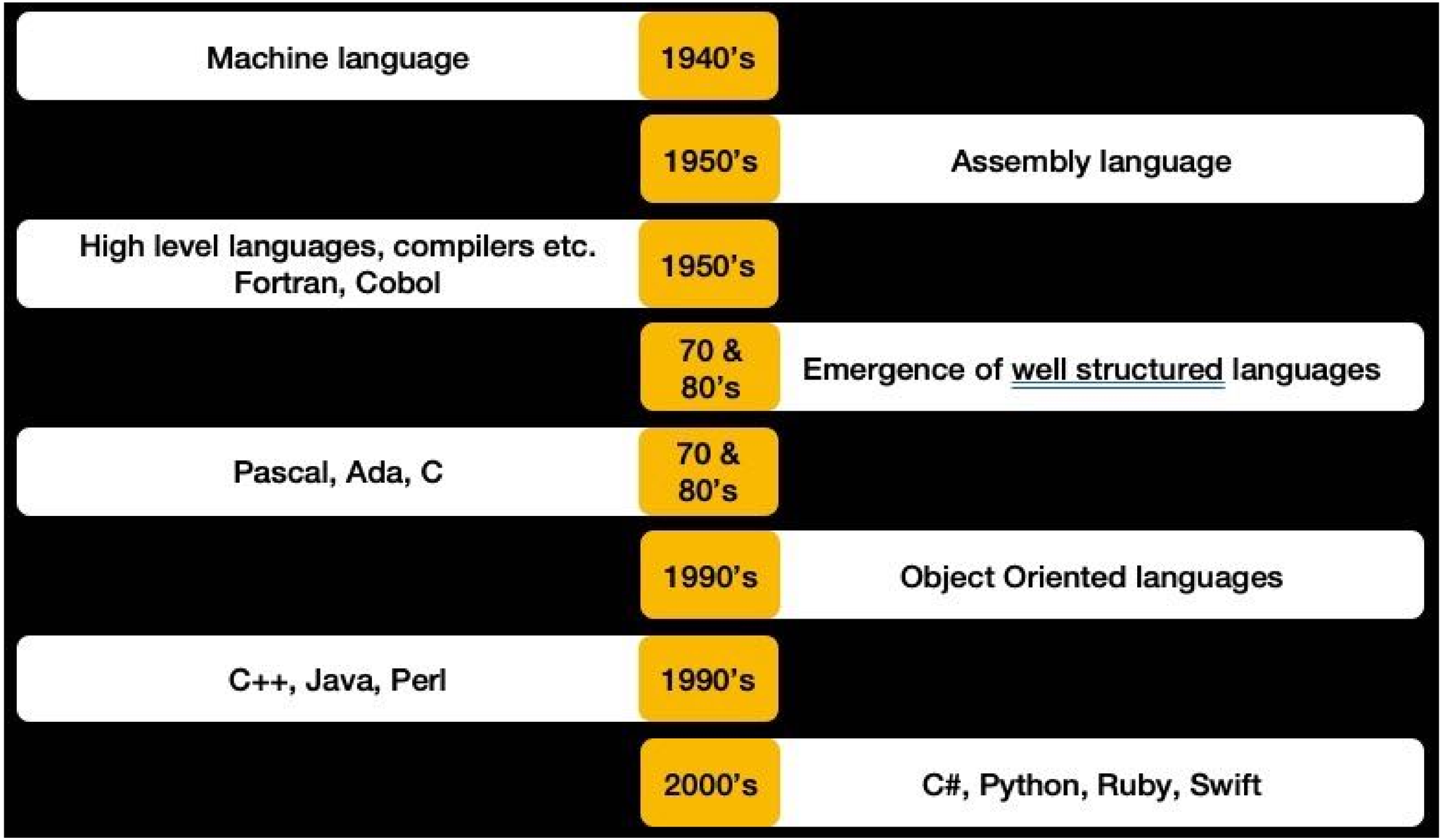
Computers run software

- Systems software:
  - Operating systems, network tools, software development tools (e.g. compilers)
- Application software:
  - Word processors, spreadsheets, databases, modelling and simulation.

Software

- A set of instructions for the computer;
- Usually written in a high level programming language (for humans to read);
- Computers understand one language, its machine language, not (usually) human readable;
- Software is translated into machine language (code) for execution.

Programming languages



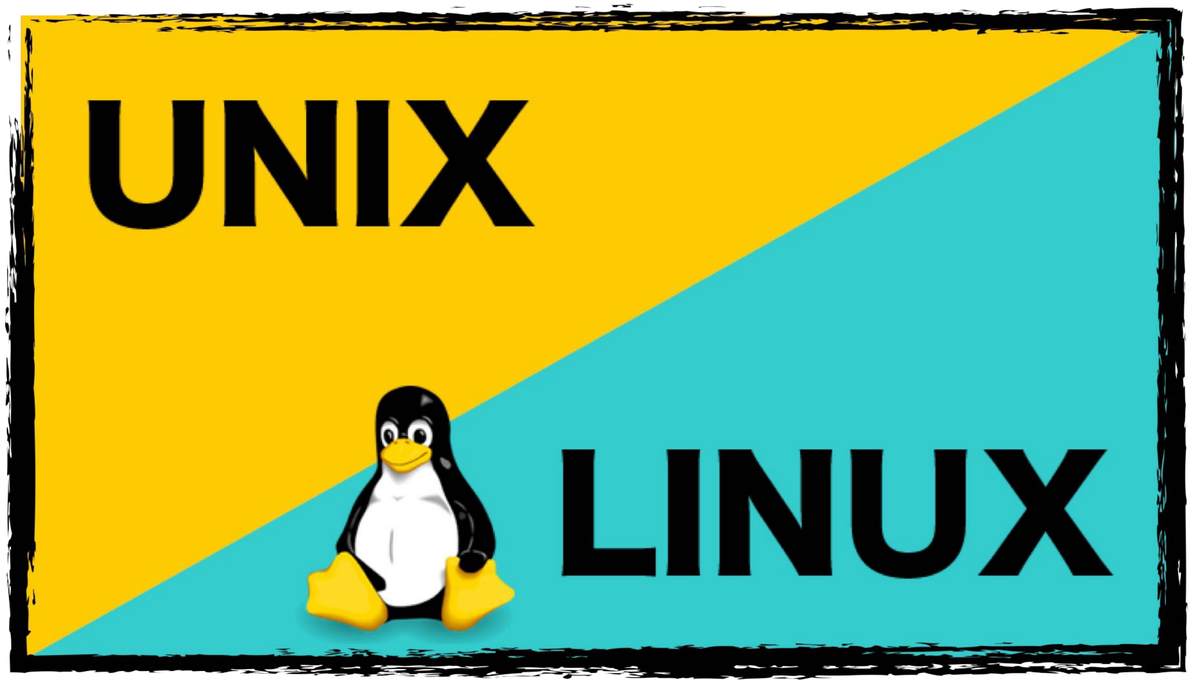
Data Representation in Computer Memory

- Computer memory: made of components that can be in one of two states:
  1. On;
  2. Off.
- Other binary information methods include:
  1. Dot and Dash (Morse);
  2. North and South (Magnet).
- Can be used in two ways:
  1. Represent actual values in base 2;
  2. Hold an arbitrary series of states coded with particular meanings.

UNIX

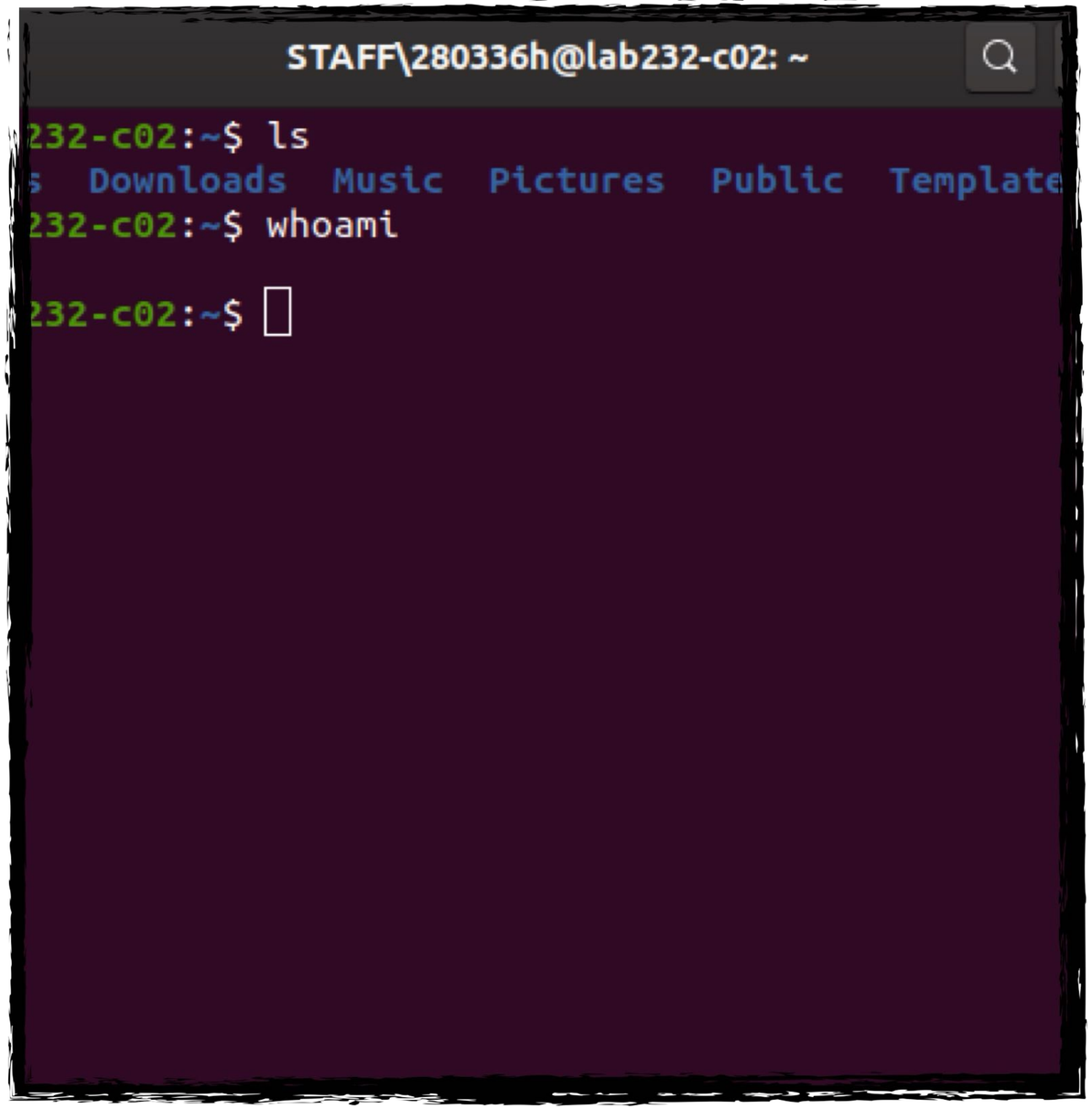
Introduction to UNIX

- In COMP1007 Linux is the operating system used;
- Linux is a UNIX variant;
- UNIX:
  - May be totally different to what you may be used to;
  - Is an operating system providing a huge scope in what can be accomplished but at a cost of a sometimes challenging user interface.
- macOS is a UNIX operating system;
- As future computing professionals familiarity with UNIX or UNIX-like environment is essential.



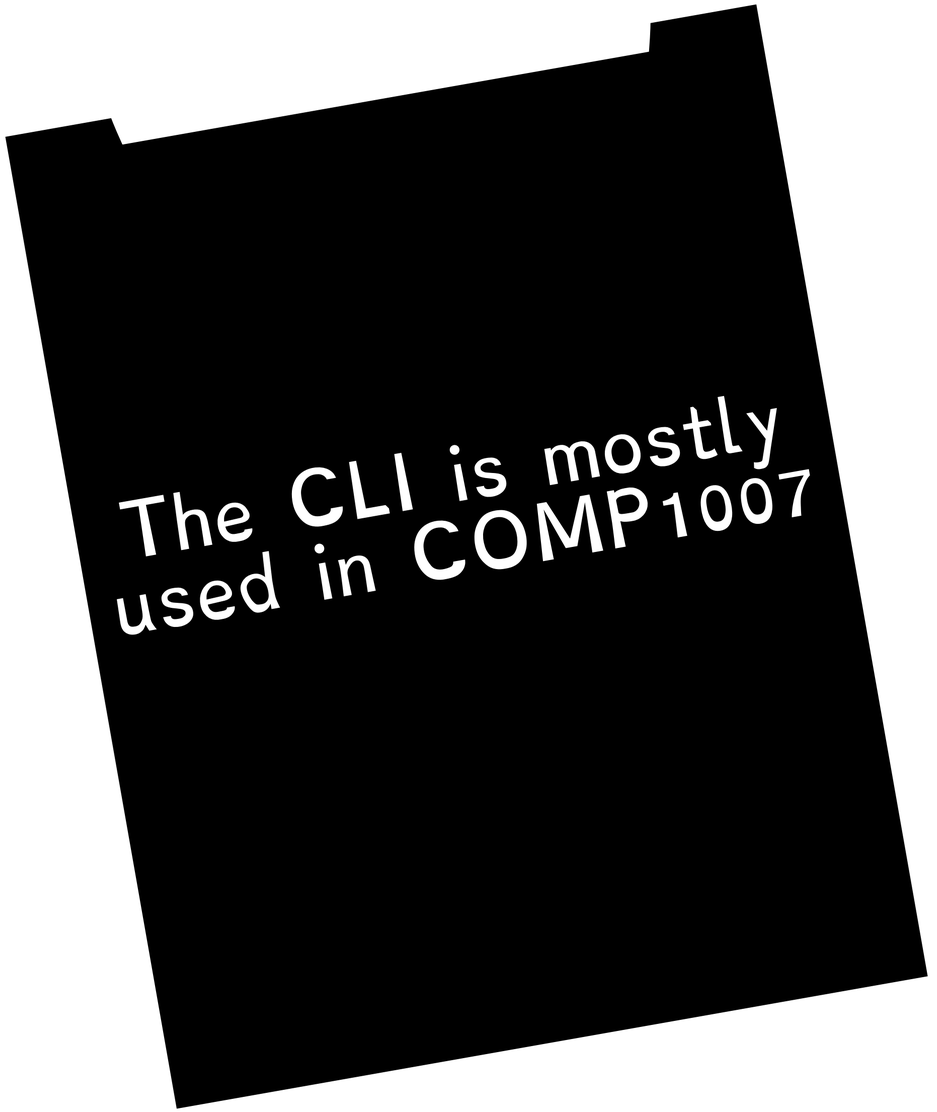
Command Line Interfaces (CLI)

- CLI is composed of:
  - a prompt signalling when the user can enter commands;
  - a command line on which the entered commands appear.



CLI vs GUI

- Graphical User Interfaces can be:
  - easy to learn how to use;
  - inefficient when dealing with repeated sets of the same functionality.
- Command Line Interfaces can:
  - require understanding of a number of concepts;
  - require familiarity with a command language;
  - be very fast;
  - be highly efficient, in dealing with repeated sets of the same functionality:
    - e.g. Multiple files that have a similar name or extension.



CLI vs GUI

- Graphical User Interfaces can be:
  - easy to learn how to use;
  - inefficient when dealing with repeated sets of the same functionality.
- Command Line Interfaces can:
  - require understanding of a number of concepts;
  - require familiarity with a command language;
  - be very fast;
  - be highly efficient, in dealing with repeated sets of the same functionality:
    - e.g. Multiple files that have a similar name or extension.

UNIX Shells

- In UNIX the Command Line Interface is known as a Shell;
- Allow us to perform commands on the Operating System as a user:
  - It is a communication medium between you and the hardware.

- We will use the **bash** shell:
  - Bourne Again Shell;
  - Covered in depth in Unix and C Programming (COMP1000).

Paths and Directory Structure

- Everything is a file in UNIX, it can be accessed with a direct path;
- Generally, we will only access files in our Home (~) directory:
  - the shortcut 'tilde' can be used to access home from anywhere.

```
dam@314lab:~$ cd ~/dam/Documents/PDI
```

- A direct path can also be use, specified from the root directory (/)

```
dam@314lab:~$ cd /home/dam/Documents/PDI
```

- Anytime you specify a path, make sure it is accurate, otherwise the OS does not know where to take you.

Common Bash (UNIX) Commands

- **cd** change directory: moves to the directory specified:

  dam@314lab:~$ cd Documents
- **cp** copy: copies a file or folder from source to destination

  dam@314lab:~$ cp Source.txt Destination.txt
- **mv** move: moves a file or folder from source to destination

  dam@314lab:~$ mv Source.txt Destination.txt
  - NB: The original file is MOVED

More Common Bash (UNIX) Commands

- **ls** Lists all (non hidden) files and folders in current directory:

  dam@314lab:~$ ls

  dam@314lab:~$ ls -al (will include the hidden files in the listing)

- **mkdir** make directory: Makes new folder as specified:

  dam@314lab:~$ mkdir PDI_practicals

- **rm** remove: removes the specified file

  dam@314lab:~$ rm AssignmentFile.txt

  dam@314lab:~$ rm -rf PDI_practicals (force removes directories (forever))

VIM

VI(M) – Visual Editor

- A text editor created for **UNIX**;
- It is primitively powerful, like nothing you've used before;
- Launch from the Terminal window;

```
dam@314lab:~$ vim
```

VI(M) – Visual Editor

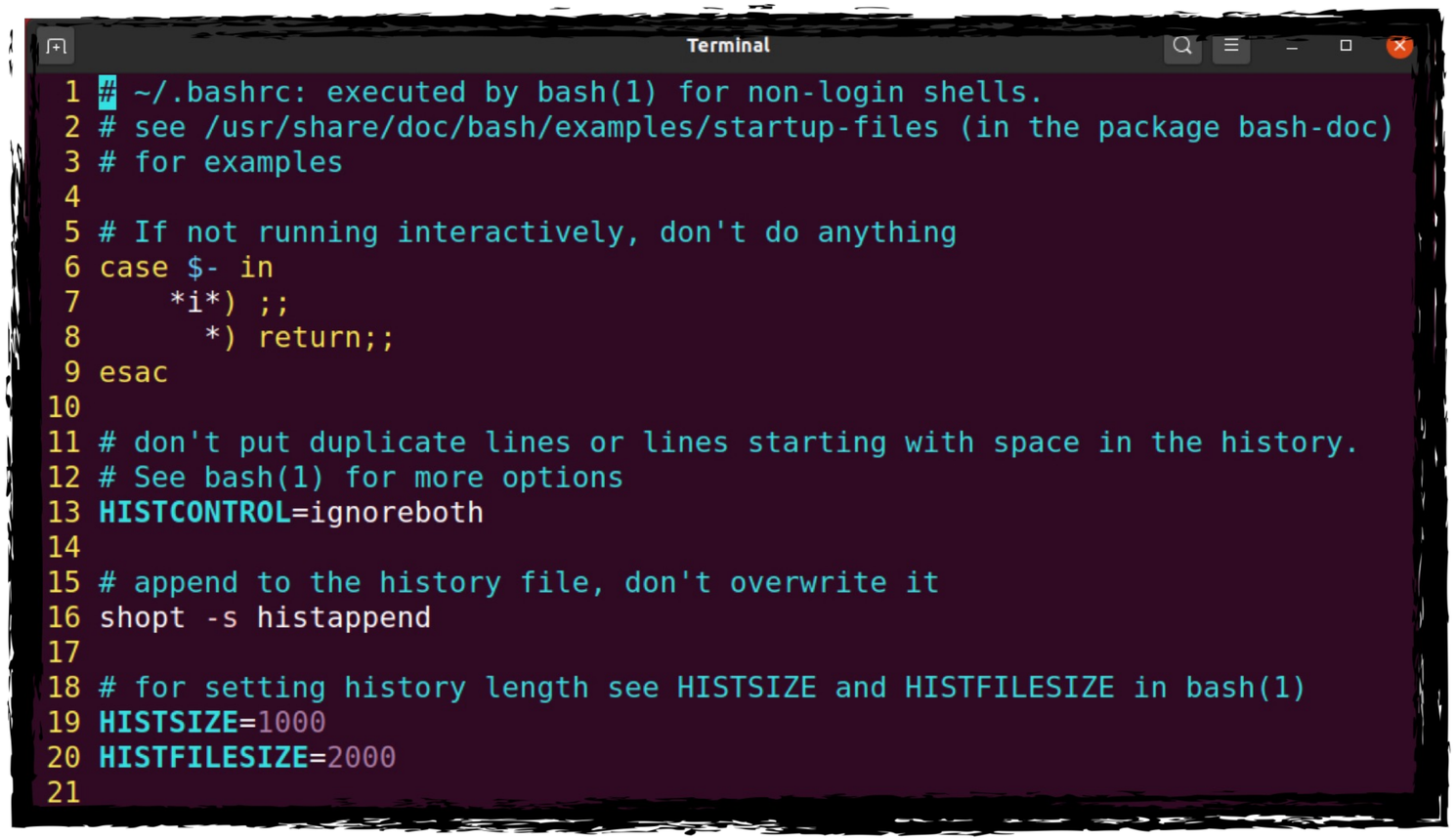
- A text editor created for UNIX;
- It is primitively powerful, like nothing you've used before;
- Launch from the Terminal window;

dam@314lab:~$ vim

Why use VIM?

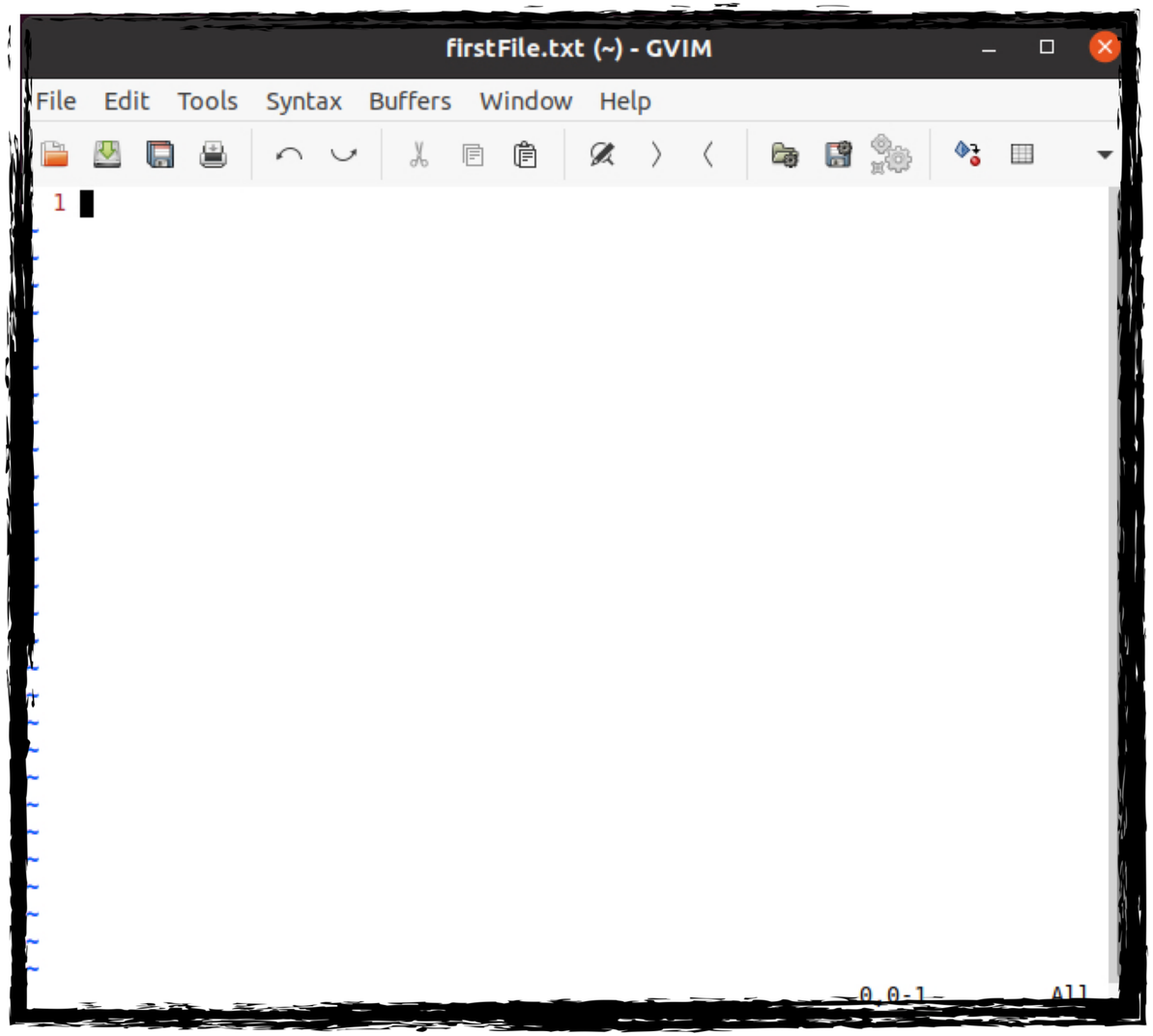
- Available on all UNIX and Linux systems;
- Allows easy edit access to system files from the Terminal.

dam@314lab:~$ vim .bashrc



gVIM: VIM with a GUI

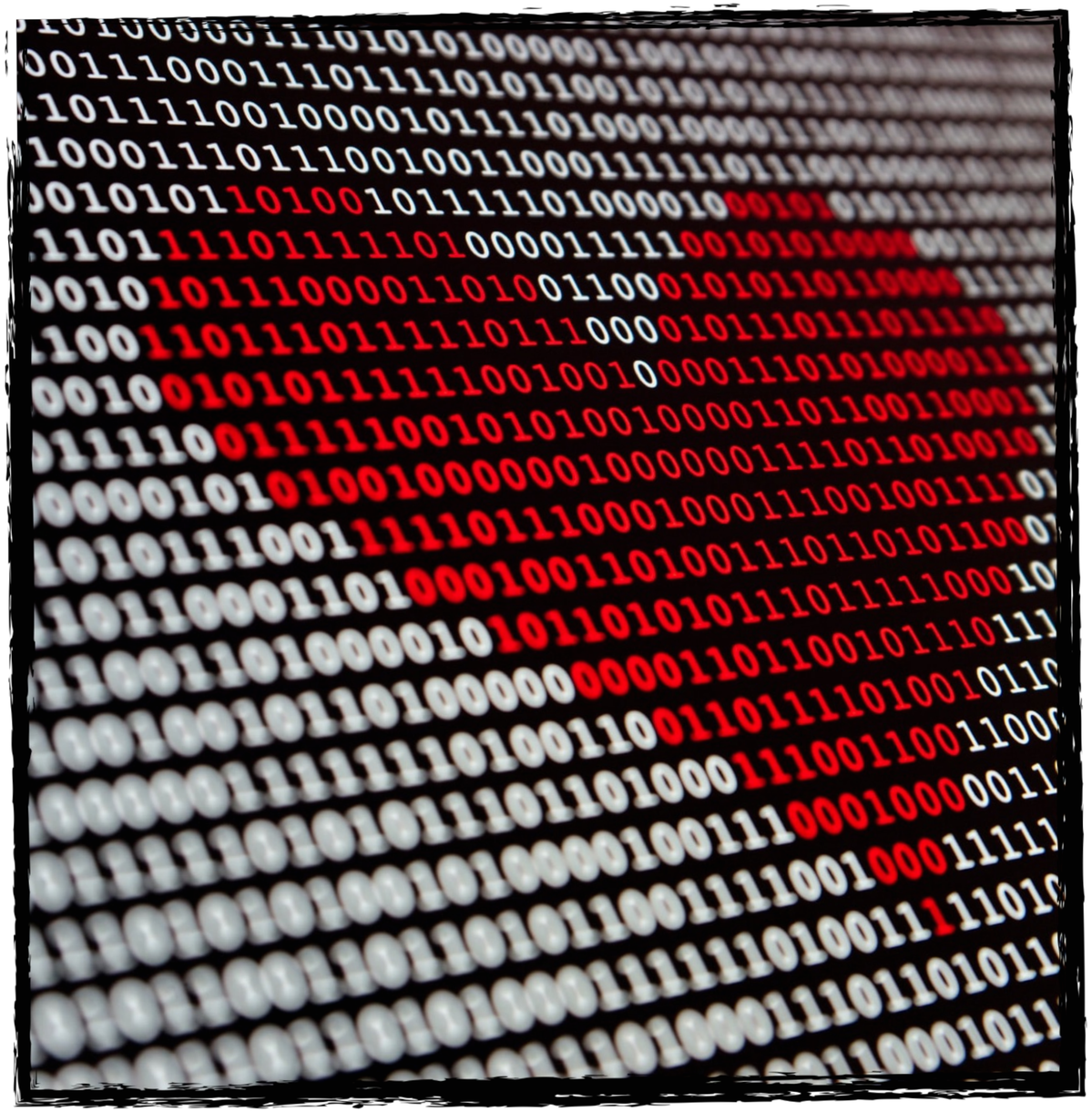
dam@314lab:~$ gvim firstFile.txt



VIM/gVIM has Two Modes

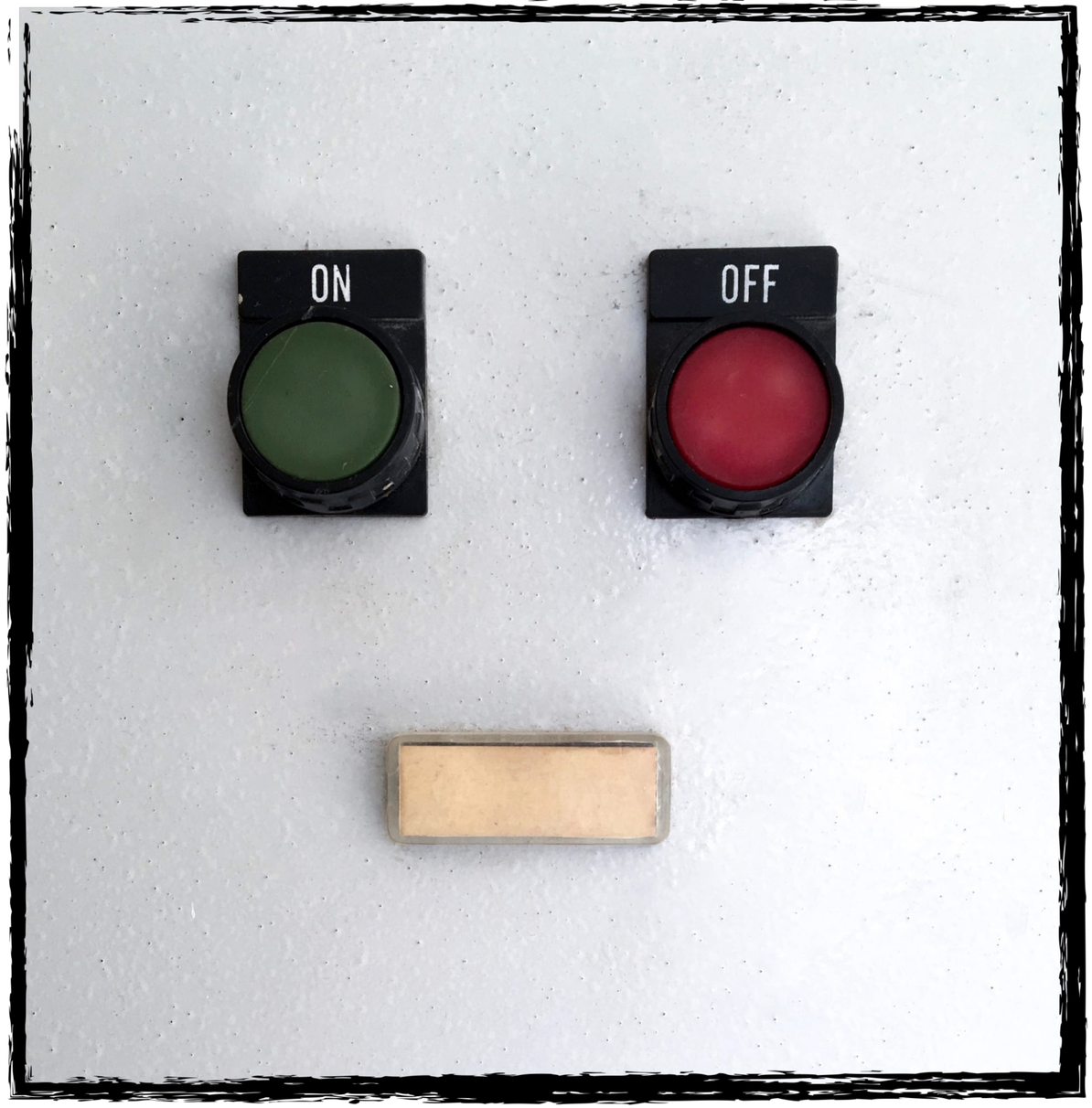
- Command Mode: causes actions to be executed on the file;
  - Default/Starting mode;
  - Each letter typed may be a command executed on the file.
- Insert Mode: edit the file's content;
  - Change to edit mode by pressing Esc followed by i key;
  - The file can now be edited.

Binary



Data Representation in Computer Memory

- Computer memory, made of components that can be in one of two states: on or off;
  - On and Off (Flashing light);
  - Dot and Dash (Morse); and
  - North and South (Magnet)
- Can be used in two ways:
  1. Represented actual values in base 2;
  2. Hold an arbitrary series of states coded with particular meanings.



Terminology

- Each Binary digit is called a bit;
- A group of eight (8) bits is a byte;
- Memory is broken up into wordsize storage locations;
- Wordsize: machine dependent and one or more bytes long:
  - Now 64 bits: 8 bytes.
- Each memory location is located through its memory address;
- All data and programs are stored in memory using various interpretations of these groups of 1's and 0's.

Data Types

- Manner of interpretation of the 1's and 0's varies for different data types stored:
- Interpretation of 1's and 0's depends on the data type being represented:
  - Address;
  - Instruction;
  - Integer value;
  - Real value;
  - Boolean;
  - Character:
    - Single character; or
    - Character String.

Integer Range

- Determined by how many distinct base2 values can be stored in the given number of bits:
  - Each additional bit doubles the size of the range.
- For N bits:
  - 1 bit required for the sign;
  - the remaining N−1 bits can represent 2^{N-1} different combinations, directly related to the binary value.
- Note: lack of symmetry is because of the need to represent zero (0) as one of the 2^{N-1} values:
  - {2^{N-1} negative, 0, $2^{N-1}-1$ positive} values
    - Negative values stored as the 2's compliment of the number.
- Attempting to store a number larger/smaller than the maximum/minimum value leads to Integer Overflow.

Decimal Number System

- We use the base-10, ‘Decimal’, number system;
- Digits used: 0 – 9 (0 1 2 3 4 5 6 7 8 9);
- 2546:

10^3	10^2	10^1	10^0
1000	100	10	1
2	5	4	6

$(2 \times 1000) + (5 \times 100) + (4 \times 10) + (6 \times 1) = 2546$

Computer Used Number System

- There are 10 types of people:
  - those who understand binary; and
  - those who don't.
- Correctly written: there are 10_2 types of people;
- Computers use base-2 (Binary);
- Often represented by base-8 (Octal) or base-16 (Hexadecimal);
- Easy to convert as they fall along the base-2 scale.

Converting Decimal to Binary

- Use the Quotient Remainder method;
- Divide number by 2: record the remainder 1 or remainder 0:
- 2546 =

2546 ÷ 2 = 1273	Remainder: 0	0
1273 ÷ 2 = 636	Remainder: 1	10
636 ÷ 2 = 318	Remainder: 0	010
318 ÷ 2 = 159	Remainder: 0	0010
159 ÷ 2 = 79	Remainder: 1	10010
79 ÷ 2 = 39	Remainder: 1	110010
39 ÷ 2 = 19	Remainder: 1	1110010
19 ÷ 2 = 9	Remainder: 1	11110010
9 ÷ 2 = 4	Remainder: 1	111110010
4 ÷ 2 = 2	Remainder: 0	0111110010
2 ÷ 2 = 1	Remainder: 0	00111110010
1 ÷ 2 = 0	Remainder: 1	100111110010

Convert Binary to Decimal (Double Check)

- 100111110010 =

2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
2048	**1025**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	0	1	1	1	1	1	0	0	1	0

2048 + 0 + 0 + 256 + 128 + 64 + 32 + 16 + 0 + 0 + 2 + 0 = 2546

Your Turn

- Convert 126_{10} to a Binary number;
- Convert 1101011_{2} to a Decimal number.
- Check your work by converting them back.

Octal & Hex

Octal

- Base 8;
- Digits used: 0 – 7 (0 1 2 3 4 5 6 7);
- 4762_8:

8^3	8^2	8^1	8^0
512	**64**	**8**	**1**
4	7	6	2

- Converting from Octal to Decimal:

$$(4 \times 512) + (7 \times 64) + (6 \times 8) + (2 \times 1) = 2546_{10}$$

$$4762_8 = 2546_{10}$$

Converting Decimal to Octal

- Use the Quotient Remainder method;
- Divide number by 8: record the remainder:
- 2546 =

2546 ÷ 8 = 318	Remainder: 2	2
318 ÷ 8 = 39	Remainder: 6	62
39 ÷ 8 = 4	Remainder: 7	762
4 ÷ 8 = 0	Remainder: 4	4762

Octal

- Extremely important for Unix file permission setting;
- Converting binary to octal, group the binary number in sets of 3, from right to left, then convert each group to an octal digit:
- $100111110010_2 = 4762_8$

100	111	110	010
4	7	6	2

- 342391_{10}: convert to binary, 001010011100101110111, then:

001	010	011	100	101	110	111
1	2	3	4	5	6	7

- $342391_{10} = 001010011100101110111_2 = 1234567_8$

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Hexadecimal (Hex)

- Base 16;
- Digits used: 0 1 2 3 4 5 6 7 8 9 A B C D E F;
- $9F2_{16}$:

16^2	16^1	16^0
256	16	1
9	F	2

- Convert Hex to Decimal:

$$(9 \times 256) + (F \times 16) + (2 \times 1) = 2546_{10}$$

$$(9 \times 256) + (15 \times 16) + (2 \times 1) = 2546_{10}$$

Hexadecimal

- Almost always used to display memory in computers;
- The Digits:

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hex	Binary
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Converting Hex to Binary

- Group the binary number in sets of 4, from right to left;
- Convert each group to hex digit (pad left with zeros if necessary);

  229656673502_{10} is $0011010101111000100110101011110011011110_2$

- Convert to Hex:

0011	0101	0111	1000	1001	1010	1011	1100	1101	1110
3	5	7	8	9	A	B	C	D	E

Signed Integers

Positive and Negative in memory

- The most significant bit (msb) determines the sign;
- MSB is the bit furtherest to the left in the number
  - 0 for positive and 1 for negative;
- Weight of each remaining bit is a power of two.
- For negative the weight is the negative of the corresponding power of two.
  - Calculated and stored using two's complement
- Must always know how many bits the number is stored in.
  - 8, 16, 32 or 64.

2's Complement

- To calculate 2's complement:
  - invert the bits using the bitwise **NOT** operation (flip them);
  - add 1 to the resulting value.
- **85_{10}:**

8 bit	
Positive Binary:	01010101
Flip the Bits:	10101010
	+1
Negative Binary:	10101011
16 bit	
Positive Binary:	0000000001010101
Flip the Bits:	1111111110101010
	+1
Negative Binary:	1111111110101011

Another 2's Complement

- **80_{10}:**

8 bit		
	Positive Binary:	01010000
	Flip the Bits:	10101111
		+1
	Negative Binary:	10110000
16 bit		
	Positive Binary:	0000000001010000
	Flip the Bits:	1111111110101111
		+1
	Negative Binary:	1111111110110000

Real Numbers

Real Numbers

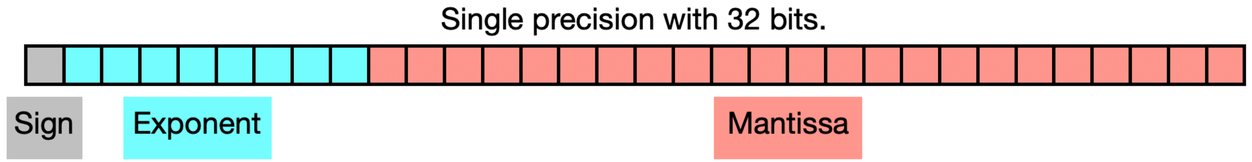
- Positive or negative value consisting of a whole number plus a fractional part (expressed in floating point, or scientific notation);
- The ‘float’ and ‘double’ data types, are an abstraction of the real numbers that exist in the mathematical world;
- The range and accuracy of real numbers are limited in any computing system;
- Why? How would you store or ?

Range and Accuracy of Real Numbers

- Determined by number of bits and the split up of the mantissa and exponent;
- Range has to be limited, by definition, an infinite number of bits needed to represent infinity (∞)
- Accuracy is therefore limited:
  - Number of significant digits is limited:
    - There are an infinite number of real values between any two points on the number line.
  - Irrational numbers;
  - Recurring decimals;
  - IEEE 754 form (binary conversion).

IEEE 754 (Floating Point) Numbers

- Comprises of sign, exponent and mantissa:
  - Mantissa A.K.A Significand.
- Single precision – binary32:
  - Sign bit: Most Significant bit; 0 positive, 1 negative;
  - Exponent width: 8 bits biased to 127;
  - Mantissa (significand) precision: 24 bits (23 explicitly stored)



Real Conversion

(refer to the IEEE 754 standard for strict conversion including rounding behaviour)

- To convert a base 10 real number into an IEEE 754 binary32 format use the following outline:
- A real number with an integer and a fraction part 12.375;
  - Determine the sign bit, positive, hence: 0
- Convert the integer part into binary:
  - 12 = 1100
- Convert the fraction part into binary:
  - 0.375 = .011 (RELAX: explained in the following slides)

Fraction Conversion

- Multiply the fraction by 2
- If value >= 1:
  - write a 1 to the mantissa, then
  - subtract 1 from the value;
- Otherwise:
  - write 0 to the mantissa.
- Repeat the preceding steps for the appropriate number significant bits:
  - stop if the value is 0 (after subtracting 1)
  - when the required number of significant bits is reached.

Fraction Conversion (2)

- `0.375`

Calculate	Result	Mantissa
2 x 0.375	0.75	0
2 x 0.75	1.5	1
1.5 - 1	0.5	- -
2 x 0.5	1.0	1
1.0 - 1.0	0	Done

`.011`

Conversion

- Add the two results:
  - `1100.011`
- Normalise them:
  - Move decimal point until it is right of significant bit;
  - `1.100011`
  - Moved 3 places, used for the base 2 exponent;
  - 1.100011×2^3

Conversion (2)

- Exponent based on precision (`127` for single bit precision):
  - `127 + 3 = 130`
  - Convert to binary `10000010`
- Grab the mantissa
  - `100011` (Integer part (always `1`) is not stored)
- Final encoding is (exact in this case):