A brief guide to how computers encode data

- Some useful background for handling files (especially for sequence bioinformatics)

Basic bits and pieces

- Bits
  - 1 bit = 1 binary digit, i.e. 2 possible values; think of these as:
    - ‘on’ or ‘off’
    - ‘set’ or ‘clear’
    - 0 or 1

- Bytes

  1 byte = 8 bits

00000000	0 in decimal
00000001	1
00000010	2
00000011	3
…	
…	
11111101	253
11111110	254
11111111	255

- 2^8 = 256 different values can be stored: i.e. 0 to $2^8 - 1$; 0 … 255

More on bits

- Terms like '32-bit' and '64-bit' refer to the number of bits that a computer processor can handle for various very basic operations
  - Including addressing **memory** locations, etc
- It can be important to be aware of this when obtaining and installing software (including OS)
  - The processor architecture known as 'x86' is strictly speaking 32-bit
  - x86-64, commonly known as 'x64', is 64-bit
- More on this in a later session

Back to bytes

- File size considered in terms of numbers of bytes (or Kilobytes, Megabytes, Gigabytes etc)

  kb **Mb** **Gb**

- Strictly speaking, 1kb = 1,024 bytes (not 1,000 bytes)
  - Because 1,024 is a round number in base 2 (= 2^{10})

- A collection of bytes can be used to mean just about anything…
- … that the computer program reading/writing them wants – it's all about context

One very common and useful approach is to use each byte to store **one 'character'**

Thus…..

THE PLAIN TEXT FILE

A bedrock standard

- In a long established and globally used code, each of the values from:
  - 0 to 127
  - i.e. byte values 00000000 to 01111111
- Is used to represent a particular *character* (letter, digit, symbol) or *control code*
- This is the **ASCII** standard

Example ASCII codes: printable characters

Decimal ASCII no.	Character	name
32		(space)
33	!	
47	/	
62	>	
65	A	
66	B	
67	C	
71	G	
97	a	
116	t	
117	u	

- The fact that only the characters with ASCII codes 33 and above are 'visible' has direct implications for some specialist uses of plain text files.
- Of particular interest to us is the use of a single symbol to denote the **Quality Score** associated with a single nucleotide in a sequence read. This will be covered later on.

What about ‘control codes’?

- ASCII codes 0 to 31, and 127, designate non-printable ‘control characters’
- Invisible basically, but some have visible (and audible!) effects

Decimal ASCII no.	Formal name
1	Start of Heading
3	End of Text
4	End of Transmission
7	Bell
8	Backspace
9	Horizontal Tab
10	Line Feed
13	Carriage Return
26	Substitute Character
127	Delete

- Do we really need to know about these control codes?
- Well, a little knowledge of these codes is certainly useful.
- Unlike the printable characters, these codes are ***not always*** used in exactly the same way…
- **<u>…. By different Operating Systems.</u>**

Doing bioinformatics in a multi-platform environment

- Many of these can be generated - by simultaneously:
  - holding down the Control key (usually marked ‘Ctrl’ on the keyboard) and
  - Pressing a key ; for example:
    - Ctrl+A generates control code 1
    - Ctrl+G generates control code 7
    - Ctrl+I generates control code 9 …. etc
- But some control codes are generated by a dedicated key on the keyboard

Decimal ASCII no.	Formal name	Key(s)	Usage in the UNIX/Linux shell context
1	Start of Heading	Ctrl+A (or 'Home')	Move cursor to
3	End of Text	Ctrl+C	Interrupts a process
4	End of Transmission	Ctrl+D	Sort of what it says – but a bit involved
7	Bell	Ctrl+G	sounds a bell!
8	Backspace	Ctrl+H (or 'Backspace')	Like it says
9	Horizontal Tab	Ctrl+I (or 'Tab')	Tab character OR autocomplete
10	Line Feed	Ctrl+J	In shell, behaves like Ctrl-M
13	Carriage Return	Ctrl+M (or 'Return')	See later slide…
26	Substitute Character	Ctrl+Z	Suspends a process
127	Delete	'Delete'	

So – what about sequence data files?

- Most biological sequence data file formats you will encounter (but some are not)
- **Are simply plain-text format**
- Despite the simple sequence 'alphabets'
- Because, other non-sequence information is nearly always necessary:
  - Sequence name/identifier
  - Other annotation
  - (can be very extensive)
- So, most sequence data files consist of standard letters, numbers, symbols…….and newline characters

Newlines

- Have you ever had a file which looks like this:

```
>sp|A7ZCN3|RL11_CAMC1 50S ribosomal protein L11 OS=Campylobacter concisus (strain 13826) GN=rplK PE=3 SV=1
MAKKVIGEIKLQIAATKANPSPPVGPALGQKGVNIMEFCKAFNEKTKDMVGFNIPVVITV
YADKSFTFITKQPPATDLIKKAAGITKGTDNPLKNKVGKLTKAQVLEIVEKKLVDLNTND
KEQAAKIIAGSARSMGVEVVD
>sp|A7GZK3|RL11_CAMC5 50S ribosomal protein L11 OS=Campylobacter curvus (strain 525.92) GN=rplK PE=3 SV=1
MAKKVIGEIKLQIAATKANPSPPVGPALGQKGVNIMEFCKAFNEKTKDMVGFNIPVVITV
YADKSFTFITKQPPATDLIKKAAGIAKGTDNPLKNKVGKLTKAQVLEIVEKKLVDLNTND
KEQAAKIIAGSARSMGVEIVD
>sp|A0RQI9|RL11_CAMFF 50S ribosomal protein L11 OS=Campylobacter fetus subsp. fetus (strain 82-40) GN=rplK PE=3 SV=1
MAKKVVGEIKLQIAATKANPSPPVGPALGQQGVNIMEFCKAFNERTKDMAGYNIPVVITV
YADKSFTFITKQPPATDLIKKAAGITKGADNPLKNKVGQLTKAQILEIVDKKIVDMNTKD
KEQAAKIISGSARSMGITVVD
>sp|A7I3U3|RL11_CAMHC 50S ribosomal protein L11 OS=Campylobacter hominis (strain ATCC BAA-381 / LMG 19568 / NCTC 13146 / CH001A
GN=rplK PE=3 SV=1
MAKKVIGEIKLQIAATKANPSPPVGPALGQKGVNIMEFCKAFNEKTKGMEGFNIPVIITV
YADRSFTFITKQPPATDLIKKTAGVQKGSDNPLKNKVGKLTKAQVLEIVEKKMADLNTKD
KEQAARIIAGSARSMGITVE
>sp|A8FKQ9|RL11_CAMJ8 50S ribosomal protein L11 OS=Campylobacter jejuni subsp. jejuni serotype O:6 (strain 81116 / NCTC 11828)
GN=rplK PE=3 SV=1
MAKKVVGEIKLQIAATKANPSPPVGPALGQQGVNIMEFCKAFNERTKDMAGFNIPVVITV
YADKSFTFITKQPPATDLIKKAAGISKGTDNPLKNKVGKLTRAQVLEIVDKKIADLNTKD
RDQAAKIIAGSARSMGVEIVD
>sp|A7H4Q9|RL11_CAMJD 50S ribosomal protein L11 OS=Campylobacter jejuni subsp. doylei (strain ATCC BAA-1458 / RM4099 / 269.97)
GN=rplK PE=3 SV=1
MAKKVVGEIKLQIAATKANPSPPVGPALGQQGVNIMEFCKAFNERTKDMAGFNIPVVITV
YADKSFTFITKQPPATDLIKKAAGISKGTDNPLKNKVGKLTRAQVLEIVDKKIADLNTKD
RDQAAKIIAGSARSMGVEIVD
>sp|Q9PI35|RL11_CAMJE 50S ribosomal protein L11 OS=Campylobacter jejuni subsp. jejuni serotype O:2 (strain ATCC 700819 / NCTC
```

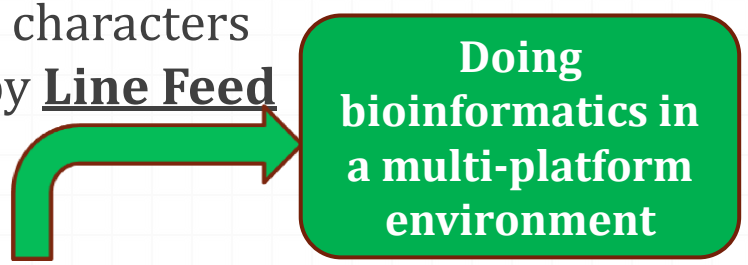
- ...end up looking like this
- - when you view it in Windows?

```
>sp|A7ZCN3|RL11_CAMC1 50S ribosomal protein L11 OS=Campylobacter concisus (strain 13826) GN=rplK PE=3 SV=1MAKKVIGEIKL
DLNTKDKEQAARIIAGSARSMGITVE>sp|A8FKQ9|RL11_CAMJ8 50S ribosomal protein L11 OS=Campylobacter jejuni subsp. jejuni seroty
plK PE=3 SV=1MAKKVVGEIKLQIAATKANPSPPVGPALGQQGVNIMEFCKAFNERTKDMAGFNIPVVITVYADKSFTFITKQPPATDLIKKAAGISKGTDNPLKNKVGKLTRAQ
```

- ...or end up looking like this
- - when you view it in Linux or OSX?

```
>sp|A7ZCN3|RL11_CAMC1 50S ribosomal protein L11 OS=Campylobacter concisus (strain 13826) GN=rplK PE=3 SV=1

MAKKVIGEIKLQIAATKANPSPPVGPALGQKGVNIMEFCKAFNEKTKDMVGFNIPVVITV

YADKSFTFITKQPPATDLIKKAAGITKGTDNPLKNKVGKLTKAQVLEIVEKKLVDLNTND

KEQAAKIIAGSARSMGVEVVD

>sp|A7GZK3|RL11_CAMC5 50S ribosomal protein L11 OS=Campylobacter curvus (strain 525.92) GN=rplK PE=3 SV=1

MAKKVIGEIKLQIAATKANPSPPVGPALGQKGVNIMEFCKAFNEKTKDMVGFNIPVVITV

YADKSFTFITKQPPATDLIKKAAGIAKGTDNPLKNKVGKLTKAQVLEIVEKKLVDLNTND

KEQAAKIIAGSARSMGVEIVD

>sp|A0RQI9|RL11_CAMFF 50S ribosomal protein L11 OS=Campylobacter fetus subsp. fetus (strain 82-40) GN=rplK PE=3 SV=1

MAKKVVGEIKLQIAATKANPSPPVGPALGQQGVNIMEFCKAFNERTKDMAGYNIPVVITV

YADKSFTFITKQPPATDLIKKAAGITKGADNPLKNKVGQLTKAQILEIVDKKIVDMNTKD

KEQAAKIISGSARSMGITVVD

>sp|A7I3U3|RL11_CAMHC 50S ribosomal protein L11 OS=Campylobacter hominis (strain ATCC BAA-381 / LMG 19568 / NCTC 13146 / CH001A)
GN=rplK PE=3 SV=1
```

- In UNIX and UNIX-like operating systems
  - (such as Linux and OSX)
  - a newline is signified by a single control character
  - namely the **Line Feed** character (ASCII 10)
- In MicroSoft DOS/Windows
  - a newline is signified by two control characters
  - namely **Carriage Return** followed by **Line Feed**
  - (ASCII 13 then ASCII 10)
- Can cause much annoyance
- And can cause software reading sequence files to get confused
- The **`dos2unix`** utility is your friend

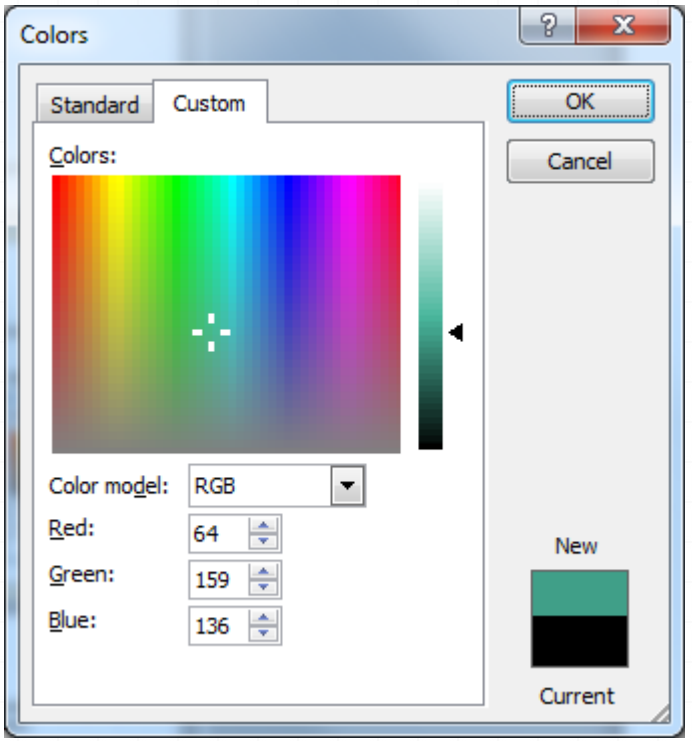


- Many configuration files in a Linux operating system are plain text files
- Some bioinformatics software may have configuration files which you will need to be aware of
  - And possibly edit
  - These are plain text files
- **Scripts** are plain text files
- **Source code** files are plain text files
- Scripts and source code are human readable-writeable, and readable by a program which interprets and runs them, and/or **compiles** them

Nybbles and Hex (a colourful interlude?)

- Very occasionally you may encounter a 1-byte number expressed as two 4-bit numbers
- A 4-bit number is called a 'nybble'
  - And can have a value between 0 and 15 (0000 and 1111)
- A nybble can also be expressed as a hexadecimal ('hex') digit
- Hex digits are:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- In some circumstances, bytes are usefully expressed as a 2-digit hex number
- E.g. the hex number **35** = 3 x 16 + 5 = 53 in decimal;
- **A4** = 10 x 16 + 4 = 164 decimal
- **FF** = 15 x 16 + 15 = 255 decimal

- Sometimes you may see e.g. colours expressed as 6-digit hex (3-byte) numbers,
- E.g. **#409F88**
- Amount of…
  - **Red: Hex 40 = decimal 64**
  - **Green: Hex 9F = decimal 159**
  - **Blue: Hex 88 = decimal 136**
- (often this format is used in HTML for example)



Beyond standard ASCII

- If the standard ASCII characters cover codes 0 to 127
  - What about the other possible values of a byte?
  - i.e. 128-255 ?
- There are various ‘extended ASCII’ encodings
  - which all use the standard ASCII for 0-127
  - and their own particular uses for 128-255
    - Such as latin letters with diacritical marks
      - E.g. ISO/IEC 8859-1
      - E.g. Windows-1252
    - Cyrillic letters, e.g. ISO/IEC 8859-5

Then there is **Unicode**

- Supports many different alphabets and sets of symbols
- Different character encoding sets are defined within the Unicode standards
- Beyond the scope of this session/slideshow

What's not a plain text file?

- Files which are not intended to be treated as plain text are usually referred to as binary files
- If these binary files are sets of instructions that can be executed directly by the processor
  - are referred to as **(binary) executables**
  - and these are one type of 'program'
  - (**scripts** are are type of non-binary program)
- Compressed data, including compressed plain text files
  - **BAM** is a binary, compressed version of the (plain-text) **SAM** sequence format
- Any other data that's not a plain text file

What's not a plain text file?

- Word documents
  - These are binary data files
  - The binary data encodes not just the text, but how it is displayed – font types, sizes, styles, colours, margin dimensions, metadata such as author etc etc
- Some alternatives to detailed formatting of text (and images) specify all that information in a format which ***is itself*** plain text; e.g.
  - HyperText Markup Language HTML
  - Rich Text Format (RTF)