SCHOOL OF
PHYSICS & ASTRONOMY

The Unix Shell
(Slides used in the workshop)

David McKain

Software Carpentry Workshop - 4th & 11th February 2020

Lesson links

Please keep open in your web browser:

1. Shared info to cut & paste during today's session: tinyurl.com/swc-feb-2020
2. The **Unix Shell** Software Carpentry lesson: http://swcarpentry.github.io/shell-novice/

Notes & hints:

- Link (2) can be accessed from (1)
- Keep the above pages open in browser tabs
- I'll regularly sync the material - put your hand up if lost!

Welcome!

- Based on **The Unix Shell** Software Carpentry lesson
- Goals:
  - Explain what Unix is why you'd want/need to use it
  - Get experience with some of the most common Unix commands
  - Get comfortable finding your way around your files on Unix systems
  - Teach you enough to be able to do cool stuff (e.g. use Eddie / supercomputers...)
  - Show you that the Unix Shell is less scary than it might seem!
  - Learn a bit about Unix Philosophy

Session outline

We'll do a mix of:

- Short expositional talks
- Live examples
- Exercises & feedback
- Random nonsense

Topics

Week 1:

- Introducing Unix & Shell
- Navigating Files & Directories
- Working with Files & Directories
- Handy Unix commands
- Pipes & Filters

Week 2:

- Loops & Variables
- Shell Scripts
- Finding Things

Preparation

Preparation

Open the **Setup** page in the lesson and:

1. Download and extract the sample data ZIP as directed
2. Make sure you can open a shell
   - Mac & Linux: Use the **Terminal** application
   - Windows: Run **Git Bash**

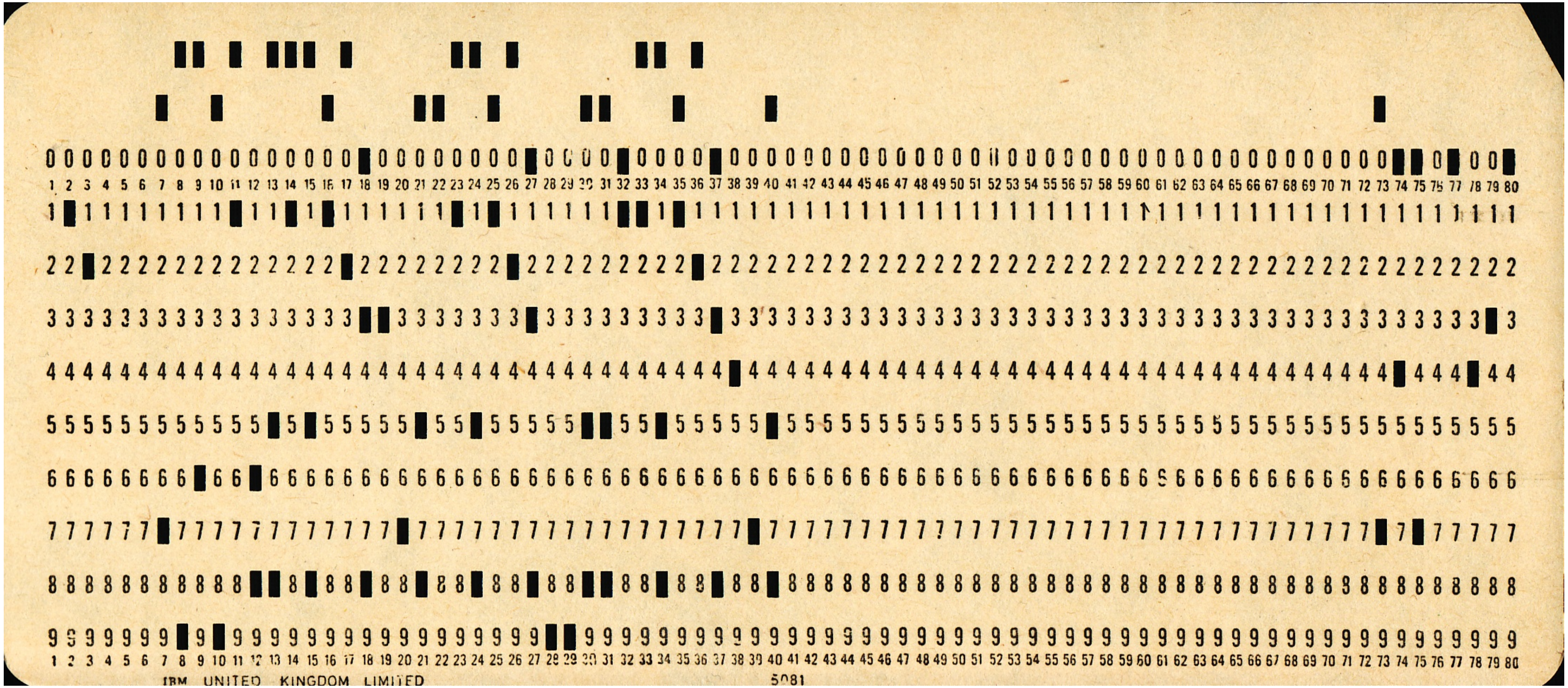
1. Introducing Unix & Shell

Introduction

Computers do 4 basic things:

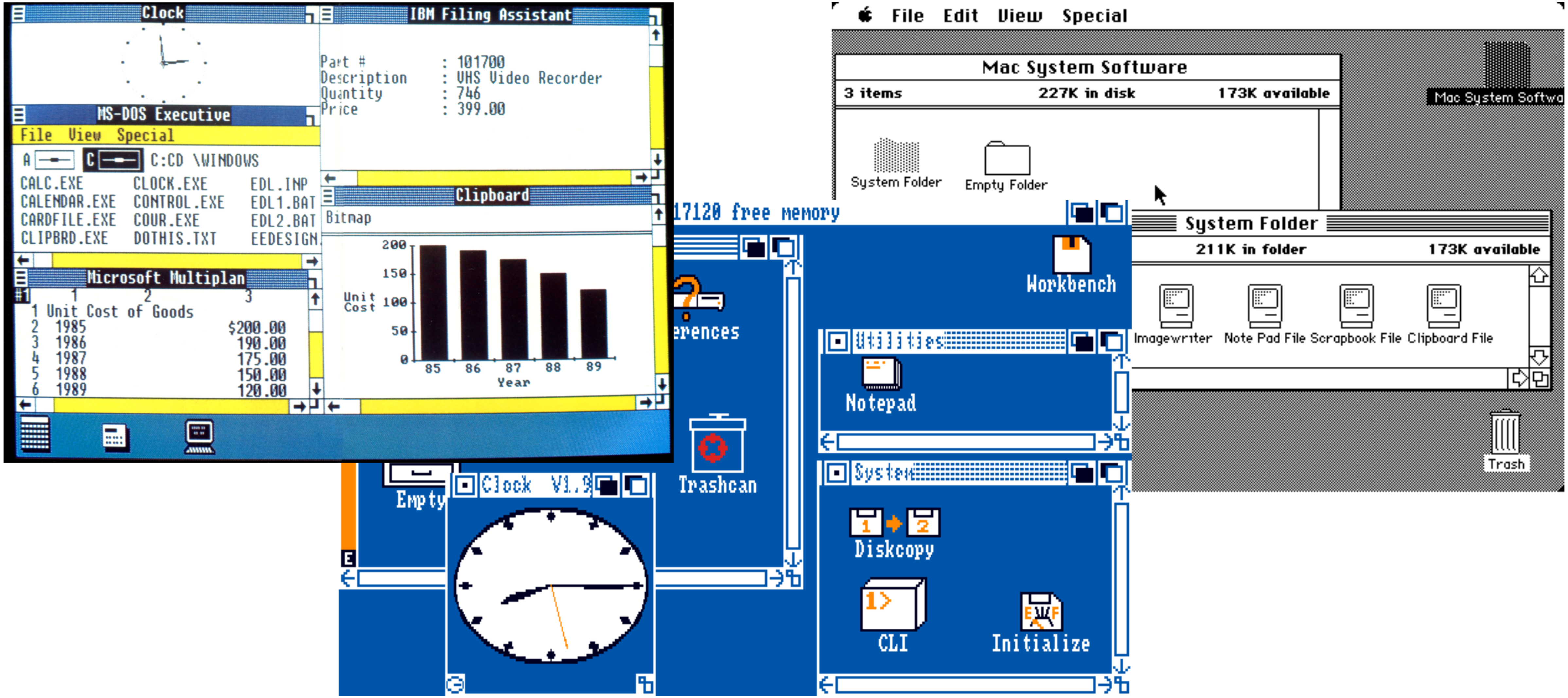
- Run programs
- Store data
- Communicate with each other
- **Interact with us**

Interaction in the 1960s: punched cards



1970s: Command Line Interfaces (CLI)

1980s: Graphical User Interfaces (GUI)



```
Part #          : 101700
Description     : VHS Video Recorder
Quantity        : 746
Price           : 399.00
```

#1	1	2	3
1	Unit Cost of Goods		
2	1985		$200.00
3	1986		190.00
4	1987		175.00
5	1988		150.00
6	1989		120.00

```
A     C    C:CD \WINDOWS
CALC.EXE       CLOCK.EXE      EDL.INP
CALENDAR.EXE   CONTROL.EXE    EDL1.BAT
CARDFILE.EXE   COUR.EXE       EDL2.BAT
CLIPBRD.EXE    DOTHIS.TXT     EEDESIGN
```

CLI vs GUI

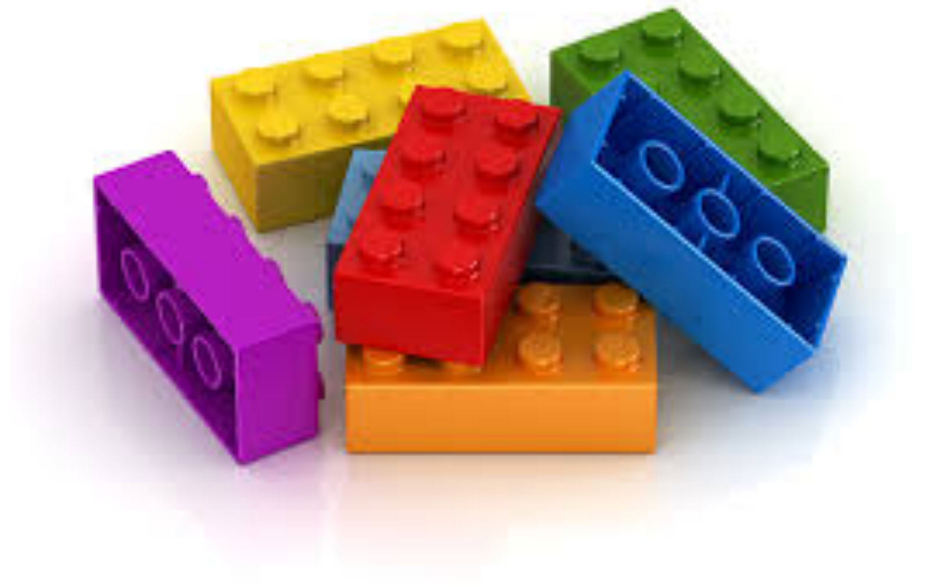
- Graphical interfaces:
  - Easier to pick up at first
  - But can become limiting & repetitive
- Command line interfaces:
  - Harder to learn at first
  - But becomes very efficient & versatile
- Command line interfaces haven’t been killed by GUI!

What is Unix?

- A family of Operating Systems (c.f. Windows)
- Originally developed in the 1970s
- Historically:
  - Operating Systems for big and expensive computers...
  - ...usually used by lots of different people at once
- Modern Unix systems:
  - Still big computers, e.g. Eddie, most supercomputers.
  - But also Apple Mac, Linux computers, Android phones (sort of)

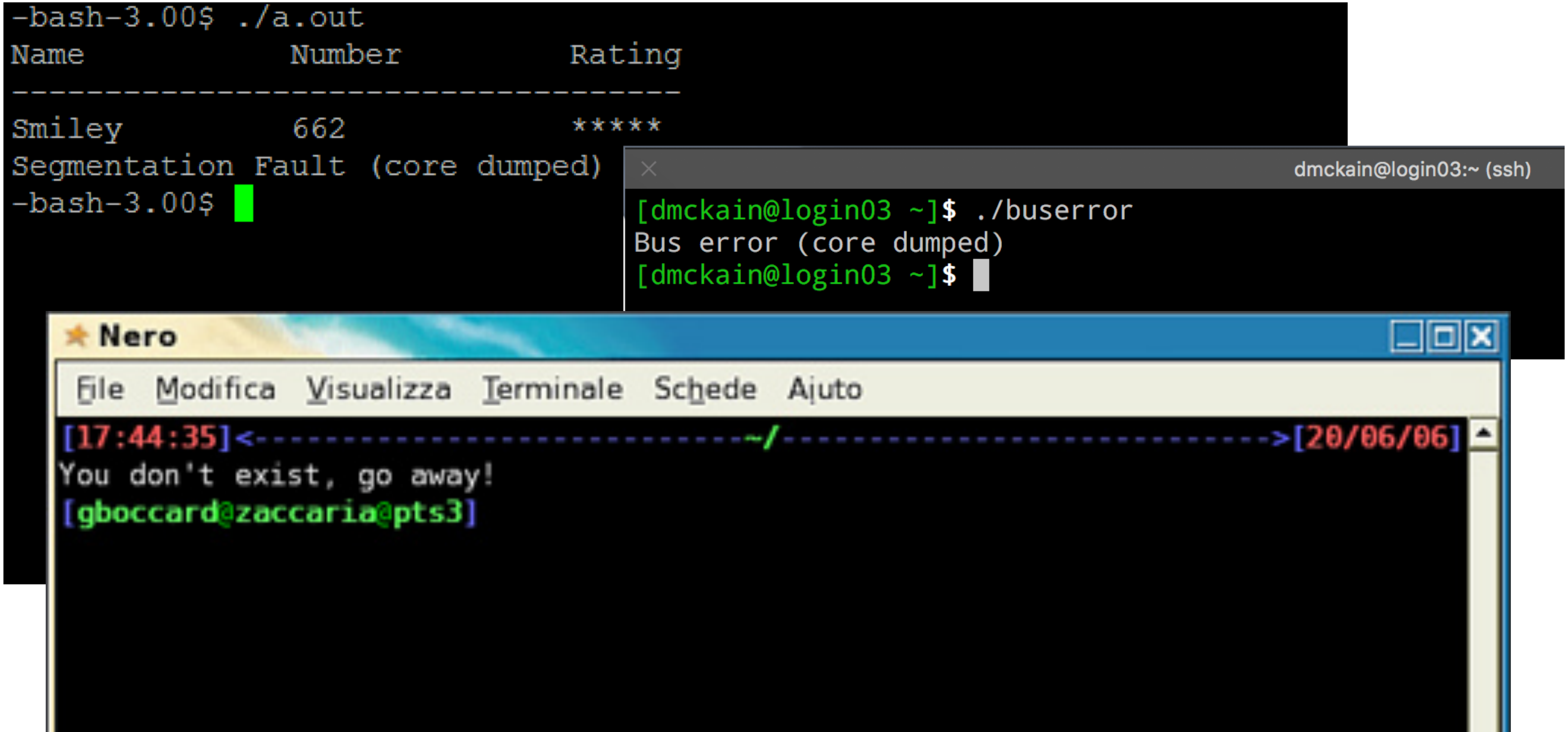
What is Unix?

- Unix philosophy:
  - Modular design
  - Lots of small tools doing "one job and doing it well”
  - Joining things up (pipelining)
  - The importance of textual data
  - Choice - lots of ways to do the same thing
- Unix is fun?!
  - Terse & cryptic commands
  - Terrible humour
  - Religious wars
  - Whimsical/scary error messages



Religious wars…

Some lovely Unix error messages



```
-bash-3.00$ ./a.out
Name             Number           Rating
----------------------------------------
Smiley           662              *****
Segmentation Fault (core dumped)
-bash-3.00$ 
```

```
[dmckain@login03 ~]$ ./buserror
Bus error (core dumped)
[dmckain@login03 ~]$ 
```

```
[17:44:35]<---------------------------~/--------------------------->[20/06/06]
You don't exist, go away!
[gboccard@zaccaria@pts3]
```

What is the Unix Shell?

- The **Unix Shell** is a CLI for communicating with Unix systems
- (Unix systems do also have GUIs)
- Actually there are lots of different shells available for Unix!
- The shell we'll be learning is called **bash** (**B**ourne **A**gain **Sh**ell... ha!)
- Bash tends to be the default shell on most Unix systems
- Some people prefer to use other shells...

Why learn/use the Unix Shell?

- Lets you interact with pretty much any Unix system in a uniform way
  - Helps make stuff portable
- Sometimes it's the only way you can interact with a Unix system!
- Pretty much essential for using a supercomputer

Why learn/use the Unix Shell?

- As a researcher, knowing a bit of shell can help with:
  - Getting your data & code from A to B
  - Checking & reporting on your data
  - Basic data wrangling
- Learning how to write scripts can:
  - Allow you to automate, record and document tasks that might be complex, repetitive, error prone etc.
  - Join disparate processes together

How do we communicate using a shell?

- Shell provides a **read** → **evaluate** → **print** (REPL) loop.
- We say what we want to do by typing in **commands**.
- Commands typically run programs installed on the system
  - Though sometimes they're special "builtin" commands provided by the shell itself
  - It's also possible to create your own commands
- Analogy: some similarities with issuing commands in English...

English analogy: Donald Trump's TODO list

- **Bomb** hurricane
- **Drink** covfefe noisily
- **Eat** hamburgers
- **Try to buy** Greenland

- **Verbs** say what you're **doing**
- Nouns say **what/who** is involved
- Adverbs provide additional information

How do we communicate using a shell?

- Shell commands are kind of similar to English
- But:
  - They need to be written precisely
  - They use funny symbols... making things harder to read
  - Commands are often cryptic / obscure / silly
  - We'll see lots of examples today!
- We can record a series of commands together as a **script**

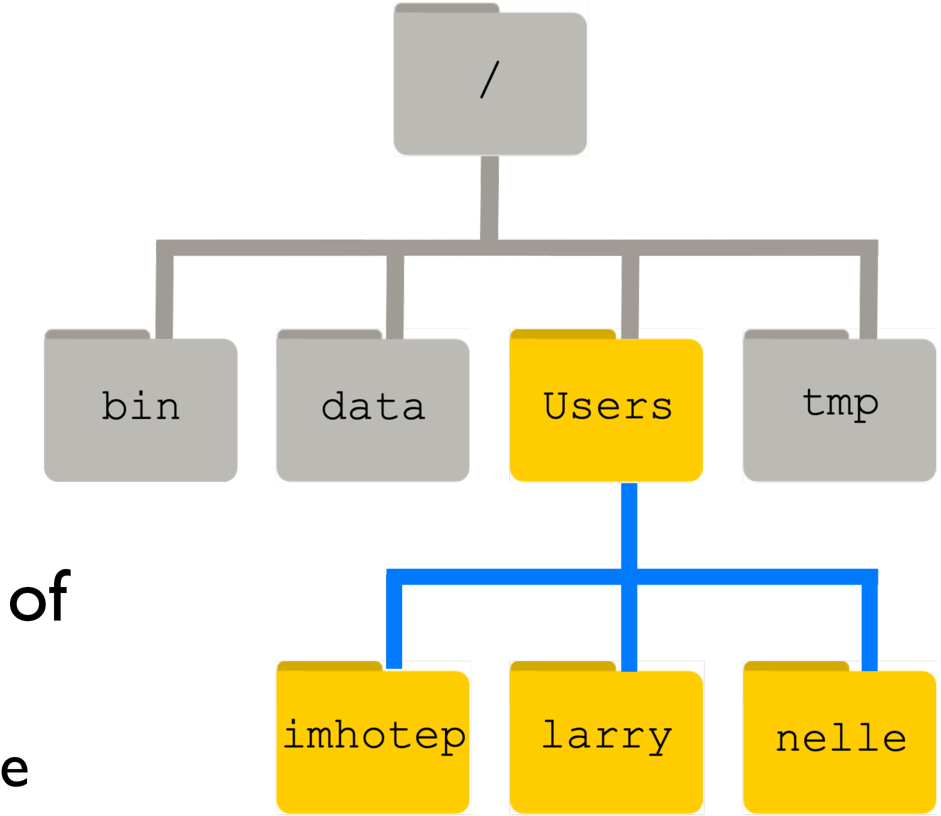
2. Navigating Files & Directories

Objectives

- Learn about Files & Directories
- Understand hierarchical (tree) file systems
- Understand absolute & relative paths
- Learn how to navigate the filesystem
- Learn some handy shortcuts

Key ideas

- **Files** contain information/data
- **Directories** are special files that contain other files and/or directories
  - Often called **Folders**, e.g. in Windows
- This makes a hierarchical (tree) structure called a **filesystem**
- Unix systems have a single **root** at the top of the tree
  - Windows has multiple roots, one for each drive



Key ideas

- Unix has concept of your **Present Working Directory (PWD)**
  - This is the directory you are "in" at any giving time
  - You usually start in your special **home** directory
  - You can move around the filesystem by changing your PWD
- **File paths** tell you where a file lives in the filesystem
  - An **absolute path** shows how to get to a file by starting from the root
  - A **relative path** shows how to get to a file by starting from a chosen directory
- In Unix we make a file path by **joining** the names of each intermediate file or directory with a '/' character

Key Unix commands for navigating

- **pwd** (present working directory) – where am I?
- **cd** (change directory) – navigate to specified directory
- **ls** (list) – see what's in the present or specified directory

Got lost?

- Type **cd** on its own to take you home!

Let's do some practical examples now!

Special navigation shortcuts

Shortcut	What it means
.	current directory
..	parent directory (i.e. up one)
/	root directory (the top of the tree)
~	your **home** (default) directory

Handy keyboard shortcuts

- **Up** and **Down arrow** keys to access typing history
- **Left** and **Right arrow** keys to move within current line
- **Tab** completion to fill in names of commands / files etc.

Getting out of things

Try:

- **Ctrl-C**: interrupts most commands
- **q**: quits some interactive commands (e.g. less)

Found yourself in **vim** and can't get out?!

- Press **Escape** key
- Then type **:q!**
- Then press **Return**

Just memorize these fourteen contextually dependant instructions
Exiting Vim
Eventually
O RLY?
@ThePracticalDev

3. Working with Files & Directories

Objectives

Learn how to...

- Create new directories and files
- Pick good names for new directories & files
- Edit text files
- Delete files & directories
- Rename, move and copy files

Creating a new directory

```
mkdir DIRNAME
```

Creating a new file

- Can use a **text editor** to create a new text file
    - Lesson uses a simple text editor called **nano** for this
    - Other common text editors are **vi(m)** and **emacs**
- Can also create an empty file using the **touch** command

Good naming for files & directories

- Try to stick to combinations of
  - Alphabetic letters (a-z, A-Z)
  - Numbers (0-9)
  - Dot (.), Underscore (_), Hyphen (-)
- Avoid starting names with hyphens
  - That's because options usually start with hyphens... confusion!
- Avoid using spaces in file names
- Avoid exotic letters, symbols and emojis!

Copying, moving or renaming things

- Copy:
  cp SOURCE DESTINATION
- Rename or move:
  mv SOURCE DESTINATION

Deleting files & directories

- **rm FILENAME**
- Beware! Deleting is forever!
- Risky usage:
  - **rm -r** deletes directory **and all of its contents**
  - **rm -rf** forceful version of the above
- Safer usage:
  - **rm -i** asks for confirmation
  - **rm -ir** safe recursive deletion
  - **rmdir** deletes a directory, but only if it's empty

Wildcards

- Wildcards allow you to specify multiple files/dirs whose names contain (match) patterns of your choice.
- Key wildcards
  - ***** - matches any (zero or more) number of characters
  - **?** – matches one character
  - **[…]** – matches any of the characters inside the square brackets

Wildcards & regular expressions

3½: Some handy Unix commands

Outputting

- **echo** – outputs a message

Peeking into files

- **cat** – concatenate, i.e. show file contents
- **more** – show file contents one page at a time
- **less** – better version of more... ho ho ho!
- **head** – show first few lines of file
- **tail** – show bottom few lines of files
- **wc** – word count... also line & character count

Extracting and reformatting data in files

These are all great for manipulating text files:

- **head** & **tail**
- **sort** – sorts file contents
- **uniq** – removes duplicates
    - sort & uniq can be combined to extract unique values or do grouping
- **cut** – picks out columns from tabular data
- **grep** – search file contents (covered in Chapter 7)
- More advanced: **sed** & **awk**
- Even more advanced: write some code (e.g. in Python)

4. Pipes & Filters

Objectives

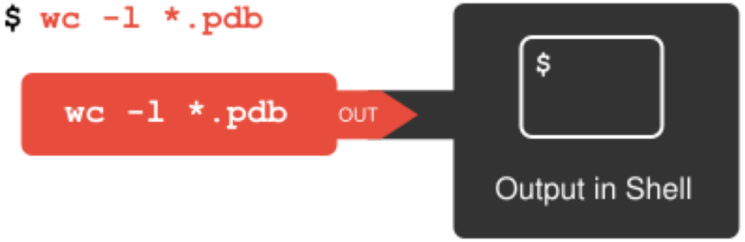
- Learn some handy Unix commands
- Learn how to redirect (save) a command’s output to a file
- Learn how to chain commands together into a pipeline
- Construct some basic pipelines and solve problems using them
- Explore Unix’s Lego brick philosophy

Redirecting output & making pipes

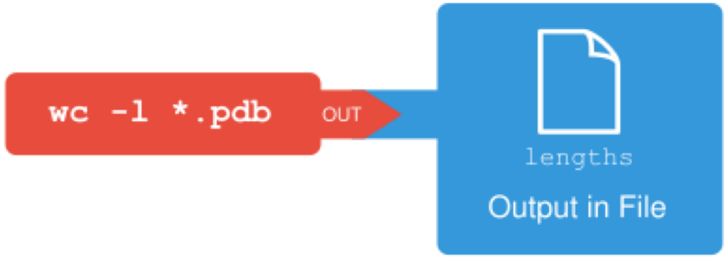
- **command > file**
  Redirects a command’s output to a file
  Overwrites any existing content!
- **command >> file**
  Appends a command’s output to a file
- **first | second**
  Creates a pipeline: the output of the first command is used as the input to the second.

Redirecting output & making pipes

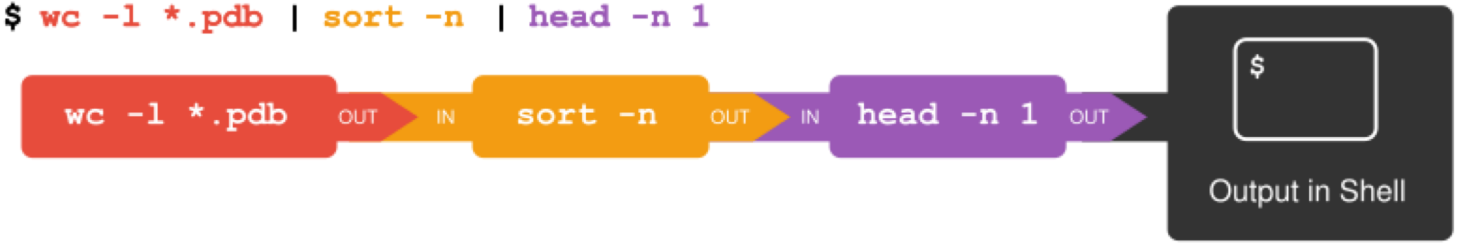
```
$ wc -l *.pdb
```



```
$ wc -l *.pdb > lengths
```



```
$ wc -l *.pdb | sort -n | head -n 1
```



5. Loops

Motivation & objectives

Motivation:

- Sometimes you need to apply the same set of commands to a bunch of files
    - Manually handling each file is tedious and error prone!
- **Loops** provide a nice solution to this
    - Loops come up in other computing contexts too so good reusable skill!

Objectives:

- Learn how to write loops in the Unix shell
- Understand the basics about variables
- Demonstrate how to see what commands you’ve recently executed
- Learn more handy keyboard shortcuts

6. Shell Scripts

Motivation & objectives

Learn how to ”record” or automate processes that you want to do over and over again

- This will save you time in the long run
- Reduces risk of making errors
- You can document what your script is doing... handy when you read it later!
- You can build up a personal library of useful scripts
- Scripts are used to submit jobs to Eddie and other supercomputers

7. Finding Things

Objectives

- Learn how to use **grep** to find content within files
- Learn how to use **find** to search for files
- Learn how to combine grep & find for more complex searching

Grep exercise

- Go back to the **creatures** directory
- Remember how we earlier extracted the CLASSIFICATION line from one of these files?
  - E.g. head -n 2 basilisk.dat | tail -n 1
- Can you use grep to do the same thing?

Searching for chemical elements

1. Go to the top of **data-shell**
2. Write a command to find all *.pdb files
   These all represent various chemical compounds
3. Pick **one** file and look at it using the **less** command
4. Note the ATOM lines - the 3rd column is a chemical element present in the compound
5. Can you write a command to find all *.pdb files for elements containing Chlorine (Cl)?

Try to make your command as reliable as you can!

Possible decent solution

```
grep -wi Cl $(find . -name “*.pdb”) | grep ATOM
```

- Using grep -i as some files say CL but others say Cl
- grepping ATOM ensures we’re only looking at the ATOM lines
- This lists more than just the matching file names though.

Wildcards & regular expressions