

# OS Agenda

Agenda OS420 Operating Systems CSC 420

## README

This file contains the agenda overview (what I had planned), the objectives (what we managed to do) and (much of the) content of each taught session of the course. I want to avoid splitting the content up over many files - so that you have to navigate as little as possible (like a book)!

The companion file to this file, less structured and with the captain's log, is the [notes.org](#) file.

## Welcome to the course - w1s1 (11-Jan)



- Aspiration and ambition (Lyon Data Science program)
- Introduction to the course & the lecturer
- Homework assignments: **GitHub**, DataCamp, Emacs
- What's next?

## Aspirations and ambitions (DS program 2021-2023)

CLASS	CODE	TERM	Topics
Data Science Tools and Methods	DSC 101	Fall 2021	R, Basic EDA, Base R
Introduction to Advanced Data Science	DSC 205	Spring 2022	R, Advanced EDA, Tidyverse
Database Theory and Applications	CSC 330	Spring 2022	SQL, SQLite
Operating Systems	CSC 420	Spring 2022	Bash, awk, sed, regular expressions
Applied Math for Data Science	DSC 482/MTH 360	Fall 2022	Probability, Statistics + R
Data Visualization	DSC 302	Fall 2022	D3, Processing, Javascript, Bokeh
Machine Learning	DSC 305	Spring 2023	Predictive algorithms, neural nets
Digital Humanities	CSC 105	Spring 2023	Data science applications

## Introduction to the course & the lecturer



Figure 2: DESY APE research group, Hamburg/Rome, 1994

- Why me?
- Syllabus for this course ([Schoology](#))

## Homework assignments week 1 (11-Jan/13-Jan-2022)

- GitHub Hello World Exercise ([Info: FAQ](#))
- DataCamp platform registration ([Link: Schoology](#))
- GNU Emacs installation ([Info: FAQ](#))

## GitHub

- What is it?
  - Software development platform
  - Built around Git by Linus Torvalds
  - Bought by Microsoft in 2018
  - AI support (e.g. [GitHub Copilot](#))

Watch: "[What is GitHub?](#)" (GitHub, 2016)



Gif: "So long binder of requirements" Source: GitHub

- Why are we using it?

*Image: Org-mode file in GitHub*

main cc100 / 2\_installation / babel.org Go to file ...

birkenkrahe tweaks Latest commit f47abca 1 hour ago History

All contributor

54 lines (39 sloc) | 1.5 KB Raw Blame

```
<<babel.org>>
```

## Babel test

This file demonstrates working with source code in Emacs for a number of different languages.

1. To run a code chunk as a whole, type `C-c C-c`. The result will appear immediately below the chunk.[fn:1]
2. look at the code in a separate buffer and run them in parts. To open a buffer with the code, type `C-c C-``.
3. To print an org-mode file, type `C-c C-e` and choose a print format from the list.

Running chunks will only work if Emacs can find the respective programs[fn:2], and if a compiler (for C), or an interpreter (for R and SQLite) were installed.

The code block needs to be named as shown. If you want the result and the code shown in the printout, you need to specify `:exports both`.

```
#include <stdio.h>

int main(void) {
    puts("Hello world");
    return 0;
}
```

```
Hello world
```

In the second version, both the header and the function definition are preset so that you can see the inside of the function only.

```
puts("Hello world");
```

```
Hello world
```

## Footnotes

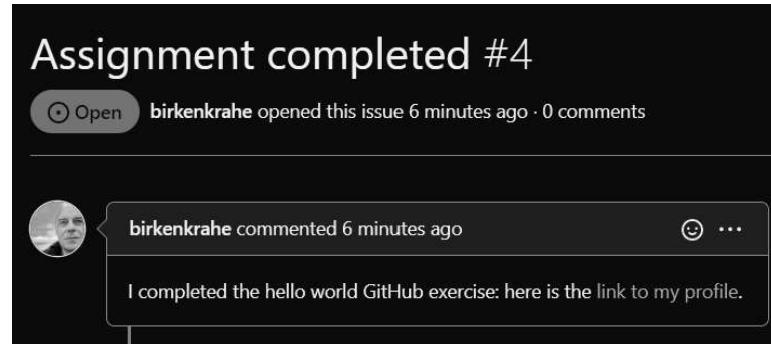
[fn:2]This is why we changed the Windows PATH variable during the installation of the programs R and GNU gcc (here).

[fn:1]Provided the block has been formatted correctly.

- It's free
- To host course materials
- Upload assignments (esp. Emacs Org-files)

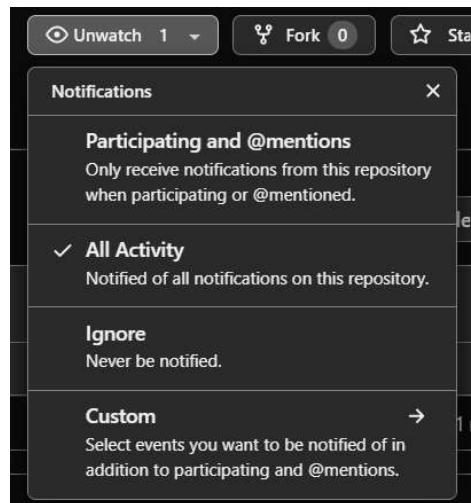
- Discussion
  - Wiki for collaboration
  - Complements Schoology
  - What will you have to do?
    - Sign up with GitHub - use Lyon Email
    - Pick an available username **using your own first and last name**, e.g. MarcusBirkenkrahe, or DonaldTrump
    - Complete GitHub Hello World exercise (see FAQ)
    - Give me your GitHub username so that I can add you as a collaborator to my private os420 repository
    - Create an issue from the os420 repository like in the example below (except from your account instead of mine).
- 

*Image: Issue "Assignment completed"*



If you do have a GitHub account already, do the exercise anyway using your existing account (it takes 10 min)! Make sure you let me know what your user name is so that I can add you to my repo.

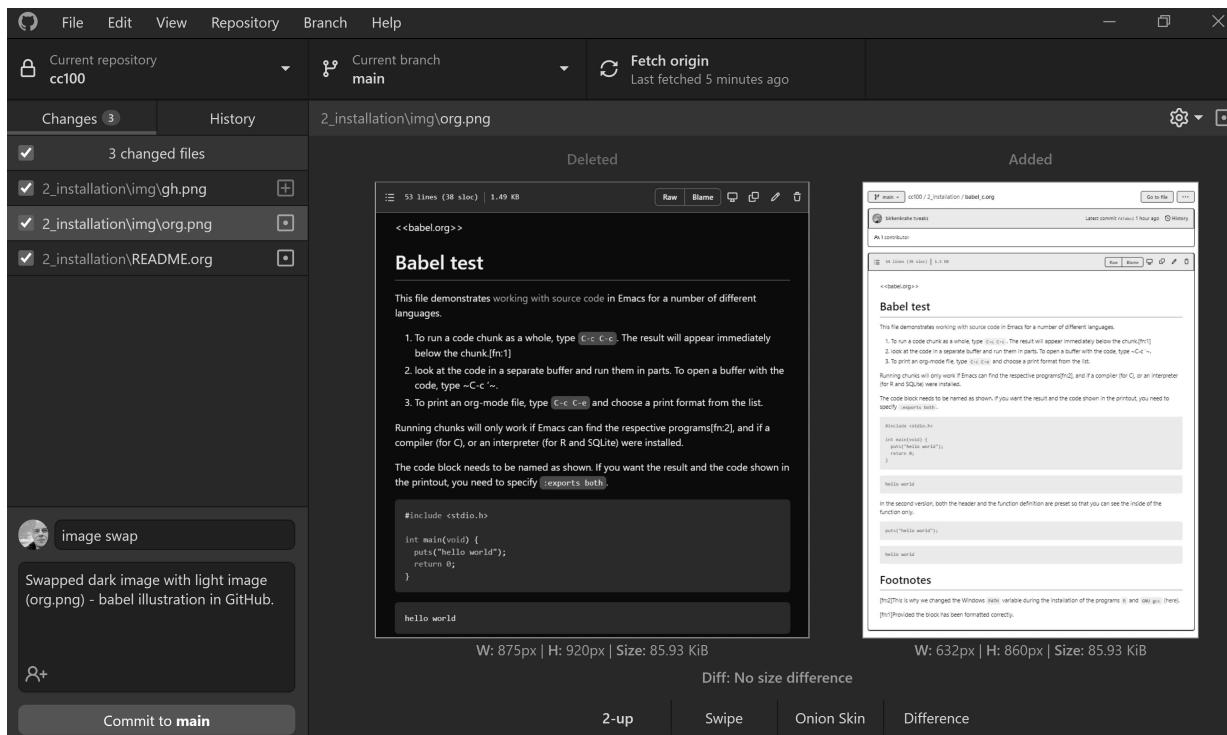
- What else can you do?
  - You can fork the os420 repository
  - You can watch the os420 repository - and set Notifications to Participating and @mentions so that you see my comments (see image below).



*Image: Notifications settings when watching a repository*

- You can submit issues from the repository (e.g. if you notice mistakes or if you want extra information, or to share a link)
- You can participate in discussions (sometimes I will make you)
- You can add to the wiki (e.g. comments and links to interesting resources)
- You can install the mobile app on your smartphone
- You can use it as a platform for projects or coding
- You can download the desktop client to manage repos on your PC (see image below).

*Image: GitHub desktop client commit*



## DataCamp

Title	Assignees	Status	Due By	C	A	CR	Details
Introduction to Shell Manipulating files and directories Chapter	Team	Active	Feb 1, 09:30 CST	0	1	0%	<a href="#">View</a>
Introduction to Shell Manipulating data Chapter	Team	Active	Feb 8, 09:30 CST	0	1	0%	<a href="#">View</a>
Introduction to Shell Combining tools Chapter	Team	Active	Feb 15, 09:30 CST	0	1	0%	<a href="#">View</a>
Introduction to Shell Batch processing Chapter	Team	Active	Feb 22, 09:30 CST	0	1	0%	<a href="#">View</a>
Introduction to Shell Creating new tools Chapter	Team	Active	Mar 1, 09:30 CST	0	1	0%	<a href="#">View</a>

- Why are we using it?
- How are we using it?
- What will you have to do?

## GNU Emacs



- Why are we using it?
- How are we using it?
- What will you have to do?

## What's next?

- See schedule ([GitHub](#))
- Watch online lecture on "Systems" (to be published)
- Everything else => followup notes ([GitHub](#))
- ~~See you (hopefully) Thursday in class! (Lyon 104)~~

## GitHub, GNU Emacs installation - w1s2 (13-Jan)

### Overview

HOW	WHAT
Review	<a href="#">GitHub Hello World exercise (FAQ)</a>
Lecture	What operating systems do
Practice	<a href="#">Install GNU Emacs (FAQ)</a>
Demo	<a href="#">Emacs guided tour</a>
Self	<a href="#">Work through the Emacs onboard tutorial (18p).</a>

### Objectives

- [X] Understand the basics of Git
- [X] Describe the general organization of a computer system
- [X] Install the GNU Emacs editor on your OS
- [ ] Understand how GNU Emacs works
- [ ] Make GNU Emacs work for you

## Interrupts, basic I/O - w2s3 (18-Jan)

### Overview

HOW	WHAT
Review (S)	<a href="#">Quiz: course, OS foundations / GNU Emacs</a>

HOW	WHAT
Resource (S)	<a href="#">Fundamentals of Operating Systems YouTube playlist</a>
Lecture (S)	Interrupts / I/O operations example / bootstrapping
Demo	<a href="#">Emacs guided tour</a>
Self	<a href="#">Work through the Emacs onboard tutorial (18p)</a>

Nachtrag: watch [History of Databases](#) and [The Computer Chronicles](#)

## Objectives

- [X] Review / retention: complete Schoology Quiz 1 (15 min)
- [X] Understand bootstrapping, and interrupts management
- [X] Understand how basic I/O processes work
- [ ] Understand how GNU Emacs works
- [ ] Make GNU Emacs work for you

## What's next?

- GNU Emacs practice exercises (class)
- Computer system architecture
- Getting started with Pi

## OS tasks, virtualization, GNU Emacs - w2s4 (20-Jan)

### Overview

HOW	WHAT
Lecture	Storage structure & OS management tasks
Practice	<a href="#">Emacs guided tour (tour)(tutorial)</a>

## Objectives

- [X] Storage structure, User vs kernel mode, multiprogramming
- [X] Management tasks of the Operating System
- [X] Virtualization and open source vs commercial system
- [X] Understand how GNU Emacs works
- [ ] Make GNU Emacs work for you

## What's next?

- Operating system services & design principles
- Emacs practice & assignment
- Getting started with Pi: bootloading Raspbian Linux
- Complete quiz 2 online **before class**
- Will do 5 min review in class together

## OS foundations, Eshell - w3s5 (25-Jan)

### Overview

HOW	WHAT
Summary	Foundations of Operating Systems (10 tenets)
Preview	DataCamp course "Introduction to Shell"
Practice	Open three shells inside Emacs
Assignment	Create hello world shell program in Emacs

## Objectives

- [X] Summarize foundations of operating systems
- [X] Understand how GNU Emacs shells work
- [X] Understand the first DataCamp assignment ([Intro to Shell](#))
- [ ] Create an bash(1) hello world program in the shell
- [ ] Run shell program inside Emacs

## Summary: foundations of operating systems

### 10 tenets

1. An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run.
2. Interrupts are a key way in which hardware interacts with the operating system. A hardware device triggers an interrupt by sending a signal to alert the CPU, whose interrupt handler manages the interrupt.
3. For a computer to do its job, programs must be loaded in main memory (RAM), which is the only memory area that the CPU can access directly.
4. To best utilize the CPU, the OS can handle several jobs in memory at the same time so that there's always one job to execute. True multitasking, however, is an illusion.
5. To prevent user programs from interfering, the system hardware has two modes: user mode and kernel ("sudo") mode.
6. Privileged instructions that can only be executed in kernel mode include: switching to kernel mode; I/O control; time management; interrupt management.
7. Process management includes creating and deleting processes, and providing process communication and synchronization. Processes are active, programs are passive.
8. Memory management means that the OS keeps track of what parts of memory are being used and by whom, and dynamically freeing and allocating memory.
9. Storage space is managed by the OS through file systems (files, directories) and managing space on mass-storage devices.
10. Virtualization involves abstracting a computer's hardware into several different execution environments.

### Short definition

The Operating System takes physical resources (CPU, memory, disk), and **virtualizes** them. It handles **concurrent** processes, and it stores files **persistently** to make them safe in the long term.

### OS attributes

- **Performance** / overhead reduction
- **Protection** between applications, and between OS and apps
- **Isolation** of processes from one another
- **Reliability** of operations
- **Security** against malicious attacks
- **Mobility** across, and towards smaller, and embedded devices

### OS Timeline

---

Early era	OS are just libraries with (human) batch operators
Mainframe era	Protection through the system handler
Minicomputer era	Interrupt-based memory management
PC era	DOS attacks and infinite MacOS loops
Modern era	Linux and the return to sanity

---

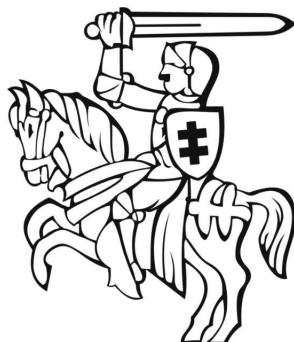


Figure 9: Unix the white knight of Operating Systems

## DataCamp course: Introduction to Shell

This chapter is a brief introduction to the Unix shell. You'll learn why it is still in use after almost 50 years, how it compares to the graphical tools you may be more familiar with, how to move around in the shell, and how to create, modify, and delete files and folders.

### Three shells inside Emacs!

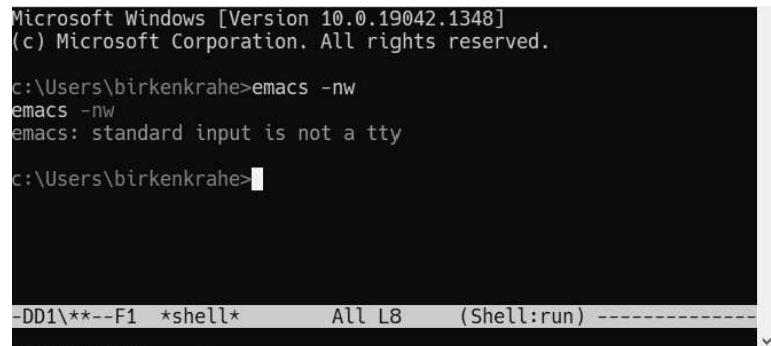
- Works really well only under Linux or MacOS

SHELL COMMAND	CHARACTERISTICS	MODELINES
M-x shell	Windows shell CMD.exe	*shell*
M-x eshell	Lisp simulated shell	*eshell*
M-x term	Terminal emulator	*terminal*

Cp. the variable `shell-file-name`.

- Start Emacs from the terminal: `emacs -nw -q`
- Start all three shells in Emacs
- Start Emacs with `emacs -nw` inside an Emacs `~*shell*`

You should get the error message `emacs: standard input is not a tty`. TTY stands for "TeleTYpewriter". The (Unix) `tty` command is used to check if the output is a terminal or not (see [Wikipedia](#)).



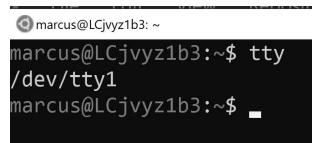
```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

c:\Users\birkenkrahe>emacs -nw
emacs -nw
emacs: standard input is not a tty

c:\Users\birkenkrahe>
```

The screenshot shows a Windows command prompt window. The user has typed `emacs -nw`. The response is `emacs: standard input is not a tty`. The window title bar says "DD1\\*\\*\\*F1 \*shell\*".

On Linux (Ubuntu App in Windows 10):



```
marcus@LCjvyz1b3: ~
marcus@LCjvyz1b3:~$ tty
/dev/tty1
marcus@LCjvyz1b3:~$ -
```

The screenshot shows a Linux terminal window. The user has typed `tty`. The response is `/dev/tty1`. The window title bar says "DD1\\*\\*\\*F1 \*shell\*".

**Assignment: hello world!**

```

c:\Users\birkenkrahe>dir hello.sh
dir hello.sh
  Volume in drive C is OS
  Volume Serial Number is 0654-135C

  Directory of c:\Users\birkenkrahe

01/25/2022  08:27 AM           50 hello.sh
               1 File(s)      50 bytes
               0 Dir(s)  354,250,600,448 bytes free

c:\Users\birkenkrahe>
-DD1\*--F1 *shell*      Bot L119  (Shell:run) -----
#!/c:/Windows/system32/bash.exe
echo hello world

-DD-\---F1 hello.sh      All L3      (Shell-script[bash]) -----
~ $ ls -la hello.sh
-rw-rw-rw-  1 Birkenkrahe    Domain U      50 2022-01-25 08:27 hello.sh
~ $

-DD1\*--F1 *eshell*      Bot L202  (Eshell) -----

```

Figure 12: Windows CMD shell in Emacs and hello world pgm

### Next @Pi: eshell demo cpu.c

- Start several processes using `cpu.exe`
- Show process list with `jobs`
- Kill processes in list with `d`
- Start same processes in Ubuntu
- Kill processes with `kill`

### What's next?

- Review 'hello world' shell exercise
- Fix .emacs issue on desktop computers
- Getting started with Pi: bootloading Raspbian Linux

## Shell scripts, Raspberry Pi setup - w3s6 (27-Jan)

### Overview

HOW	WHAT
Review	<code>hello.sh</code> <a href="#">assignment</a>
Fix	.emacs issue ( <a href="https://tinyurl.com/lyonemacs">https://tinyurl.com/lyonemacs</a> )
	Find the file on the lab desktop
	Put it into the <code>\$HOME</code> directory
	Restart Emacs
Practice	Getting started with Raspberry Pi: installation

### Objectives

- [X] Review `hello.sh` exercise
- [X] Install Raspberry Pi OS (Raspbian Linux) via NOOBS
- [X] Complete basic Pi setup with `sudo raspi-config`

## Review assignment

- Start Emacs without init file

```
> emacs -nw -q
```

- Known `Class` not registered error on Windows 10 ([fix](#)). See notes for a quick solution (installing CygWin).

## Set up Raspberry Pi

- Open the box in front of you
- Take out the Pi and connect it to the KVM switch
- Connect the Ethernet cable to the Pi
- Plug in the power cord and then plug it into the Pi
- Install using NOOBS
- Get the password from me!
- Open a terminal app on the Pi
- In terminal, enter `sudo raspi-config`
- In the configurator, change boot options > Desktop/GUI to `Console autologin`, then <TAB> to <Finish>
- Reboot the Pi (`sudo reboot`)
- Login as user `pi` with password
- Check network connection with `ifconfig`
- Update OS with `sudo apt update`
- Upgrade OS with `sudo apt upgrade -y`
- Install Emacs with `sudo apt install emacs`
- Start emacs, check it and exit again
- Finish session with `sudo shutdown now`
- Unplug the power chord, then the KVM connections
- Return Pi to box
- Cross yourself and close the box.

## What's next

- Review first DataCamp assignment
- Explore Linux on the command line

## Linux shell, UNIX man pages - w4s7 (1-Feb)

### Overview

HOW	WHAT
Review	Quiz 3: file and folders / Installing Raspbian
Lecture	Raspberry Pi - the hardware & the history
FAQ	Should you upgrade your Operating System?
Pi Practice	Understanding the shell and Unix man pages

### Objectives

- [X] Review Introduction to the shell - Files and folders (Quiz 3)
- [X] Getting OS release information
- [X] Understand the shell(s) - Unix man pages
- [ ] Understand the Linux file tree

### Setting up the Linux boxes (almost every session)

This is something you can do as soon as you arrive:

- Take out the [Raspberry] Pi
- Connect the HDMI from the KVM switch to the Pi
- Connect the USB from the KVM switch to the Pi (any USB port)
- Plug in the power charger under the desk
- Connect the mini-USB of the power charger to the Pi
- Press the button on the KVM switch (color changes to green)
- The system should boot straight to console with autologin as user `pi`

The Ethernet cable is not usually needed except for installations and for updates and upgrades.

For the Pi400, the setup is slightly different because the Pi4 has a mini-HDMI connector.

## Understanding the shell

"Typing commands instead of clicking and dragging may seem clumsy at first, but as you will see, once you start spelling out what you want the computer to do, you can **combine** old commands to create new ones and **automate** repetitive operations with just a few keystrokes." (Shotts, 2019)

- The shell is just another program to pass keyboard commands to the OS
- On the desktop GUI, the shell is emulated (an X Windows program connects to it and interprets key strokes, just like eshell on Emacs)
- The original shell is the Bourne Shell (/bin/shell).
- Linux uses an enhanced Bourne Shell, the Bourne-Again SHell bash.
- [X] Look up the man page for bash with the command `man bash`<sup>1</sup>.
- [X] To get out, press `q`
- There are many other shells, including: ksh (Korn shell), csh (C shell), zsh (MacOS shell).
- Different shells differ in editing styles (command line editing), scripting abilities (e.g. closer to C), and process management functions (what you can do with CPU processes).
- Emacs commands are Unix commands: `c-a`, `c-e` etc. all work
- Some other shell commands to try now:
  - [X] history keeps the last 1000 commands or so. Commands are numbered. How can you repeat a command?
  - [X] date and cal time functions
  - [X] df and df -H (for humans) show the free disk space
  - [X] free and free -h (for humans) show the free memory
  - [X]

`echo` repeats back what you type - except when you use variables as arguments:

```
$ echo $PS1 # prompt path name
$ echo $PS2 # prompt for inner shell (like SQLite or R)
$ marcus = "me" # defining a variable
$ echo marcus # this just returns the string "marcus"
$ echo $marcus # returns "me"
```

- A terminal shell session is ended with the `exit` command.
- [X] End your terminal session!

## Unix manual pages

- Man pages are split into eight numbered sections:

Section	Description
1	General commands
2	System calls
3	Library functions, covering in particular the C standard library
4	Special files (usually devices, those found in /dev) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellaneous
8	System administration commands and daemons

Example: look for the man page of `sshd`, the OpenSSH daemon. What does it do? ([Source](#))

## Unplugging the Pi (almost every session)

- Shut down the Pi with `sudo shutdown now` on the console
- Unplug the power mini-USB
- Unplug the other cables
- Press the button on the KVM switch to return to Windows
- Sign out of Windows if necessary

## Shell commands, Linux file tree - w5s8 (8-Feb)

## Overview

HOW	WHAT
Test info	Test on Thursday, Feb 10 at 10.00-10.45 AM
Setup	USB/HDMI > power > switch > startx
Review	Shell warm-up exercise (15 min)
<b>Poll (vote)</b>	<u>Should you update your operating system?</u>
Practice	<pre>sudo apt update -y [update OS] sudo apt upgrade -y [upgrade OS] sudo apt autoremove -y [remove old OS]</pre>
Lecture	The Linux File System ("Everything is a file")
Practice	<p>Navigating the file system (<a href="#">DataCamp</a>)</p> <p>Manipulate files and folders (<a href="#">DataCamp</a>)</p> <p>Start an <a href="#">interactive notebook bash.org</a></p>
Shutdown	<code>sudo shutdown now &gt; USB/HDMI &gt; Power &gt; switch</code>

## Objectives

1. [X] Poll/Discussion: should you upgrade your OS?
2. [X] Understand Thursday test rules and content
3. [X] Review shell command structure
4. [X] Understand the Linux file system structure and content
5. [X] Understand navigation and the Linux file tree
6. [ ] Start an interactive Org-mode notebook
7. [ ] Understand how to manipulate data using the shell

## Test info

- Online in Schoology
- Quiz 1-3 are not visible during the test
- The 10 hardest questions of quiz 1-3 (< 50%)
- 10 brand new questions
- Maximum time = 45 min

## Review: shell command structure

- The structure of all shell commands: [cmd] -[options] [arguments]
- There must be > 1 space between all elements
- There must not be a space between the option and the dash
- Many commands have an -v option that provides you with information at run-time
- There are short options (e.g. -1) and long options (e.g. --reverse)

```
pi@raspberrypi:~/GitHub/os420$ ls -lt *.org
-rw-r--r-- 1 pi pi 31073 Feb  6 10:45 agenda.org
-rw-r--r-- 1 pi pi 20720 Feb  6 09:38 notes.org
-rw-r--r-- 1 pi pi 3064 Feb  5 08:56 FAQ.org
-rw-r--r-- 1 pi pi 11641 Feb  4 08:07 schedule.org
-rw-r--r-- 1 pi pi 21862 Feb  4 08:07 syllabus.org
-rw-r--r-- 1 pi pi 1157 Feb  4 08:07 bookmarks.org
-rw-r--r-- 1 pi pi 1828 Feb  4 08:07 diary.org
-rw-r--r-- 1 pi pi 1389 Feb  4 08:07 README.org

pi@raspberrypi:~/GitHub/os420$ ls -lt --reverse *.org
-rw-r--r-- 1 pi pi 1389 Feb  4 08:07 README.org
-rw-r--r-- 1 pi pi 1828 Feb  4 08:07 diary.org
-rw-r--r-- 1 pi pi 1157 Feb  4 08:07 bookmarks.org
-rw-r--r-- 1 pi pi 21862 Feb  4 08:07 syllabus.org
-rw-r--r-- 1 pi pi 11641 Feb  4 08:07 schedule.org
-rw-r--r-- 1 pi pi 3064 Feb  5 08:56 FAQ.org
-rw-r--r-- 1 pi pi 20720 Feb  6 09:38 notes.org
-rw-r--r-- 1 pi pi 31073 Feb  6 10:45 agenda.org
pi@raspberrypi:~/GitHub/os420$ █
```

Figure 13: long time listing of Org-files (and reversion)

- Another useful long option for the `emacs` command that starts GNU Emacs is `--chdir=[dirname]` where `[dirname]` is the name of the folder where you want Emacs to "wake up"

## Shell warm-up exercise

Complete the exercise on the command line (15 min):

---

Go to your `$home` directory, `/home/pi`

Use `wget` to get `sample.txt` from `os420` in GitHub<sup>2</sup>

Display first line of `sample.txt`

Make a sub directory titled `practice` in `$home`

Change directory to `/home/pi/practice`

Verify that you are where you want to be

Move `sample.txt` to your current directory as `sample_1.txt`

Make a copy of `sample_1.txt` and name it `sample_2.txt`

Ascertain that the files are indeed identical by using `diff`

View both files together using `less`

Inside `less`, move between the two files

Leave `less`

---

- Solution in the [interactive notebook](#)
- Will make a solution screencast, too

## THE LINUX FILE SYSTEM ("EVERYTHING IS A FILE")

- Open a terminal and look at the file system

```
ls -lF /
```

DIRECTORY	CONTENT
/	Root directory where everything begins
/bin	Executable binaries for the OS to boot and run
/boot	Linux kernel, initial RAM disk image to boot
/dev	List for kernel with all known devices
/etc	System configuration files (e.g. /etc/passwd)
/home	Directory for user directories (e.g. /home/pi)
/lib	Shared library files (like Windows DLLs)
/lost+found	Panic room for each formatted disk partition
/media	Mount points for removable media (e.g. USB stick)
/mnt	Mount points for manually mounted removable media
/opt	Optional commercial software
/proc	Virtual FS for the kernel (e.g. /proc/cpuinfo)
/root	\$HOME directory of the root super-user
/sbin	System binaries for system tasks (sudo shutdown)
/tmp	Holding bay for temp files, emptied at reboot
/usr	Programs and support files for regular users
/usr/bin	Executable programs of the distro <sup>3</sup> (e.g. cat)
/usr/lib	Shared libraries for /usr/bin programs
/usr/local	Programs not included in your distro
/usr/sbin	More system administration programs
/usr/share	Shared data for /usr/bin programs (e.g. sound files)
/usr/share/doc	Man pages and other package documentation
/var	Databases, spool files, user mail (volatile files)
/var/log	Records of system activity (e.g. /var/log/syslog)

## What's next

- Test 1, Thursday 10 February 2022, 10:00-10:45
- Raspberry Pi history & hardware
- Manipulating files and data
- New DataCamp assignment due Feb 15 ("Combining tools")

## Raspberry Pi history and hardware - w5s9 (10-Feb)

### Overview

HOW	WHAT
Lecture/demo	Raspberry Pi - the hardware & the history 1
Test 1	10:00-10:45 AM

### Objectives

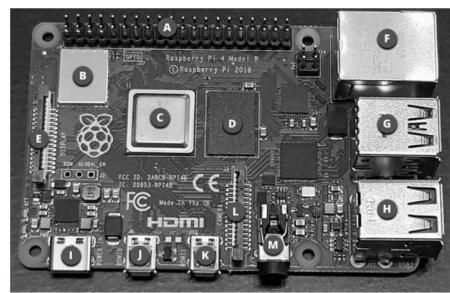
- [X] Know Raspberry Pi: history & hardware (part 1)
- [X] Complete test 1 (online)

### Raspberry Pi - hardware & history: hardware

- Raspberry Pi history

- Which other devices have an Arm processor?

# COMPONENTS



- A: GPIO Pins
- B: Wireless Card
- C: Processor (CPU)
- D: Memory (RAM)
- E: Display connector (DSI)
- F: Ethernet port
- G: 2x USB3 ports
- H: 2x USB2 ports
- I: USB-C power input
- J: Micro HDMI display output 1
- K: Micro HDMI display output 2
- L: Camera connector
- M: Jack 3.5 mm output

Figure 14: Raspberry Pi 4 Board (Source: raspberryptips.com)

	Models A			Models B					Raspberry Pi 400	Raspberry Pi Zero v1.2	Raspberry Pi Zero v1.3	Raspberry Pi Zero WH
Release Date	Raspberry Pi 1 A	Raspberry Pi 1 A+	Raspberry Pi 3 A+	Raspberry Pi Model B	Raspberry Pi B+	Raspberry Pi 2 B	Raspberry Pi 3 B	Raspberry Pi 3 B+	Raspberry Pi 4 B			
Architecture	Feb 2012 ARM v6	Nov 2014 ARM v6	Nov 2018 ARM v6	Mar 2012 ARM v6	Jul 2014 ARM v6	Feb 2015 ARM v6	Feb 2016 ARM v6	Mar 2018 ARM v6	Jun 2019 ARM v6	Nov 2020 ARM v6	May 2016 ARM v6	Feb 2017 ARM v6
CPU	BCM2835	BCM2835	BCM2835	BCM2835	BCM2835	BCM2836	BCM2836	BCM2837	BCM2837	BCM2837	BCM2835	BCM2835
Frequency	700 MHz	700 MHz	1.4 GHz	700 MHz	700 MHz	900 MHz	900 MHz	4.4 GHz	4.4 GHz	4.4 GHz	1 GHz	1 GHz
64 bits	-	-	-	-	-	-	-	x	-	-	-	-
RAM	256 MB	512 MB	512 MB	512 MB	512 MB	1 GB	1 GB	1 GB	1 GB	4 GB	512 MB	512 MB
USB	1	1	1	2	4	4	4	4	4x 1.8 GHz	3 [x] USB 2.0, 2x USB3.0	1 [Micro-USB]	1 [Micro-USB]
Display	HDMI	HDMI	HDMI	HDMI	HDMI	HDMI	HDMI	HDMI	HDMI	VGA HDMI	Mini-HDMI	Mini-HDMI
Network	-	-	Wireless	Ethernet	Ethernet	Ethernet	Ethernet, Wi-Fi, Bluetooth	Ethernet, Wi-Fi, Bluetooth	Ethernet, Wi-Fi, Bluetooth	Ethernet, Wi-Fi, Bluetooth	-	-
Dimensions	85 x 56 x 10	85 x 56 x 10	85 x 56 x 10	85 x 56 x 17	85 x 56 x 17	85 x 56 x 17	85 x 56 x 17	85 x 56 x 17	85 x 56 x 17	296 x 122 x 23	65 x 30 x 5	65 x 30 x 5
My advice	A good balance between price and performance. The square form factor is more compact than the B models (square form).			High-end products with the most features are generally models B. Take the latest here for better performances and possibilities.					Perfect to get started with a minimal budget, even if you probably need many accessories.			
Recommended model	<b>Raspberry Pi 3 A+</b>			<b>Raspberry Pi 4 B</b> (4GB is enough in general)					<b>Raspberry Pi Zero W</b> (or WH if you need GPIO pins)			
Recommended kit	<a href="https://raspberrypi.com/kits/a">https://raspberrypi.com/kits/a</a>			<a href="https://raspberrypi.com/kits/b">https://raspberrypi.com/kits/b</a>					<a href="https://raspberrypi.com/kits/zero">https://raspberrypi.com/kits/zero</a>			

Figure 15: Raspberry Pi 4 Board (Source: raspberryptips.com)

- [Identify your model of Raspberry Pi](#) (cat /proc/cpuinfo)
- [Introduction to GPIO pins](#) (gh) - my fifth Pi project

## Review of test 1 - w6s10 (15-Feb)

### News

- **Eliminated** some DataCamp assignments (bash scripting) because otherwise we'll get too far out of step with the course content
- **Join us** for a talk by Matthew Stewart, Stone Ward (Fri 18 Feb 3-3.50 PM) via Google Meet

### Objectives

- [x] Understand test results
- [x] Know what to do different next time
- [x] Discuss all questions and answers
- [x] Share your views & suggestions

### Test results - stats and plots

- The results are nothing to write home about - though > 50% means that the class passed (on average).

Statistics			
# of Grades	22	Average	13.3 (66.49%)
Max Points	20	Standard Deviation	2.29 (11.46%)
Highest Grade	17.08 (85.42%)	Median	13.01 (65.04%)
Lowest Grade	8.67 (43.33%)	Mode	12.43 (62.17%)

Figure 16: Test 1 results (Schoology)

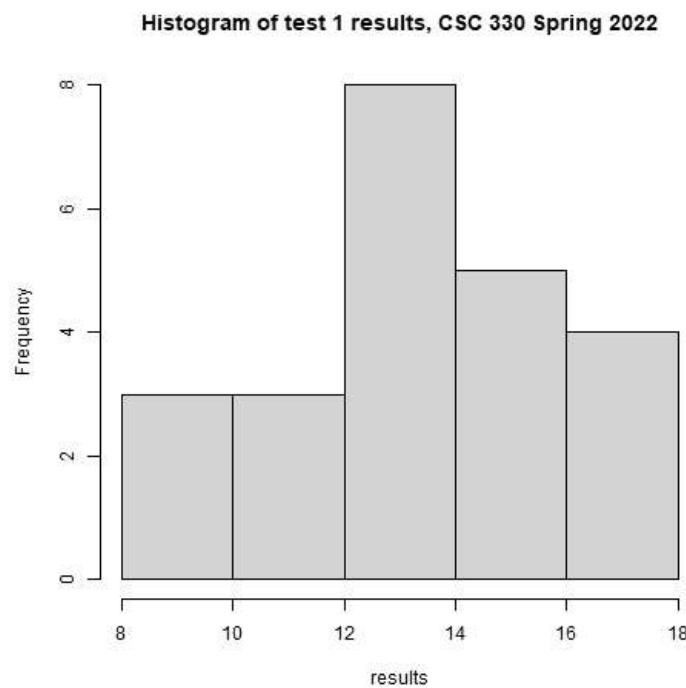
- I am an obsessive fact-checker. When checking the stats with R, I find slightly different results:

```
results <- c(11.88,13.92,12.43,13.6,9.93,8.67,14.87,
           14.87,13.02,10.58,17.08,9.72,12.77,15.42,16.5,
           17.02,13,16.13,14.03,12.93,12.43,14.73,11.9)
length(results)
sd(results)
summary(results)
```

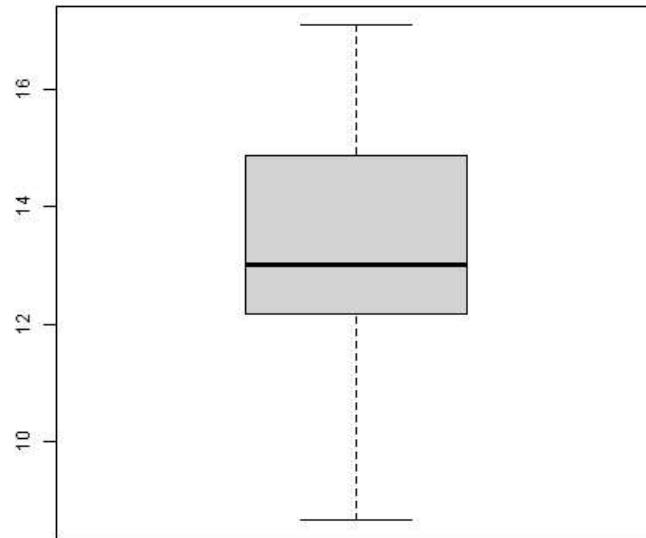
```
[1] 23
[1] 2.314336
   Min. 1st Qu. Median     Mean 3rd Qu.    Max.
  8.67    12.16   13.02   13.37   14.87   17.08
```

- Let's make some plots: histogram, boxplot and density plot. I'd like the histogram and the density plot (a smoothed histogram) to peak more to the right, and for the boxplot to be smaller and higher up.

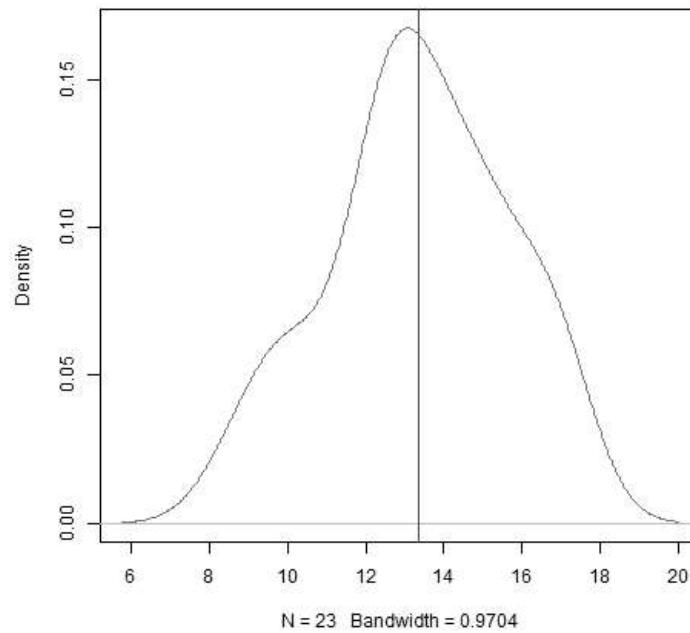
```
hist(results, main="Histogram of test 1 results, CSC 330 Spring 2022")
```



```
boxplot(results, main="Test 1 results, CSC 330 Spring 2022")
```

**Test 1 results, CSC 330 Spring 2022**

```
ave <- mean(results)
d <- density(results)
plot(d, col="steelblue", main="Test 1 Results CSC 330 Spring 2022")
abline(v=ave, col="red")
```

**Test 1 Results CSC 330 Spring 2022**

## Analysis - feedback and action points

- Test 1 can now be played an unlimited number of times. I will add feedback to all new questions today.
- What surprised me most was that many of you did not use the available time. However, I have not (yet) been able to correlate test time and test success.
- See also: "I can teach it to you but I cannot learn it for you"
- Questions:
  - How did you study for this test?
  - If you didn't perform well, what will you change?
  - What can I do to help you help yourself?
- Changes to be applied in future quizzes/tests:
  - Fewer multiple choices (max. 4)
  - Announce if a question has > 1 answer (and/or how many)
  - Try and schedule tests for different classes on different days

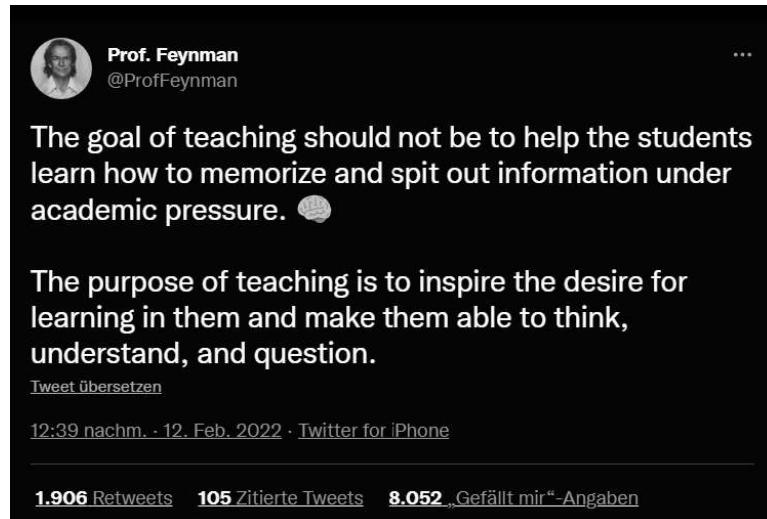


Figure 20: Feynman (via Twitter)

## Test questions and answers

- We go through all questions together
- Everybody can contribute an answer
- Write down questions and ask them now!

## Feedback for some individual questions

- "What is Booting?" - The kernel program must be in memory before the BIOS (Basic IO System), which is part of the kernel, can run. Then, when I/O is established, information can flow between memory (RAM), non-volatile memory (disk/card) and CPU. The system daemon oversees these processes.
- "Sort the OS timeline" - this is a question answered in the agenda file. 1) The whole thing started with OS as libraries without supporting programs - the support (e.g. for switching jobs) was done by humans. 2) Mainframe OS are for large computers - decades before the PC - and the introduction of a system handler program replaced the human operators. 3) Minicomputers are dedicated to specific jobs - e.g. visualization, and graph generation, or number crunching. These computers (which look like large PCs - workstations) already have the modern interrupt-based OS architecture. 4) PCs came up in the 1980s - Microsoft and Apple introduced OS (DOS/Windows and AppleOS/MacOS) that sacrificed security and reliability (compared to UNIX) to convenience and usability. 5) Since the mid 1990s, we are slowly returning to the pre-PC world - the reintroduction of UNIX via Linux, but also in Windows and MacOS itself. Of course, if you can, go Linux all the way, why compromise?
- "Command line prompt" analyzed:
  1. pi is the standard user on the Raspberry Pi, belongs to the gpio group and can assume root rights with sudo.
  2. @ is the at-sign separating the user from the hostname.
  3. raspberrypi is the hostname (you can get it with the command `hostname` or `hostname -I` for the internet address), or the name of the host = your computer.
  4. / is the current directory. In this case `~$HOME`.
  5. \$ is the prompt sign for regular users, after which you enter commands on the command line. For the root user, this prompt changes to #.
- 6. Operating system definition:

The Operating System takes PHYSICAL resources (CPU, memory, disk), and VIRTUALIZES them. It handles CONCURRENT processes, and it stores files PERSISTENTLY to make them SAFE in the long term.

Focuses on the three core characteristics of a modern OS: VIRTUALIZATION (i.e. the user does not see what's happening, everything seems to be in one place), CONCURRENCY (the user thinks processes run simultaneously, while actually the CPU runs one job at a time), and PERSISTENT storage (referring to mass-storage management).

## This week

- Test review
- Raspberry Pi Hardware - GPIO pins

## GPIO pins - w6s11 (17-Feb)

### Overview

HOW	WHAT
Setup	USB/HDMI > Power > Switch > <code>startx</code>
Lecture/demo	Raspberry Pi - the hardware & the history 2
Review/Practice	Manipulating data / Combining shell tools (DataCamp)
Shutdown	<code>sudo shutdown now</code> > USB/HDMI > Power > switch

### Objectives

- [X] Know Raspberry Pi: hardware and history part 2
- [X] Understand GPIOs and how to see and control them

### What's next

- Catching up with DataCamp: back to the Linux command line tools

## Wildcards and Links - w7s12 (22-Feb)

### Objectives

- [X] Review shell commands in DataCamp so far
- [X] Understand wildcards
- [X] Understand file links
- [X] Practice file and data manipulation

### Warmup exercise: what's wrong with these commands?

Situation:

- `agenda.org` is this file
- the directory `testdir` exists in this directory (`./`)
- `test3` contains the file listing of this directory

Complication: none of these commands will run! Why?

```
head n 3 agenda.org # print top three lines to stdout
rm ./testdir/ # remove test
grep -count "org" ./test3 # search test3 for "org"
cat ./testdir/ # view testdir
ls-1a # list long info of all files
```

### Manipulating files and directories

- Files are organized in a hierarchical directory structure<sup>4</sup>
- Imagine you stand somewhere inside an upside down tree:
  - the `pwd` command tells you where you are
  - above you (towards the root) are *parent directories*
  - below you (away from the root) are *child directories*
  - At the top is the *root directory*



Figure 21: upside down tree

- [X]

To see the tree visualized, install the package as super user:

```
$ sudo apt install tree  
$ tree
```

```

pi@raspberrypi: ~
File Edit Tabs Help
|
|   ssh.png
|   wicd.png
|   README.org
|
|   qr
|       qrcode_github.com.png
|       qrcode_linuxcommand.org.png
|       qrcode_linuxfromscratch.org.png
|       qrcode_os_linux_pi.png
|       README.org
|       schedule.org
|       syllabus.org
|
|   Music
|   Pictures
|       gpiodir.png
|       Screenshot from 2022-02-05 14-05-42.png
|       Screenshot from 2022-02-05 14-06-18.png
|       plot.png
|       plot.R
|       printf.c
|       printf.c~
|       Public
|       puts
|       puts.c
|       puts.c~
|       Rplots.pdf
|       sql.db
|       Templates
|       test.db
|       test.org
|       test.org~
|       Videos
|       wiringpi-latest.deb
|
82 directories, 785 files
pi@raspberrypi:~ $ █

```

Figure 22: Linux tree command (screenshot)

- To list files, use `ls`. It has many useful options. I usually use `ls -la` to get a long, complete listing. This is also the standard `Dired` setting in Emacs.
- [X] Compare `ls`, `ls -l`, `ls -la`, `ls -lf`, and `ls -laF`
- [X] Which command would show the top directories of the OS?<sup>5</sup>
- To create an empty file, use `touch`.
- [X] Create a file from the command line with `touch`.
- Filenames are character-sensitive.
- Don't embed spaces in file names, replace them by `_`
- Linux has no concept of file extensions - though many applications do (like Emacs, to identify a major mode like Org-mode)
- Hidden files (often for configuration like `.emacs`) remain hidden unless you invoke the `-a` option of `ls`

## Shell commands so far (DataCamp course)

COMMAND	FUNCTION	OPTIONS
<code>cp</code>	copy file	<code>-iruv</code>
<code>mv</code>	move file	<code>-iuv</code>
<code>mkdir</code>	make directory	
<code>rm</code>	remove files or directories	<code>-irfv</code>

COMMAND	FUNCTION	OPTIONS
rmdir	remove directory	
ln	link	-s
head	show N first lines	-n N
tail	show M last lines	-n M
less	view file pagewise	
more	view file pagewise	
cat	view file	
cut	print COLS separated by SEP	-d SEP -f COLS
paste	merge lines of files	-d SEP
grep	search pattern	-chilnv
wc	word count	-cwl
sort	sort row data	-nrdf
uniq	remove adjacent duplicates	-c
history	last commands entered (!)	
man	Unix man page (sect. 1-8)	

## Wildcards ("globbing")

- `bash(1)` does not recognize regular expressions - there are utilities like `sed` and `awk` that interpret them
- The commands below can be tested with the `echo` command or with `ls`
- Wildcards use character sets and classes, like `:digit:` for numerals, or `:upper~:/lower:~` for uppercase/lowercase letters

WILDCARD	MEANING	Example
*	Matches any characters	*.org
?	Matches any single character	?????.org
[char]	Match any character in [char]	[ba]*.*
		ls FILE[0-9][0-9][0-9]
[!char]	Match any character not in [char]	[!ba]*.*
[[:class:]]	Match any character in class	[[:upper:]]*.org
		[[:!digit:]]*

- Wildcards can be used with any commands that accept file name arguments
- Powerful in connection with pattern search cmd's like `grep`
- Many wildcards have found their way into graphical user interfaces, too<sup>6</sup>
- The SQL injection hack is based on wildcards - the openness of Linux can be detrimental if system administrators are too lax

## Links

- The `ln` command creates hard or symbolic (soft) links (symlinks)

```
ln file link # create hard link from file
ln -s item link # create soft link from item
```

### Hard links and inode

- A hard link creates an additional directory entry for a file
- A hard link cannot leave its (physical) file system
- A hard link is indistinguishable from the file itself
- A file is made up of a **name** part and of a **content** part: when creating hard links, the system assigns a chain of disk blocks to an **inode**, a unique number in the file system that you can see with the listing command `ls -i`

```
ls -i *.org
```

```
388132 agenda.org
389984 bash.org
388137 bookmarks.org
388142 diary.org
388130 FAQ.org
388201 notes.org
388131 README.org
388219 schedule.org
388220 syllabus.org
```

## Soft (sym) links

- A symlink contains a text pointer to the reference file or directory
- Operates like a Windows OS shortcut
- If you write to the symbolic link of a file, the file is written to
- When you delete the link, the file is untouched
- If the file is deleted first, the link is said to be broken

## Practice

You can get the Org-mode file for practice [manipulate.org](#) from GitHub.

## Next topics

- Redirection
- Permissions
- Processes

## Standard streams, Notebook tutorial - w8s13 (1-Mar)

### Objectives

- [X] Review: quiz 4-5 (comment/complain/challenge)
- [X] Review: warming up exercise "N'est pas une pipe"
- [X] Lecture: standard streams
- [X] Practice: getting started with interactive notebooks
- [ ] Practice file and data manipulation 2
- [ ] Getting (more) help

### Always first

- Update your system

```
sudo apt update -y
sudo apt upgrade -y
```

### Warming up - "N'est pas une pipe"



Figure 23: Magritte's "The Treachery of Images" (1928)

After completing the last DataCamp assignment, I got my wires crossed somehow. Can you define these terms and give examples?

TERM	MEANING
grep -v [expr]	inverse pattern search
:digits:	numerical wildcard class
pipeline operator	connects stdout to stdin
redirection operator >	directs stdout to file
???	wildcard for any 3 characters
wc -l	count number of lines

## Standard streams

- Every Unix tool does only one thing really well
- E.g. grep filters, sort sorts, wc counts lines
- Power = OS manages tool communication streams
- There are three such streams: stdin, stdout, stderr

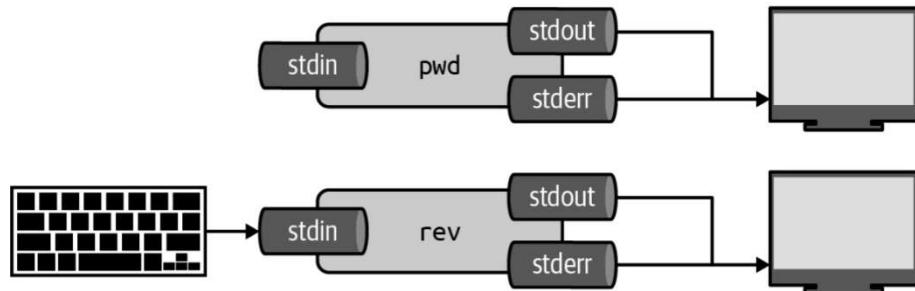


Figure 24: stdin, stdout, stderr streams (Source: Janssens)

- stdout and stderr are by default redirected to terminal
- E.g. rev will wait until it gets stdin (end with C-c C-c)

```
Birkenkrahe@LCjvyz1b3 ~
$ rev
newline
enilwen
I don't know what this command does
seod dnammoc siht tahw wonk t'nod I
```

Figure 25: rev reverses stdin to stdout characters

- One command's stdout can become another command's stdin

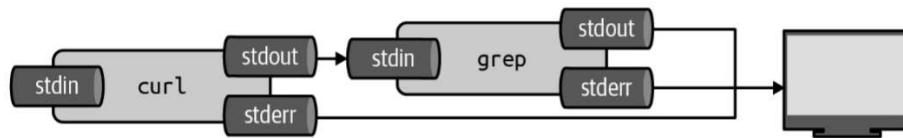


Figure 26: pipe with curl and grep (Source: Janssens)

- E.g. download the book "Alice's Adventures in Wonderland" by Lewis Carroll and pipe it to grep for a pattern search:

```
curl -s "https://www.gutenberg.org/files/11/11-0.txt" | grep " CHAPTER"
```

```
Birkenkrahe@LCjvyz1b3 ~
$ curl -s "https://www.gutenberg.org/files/11/11-0.txt" | grep " CHAPTER"
CHAPTER I. Down the Rabbit-Hole
CHAPTER II. The Pool of Tears
CHAPTER III. A Caucus-Race and a Long Tale
CHAPTER IV. The Rabbit Sends in a Little Bill
CHAPTER V. Advice from a Caterpillar
CHAPTER VI. Pig and Pepper
CHAPTER VII. A Mad Tea-Party
CHAPTER VIII. The Queen's Croquet-Ground
CHAPTER IX. The Mock Turtle's Story
CHAPTER X. The Lobster Quadrille
CHAPTER XI. Who Stole the Tarts?
CHAPTER XII. Alice's Evidence
```

Figure 27: piping example with curl and grep

- And if we want to know how many chapters there are, apply `wc -l`
- Output redirection works with the `>` operator<sup>7</sup>: you can e.g. pipe the chapter list into a file.

```
chap > chapterlist
```

- This presumes that you defined an alias `alias chap='[cmd]'`. Try it!
- You can also append output to a file with `>>`
- If you want to pass a file through a command without running an additional process, use `<`. These two commands do the same thing:

```
cat sample.txt | wc -w
< sample.txt wc -w
```

- We'll deal with more complicated examples in another session!
- This section is based on Janssen's "Data Science at the Command Line (2nd ed)" (2021), which you can [read for free online](#).

## Getting started with interactive notebooks

- Get the files from the GDrive repository
- Work through the Notebook Tutorial ([start\\_nb.org](http://start_nb.org))

## Next

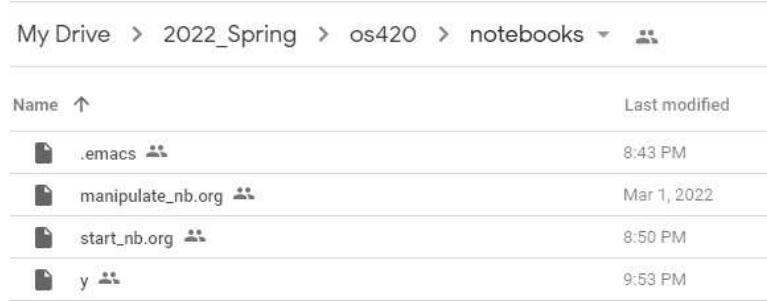
- Redirection
- Permissions
- Processes

## Mid-term speech, REPLIT, redirection stdout- w8s14 (3-Mar)

### Objectives

- [X] Notebooks stuff moved to GDrive
- [X] Mid-term grades improvement
- [X] REPLIT in the cloud - scripting and shell
- [X] Understand redirection better 1

### Notebooks stuff moved to GDrive



The screenshot shows a file list in a web-based file manager. The path is 'My Drive > 2022\_Spring > os420 > notebooks'. The list contains four files:

Name	Last modified
.emacs	8:43 PM
manipulate_nb.org	Mar 1, 2022
start_nb.org	8:50 PM
y	9:53 PM

Figure 28: Replit.com startup screen

### Mid-term grades improvement

- You can ask me personally and specifically, what to do to get your grades up
- There is no reason not to have at good grade in my class:
  1. You can usually submit in-class assignments late
  2. The deadlines of the DataCamp assignments are well known
  3. The quizzes contain ample instructions and can be repeated
  4. Class attendance + Whiteboard screenshots + GitHub info
  5. You can always talk to me for personal support
- Hence, to improve your grade, do:
  - Submit in-class assignments if you could not attend class
  - Complete DataCamp assignments on time
  - Play the quizzes until you have 100% and read the feedback
  - Attend class + look at screenshots + files afterwards
  - Practice your skills whenever you can
  - When you are attending in person, really attend
  - Ask me in or outside of class if anything is unclear
- These skills are related to successful studying, which in turn is related to success in life through traditional values: **discipline, duty, and diligence**. This doesn't have anything to do with computer science.
- What I'm going for in my classes is what I think computer scientists need more than anything else:
  1. Critical thinking and analysis skills
  2. Troubleshooting skills
  3. Research skills

This is nicely mirrored in [this comment](#) to the question "Why are computer science degrees so math intensive when the field doesn't seem to use much math at all?" on Quora.

 **Jeff French** · September 21

Well, I graduated with a math degree, computer science minor, and supplemented that with a nice rack of physics courses. I don't use the computation math skills often, but there is a core of logical thinking, proof theory, and troubleshooting that make me a superior developer. I can look into someone else's work, pull apart their efforts, and find the root problems and errors in their work.

That's what my education taught me:

1. Critical thinking and analysis skills (most of my coworkers are not as strong when it comes to analysis)
2. Troubleshooting skills (I can develop test scenarios to expose complex errors, and then rework the algorithms to be efficient and correct)
3. Generic Research skills (I can read the F\*ing documents, and then make the equipment work, follow the process, fill out the paperwork, etc.)

One of my coworkers didn't get a technical degree, but she can do #3... and just that.. it's enough. It means she stays employed.

Figure 29: What's math got do to with computer science?

## bash Read-Eval-Printing-Loop in the cloud

- In a perfect world, we'd all be working on Linux boxes so that you can do bash scripting at home
- In the real world, you can also use a cloud REPL to learn and practice scripting with bash
- repl.it.com uses a Nix Docker container to emulate the shell (this is another way to get hold of Linux tools...do you remember the others?<sup>8</sup>)
- Login to your Google or GitHub account (the latter is more useful)
- Register with Replit.com

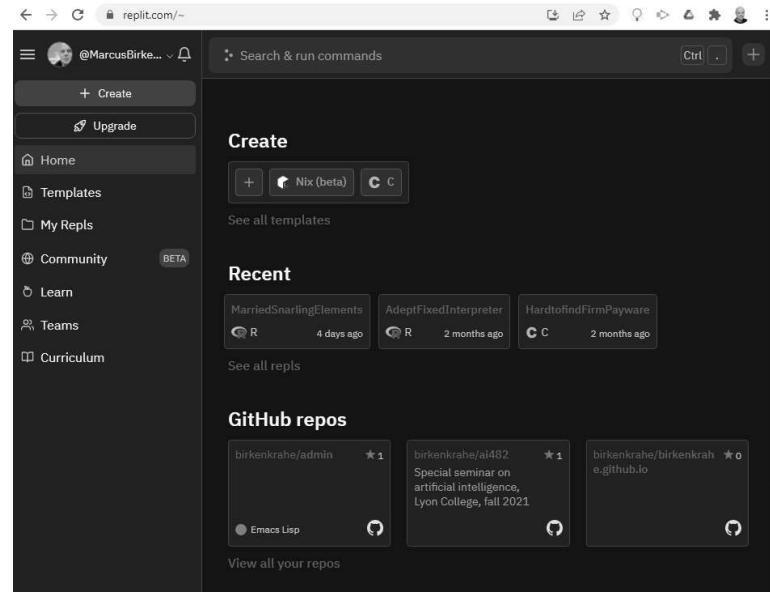


Figure 30: Replit.com startup screen

- Once registered, you can pick among many language templates
- You can create as many REPLs as you like on a free account

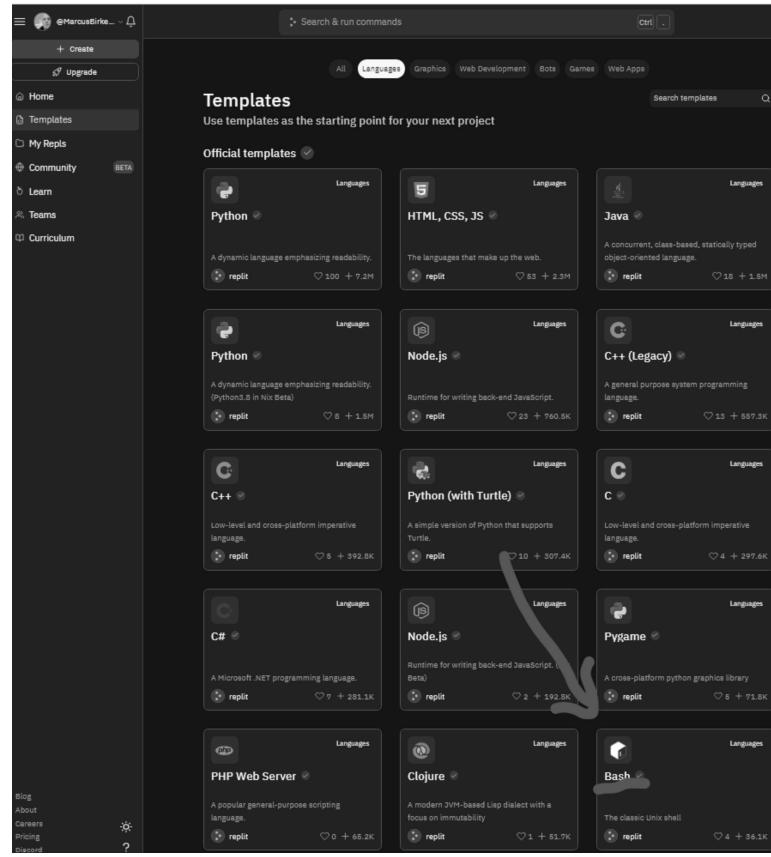


Figure 31: Replit.com language templates

- Pick Bash! Your REPLs are automatically public and shareable.
- For example, to join the bash REPL shown below, [use this link](#).

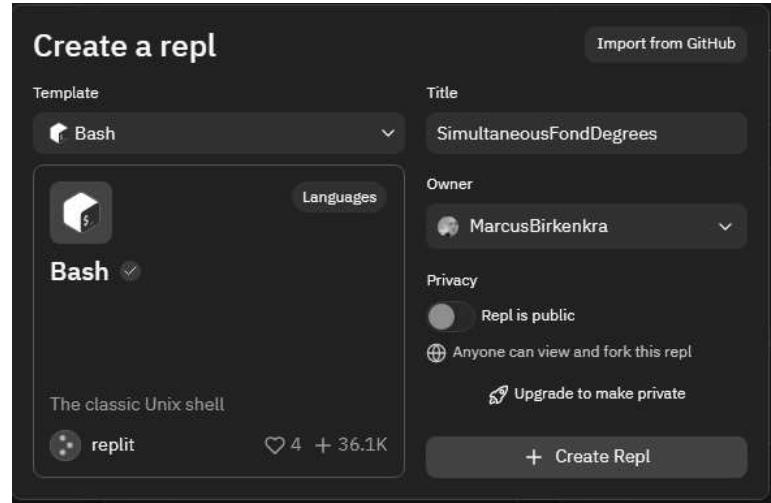


Figure 32: Create a titled, shareable repl

- You can now run any (?) bash script in the REPL

```
main.sh
1 echo Hello World
bash main.sh
Hello World
```

Figure 33: Run bash script in the REPL

- It gets better: you also have access to a shell REPL
- Open the Shell tab. Looks like this:

```
main.sh
1 echo Hello World
-/SimultaneousFond@Degrees:~/Desktop$ cd ..
-/SimultaneousFond@Degrees:~$ ls -l
total 0
-/SimultaneousFond@Degrees:~/Desktop$ ls -a new newdir new-repl.nix
total 0
-/SimultaneousFond@Degrees:~/Desktop$ ls -la
total 0
-/SimultaneousFond@Degrees:~/Desktop$ ls -l .
total 0
-/SimultaneousFond@Degrees:~/Desktop$ ls -ld ..
/nix/store/v81mcxvzyzhgv9c6cmhggp4ngs3ifj-binutils-2.35.1/b
in/MANIFEST
-/SimultaneousFond@Degrees:~/Desktop$ ls -a new new-sym
-/SimultaneousFond@Degrees:~/Desktop$ ls -la
total 16
drwxr-x 1 runner runner 36 Nov 11 22:09 main.sh
drwxr-x 1 runner runner 0 Mar 3 04:10 new
drwxr-x 1 runner runner 0 Mar 3 04:10 newdir
drwxr-x 1 runner runner 62 Nov 11 22:09 replit.nix
-/SimultaneousFond@Degrees:~/Desktop$ ls -d
-/SimultaneousFond@Degrees:~/Desktop$ ls -d ..
```

Figure 34: Use the shell in REPL mode

- Perform a little shell gymnastics:
  1. Check where you are
  2. Check if you can get to the root
  3. Check the operating system
  4. What timezone is the server linked to?
  5. Check your shell program
  6. Go to your home directory
  7. Make another directory and go there
  8. Create an empty file called `file`
  9. Create a symbolic link called `file-sym`
  10. Store the symlink creation command in an alias 'symlink'
  11. Test the alias on another file
  12. Unalias the alias
  13. Go back to your home directory and clean up
  14. Bookmark your REPL before closing the application

## Redirection revisited

- I/O redirection is the coolest command line feature
- Important commands include:
  - `cat` to concatenate files
  - `sort` to sort lines of text
  - `uniq` to report or omit repeated lines
  - `grep` to print lines matching a pattern
  - `wc` to print file newline, word, and byte counts
  - `head` to output the first part of a file
  - `tail` to output the last part of a file
  - `tee` to read from `stdin` and write to `stdout` and files
- `stdin`, `stdout` and `stderr` are special files
- [X] Execute the following commands in your REPL of choice.

## Redirecting standard output

- [X]

It's often useful to store results in a file. What does the command in 1 do?

```
ls -l /usr/bin > ls-output.txt
```

- [X]

Let's look at the file from the outside first.

```
ls -l ls-output.txt
```

- [X]

Let's look at the first and the last part of the file

```
head ls-output.txt
tail ls-output.txt
```

- [X]

We redirect again, this time using a directory that does not exist. This results in an error - but why is it not sent to the file instead of stdout?

```
ls -l /bin/usr > ls-output.txt
```

- [X]

What happened to the output file?

```
ls -l ls-output.txt
```

- [X]

If you ever want to create a new empty file (without touch), you can use this trick:

```
> ls-output.txt
```

- [X]

To append redirected output to a file instead of overwriting it from the beginning, use **>>**. Let's test this:

```
ls -l /usr/bin >> ls-output.txt
ls -l /usr/bin >> ls-output.txt
ls -l /usr/bin >> ls-output.txt
ls -l ls-output.txt
```

## Redirection: stderr, stdin - w9s15 (8-Mar)

### Agenda

- [X] Waking up in AppData/Roaming? [Change your Emacs HOME now.](#)
- [X] How to [Create Symlinks in Windows 10](#) (it's not impossible!)
- [X] Improve your grade with a project ([see FAQ](#))
- [X] Preparations for test 2 (Tue 15-Mar)
- [X] Review Docker/container technology ([notes](#))
- [X] Complete Manipulate Files notebook on your own!
- [X] Review DataCamp ("Batch Processing")

### Preparations for test 2 (Tue 15-Mar)

- Test 2 will only cover questions from quiz 4-6 + new questions.
- You can find quiz 4-6 with solutions + feedback as PDF ([in /quiz](#))
- I will create an update of content Org files ([in pdf/](#))

## Creating symlinks in Windows 10

- I changed my Emacs HOME directory and needed to symlink the .emacs file from there to the old location
- You can do this with the GUI or on the CMD line. Confusingly, the argument order is the reverse from UNIX: mklink [link] [origin] instead of ln -s [origin] [link].

```
mklink C:\Users\birkenkrahe\.emacs C:\Users\birkenkrahe\Documents\GitHub\.emacs
```

```
C:\Users\birkenkrahe>dir *emacs*
Volume in drive C is OS
Volume Serial Number is 0654-135C

Directory of C:\Users\birkenkrahe

03/06/2022  03:09 PM    <SYMLINK>      .emacs [C:\Users\birkenkrahe\Documents\GitHub\.emacs]
03/05/2022  08:18 PM    <DIR>          .emacs.d
01/31/2022  10:00 AM            1,089 .lyonemacs
10/09/2021  01:30 PM            125 emacs.org
01/28/2022  10:21 AM            106 emacs.sh
                           4 File(s)     1,320 bytes
                           1 Dir(s)  345,446,686,720 bytes free
```

Figure 35: symlink example in Windows 10

## Improve your grade with a project (FAQ)

If you want to improve your grade, you can talk to me about doing a small, independent research project leading to a writeup in the form of a notebook, or a short (10-15 min) presentation. The topic must be related to the topic of this course. ([See FAQ](#).)

## Redirection revisited: stderr and stdout

- Download redirection.org from the practice directory in GDrive (Link in Schoology), or from GitHub.
- Code along while working on the file in GNU Emacs.

## Redirection: pipeline commands - w9s16 (10-Mar)

### Agenda

- [X] Complete redirection.org workbook
- [X] Review quiz questions quiz 4-6

## Test review, environment variables - w10s17 (17-Mar)

### News

- [X] Test 2 average result: 74% - not bad at all!
- [ ] Improving your grade after test 2 with a project
- [ ] Motivation to learn bash scripting by Steve Yegge

### Test 2 review

- [ ] How can you start a command on the commandline and still use that terminal? Answer: add & after cmd - process goes in background
- [X] When you type sudo before a command, you become root? Answer: No. You only get some system privileges associated with root.
- [ ]

Which command tees standard out off to an intermediate file? Answer: the tee command. Example in 1.

```
ls . | tee ls.txt | grep org | wc -l
```

8

- [ ]

How do you append standard error and standard output to the same file? Answer: you need &>> or >> and 2>1&. Example in [1](#).

```
ls -l /etc/passwd >> out1.txt 2>&1
ls -l /etc/os-release &>> out2.txt
```

## Improving your grade with a project - ideas

- Create an Org-mode file with tutorial examples for one or more shell commands - pose a problem and present a solution
- Present a bash script of your own design e.g. to automate a process (my example [git.sh](#): updating Git version control - this could be improved quite a bit with control flow).
- Expand an explanation from class, or a topic from class and write it up as a short essay, or create a presentation and screencast it, or (better) present in class
- Whatever you do is not for the lecturer but for the class, so you need to share it with everyone (e.g. YouTube or GDrive and then post an issue in GitHub)

## Bash scripting

"You have to know command-line interfaces and shell scripting (or equivalent) in order to be a great programmer and a great generalist. [...] The bottom line is that sometimes you need to do custom batch (and/or streaming) tasks that are not handled by your IDE or your favorite tools. So what do you do? Scripting to the rescue." (Steve Yegge)

Watch [Stevey's Tech Talk E37: sed -i 's/sorry/not sorry/g'](#). (2022)

Yegge is the guy from the original [Google-Amazon rant](#) (Couts, 2011)

## Environment and editors

- See interactive notebook [environment.org](#) in GDrive

## Permissions - w11s19 (29-Mar)

### Plan for the remainder of the term

- Week 11: Permissions / Intro to scripting
- Week 12: Process management / Variables
- Week 13: Customizing the prompt / Control
- Week 14: Networking / Functions
- Week 15: Scripting project?

## vi editor - w11s20 (31-Mar)

## Process management - w12s21 (5-Apr)

- [ ] How a process works
- [ ] Viewing processes statically and dynamically
- [ ] Controlling processes with signals
- [ ] Interactive notebook (GDrive)

## Warming up with pipelines

- [ ]

Add the string "(setq-default org-hide-emphasis-markers t)" to your \$HOME/.emacs file so that you don't see the emphasis markers in Org-mode. Then show the last line of the file to make sure it worked.

*Tip: you need the commands echo, ~>>, tail.*

## Cygwin assignment - w12s22 (7-Apr)

- [ ] [Cygwin assignment / controlling processes](#)
- [ ]

[DataCamp assignment](#)

# Cygwin

*Get that Linux feeling - on Windows*

## Introduction to bash scripting - w13s23 (12-Apr)

### Emacs tip

To **not** see the emphatic characters like ~ or \* or / in the Org file text, run the following code chunk (or put the code in your `/.emacs` file): if successful, you should see "t" in the minibuffer.

```
(setq-default org-hide-emphasis-markers t)
```

If you don't put it in your `/.emacs` file, the command will only work for the current Emacs session.

### DataCamp review

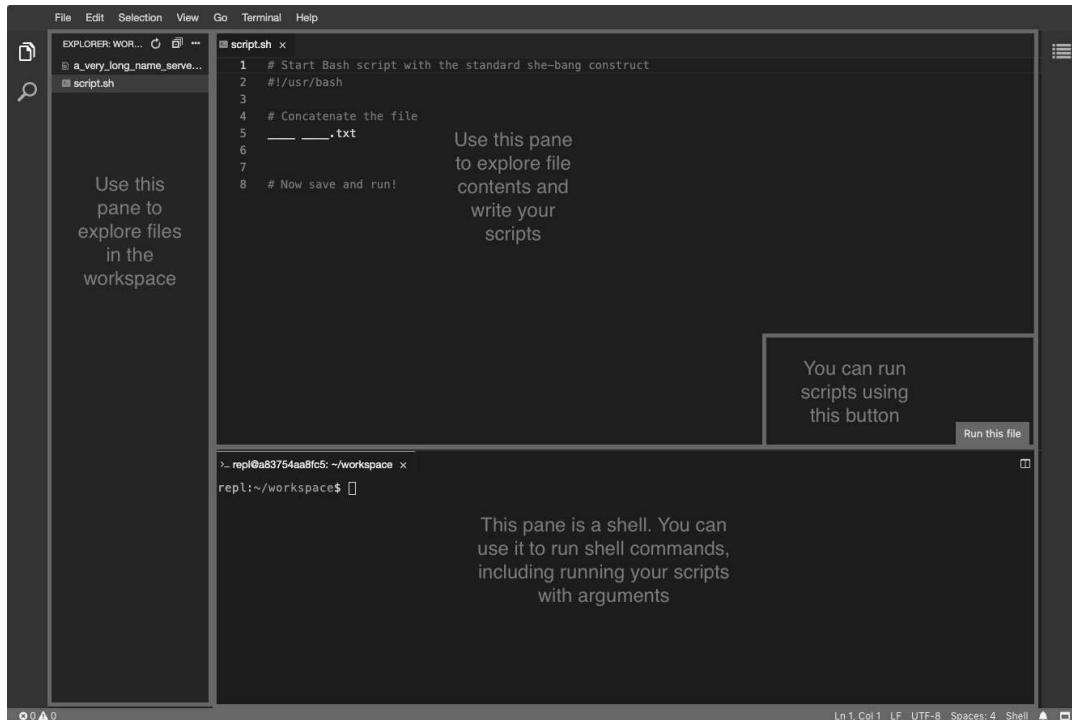


Figure 37: DataCamp bash IDE

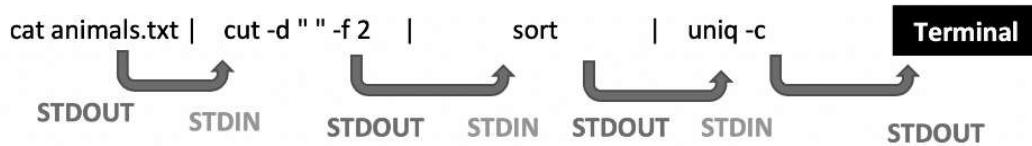


Figure 38: DataCamp pipe illustration

- UNIX stream editor sed to change one pattern into another.

```
cat file.txt | sed 's/Birkenkrahe/birchcrow/g' | grep 'Birkenkrahe'
```

## Scripting in bash - refresher

- [X] How to write a shell script in 3 steps
- [X] Script file format with *shebang*
- [X] Permission to execute with chmod
- [X] Location with \$PATH

## Bash scripting lab practice - w13s24 (14-Apr)

- We use the online bash installation [replit.com](https://replit.com) for this practice.
- You find the practice notebook in GDrive (practice directory)
- All exercises are directly or indirectly taken from the first two DataCamp chapters of
- [X] Revising shell commands
- [X] Standard streams and arguments
- [X] Single and double quotes
- [X] Shell within a shell
- [X] Numeric variables and arithmetic on the shell
- The exercises with bc cannot be completed in replit.com or in Cygwin, you need a native Linux installation for that (like Raspbian on the Raspberry Pi)

## Networking - w14s25 (21-Apr)

- [X] Last two **DataCamp** assignments for EXTRA CREDIT only
- [X] **Quiz 7-9** are live (bash scripting/permissions/processes)
- [X] Play the quizzes by April 22 (Friday) 4 pm
- [X] **Test 3** will be online - open Thu-28-Apr to Fri-28-Apr
- [X] **Test 3** will not have any new questions - only Q7-Q9
- [X] **Topic** today: Networking under Linux

## Networking

- [X] ping other network hosts
- [X] traceroute to print network travel routes
- [X] ip to show/manipulate network devices
- [X] netstat to print network stats
- [X] ftp for internet file transfer

<<<<< HEAD

- [ ] wget for non-interactive network downloads
- [ ] ssh for remote shell login

=====

- [X] wget for non-interactive network downloads
- [X] ssh for remote shell login

>>>>> eb3b79c (update)

See workbook (for use on the Raspberry Pi) in GDrive.

## Bonus features

- [X] How to connect your Pi to GDrive using `rsync`
- [X] Emacs special: using `Dired` and `shell` remotely with `Tramp`

<<<<< HEAD

## Review quiz 7-9 - w15s26 (26-Apr)

### Plans for the last weeks of term (Schoology update)

- On **Tuesday, April 26**, we will finish our look at networking with `ssh` and `wget`, and review the 3 quizzes for *test 3* (scheduled for online submission Thursday 28-April 11 am to Friday 29-April 11 am).
- On **Thursday, April 28**, we will start with finding, archiving and compressing files under Linux.
- On our last meeting, on **Tuesday May 3**, we will talk about the final exam (scheduled for Thursday 5-May 6 pm to Saturday 7-May 8 am), summarize what we've achieved together, and look at compiling source code using the [GNU make\(1\) program](#).
- You will get a detailed list with curated links and other sources from me for the DIY continuation of your OS journey (@ end of May), and a *bonus video* if my muse cooperates and if your evaluations are mostly positive :-)



Figure 39: training for finals in penguin world

### Course evaluations - open until May 5

- [X] Made a lot of changes after the last evaluations.
- [X] Support the development of this course
- [X]

Help me and Lyon College decide if I stay or if I go

**Extra credit** for evaluating! - Don't forget to click the box.

Enable Send Proof:  Allow Students to send proof of survey completion. ?

Participation Incentives:  Enable Contest Incentives for this survey

This option allows Students to send an email receipt to Responsible Faculty from their completed survey list. This is for anonymous surveys only.

Allow Faculty to leave

## Review quiz 7-9

- [X] Quiz 7: Permissions
- [X] Quiz 8: Processes
- [X] Quiz 9: Bash scripting
- Test 3 will have 15 questions and last 30 minutes
- Open between Thu 6 pm to Sat 8 am
- ALL questions for test 3 and final exam are KNOWN (in principle - I will make small modifications)

## Networking 2 - remote work - w15s27 (28-Apr)

## "Microsoft finds Linux desktop flaw that gives root to untrusted users"

- [ ] Saved as a [GitHub issue](#)
- [ ]

Is this the end of Linux and a new dawn for Microsoft?

»Vulnerabilities recently discovered by Microsoft make it easy for people with a toehold on many Linux desktop systems to quickly gain root system rights—the latest elevation of privileges flaw to come to light in the open source OS.«

- [ ]

Exhibits the strength of Open Source and independent development:

»The proof-of-concept exploit works only when it can use the “org.freedesktop.network1” bus name. The researcher found several environments where this happens, including Linux Mint, in which the systemd-networkd by default doesn’t own the org.freedesktop.network1 bus name at boot. [...]»

The researcher also found several processes that run as the systemd-network user, which is permitted to use the bus name required to run arbitrary code from world-writable locations. The vulnerable processes include several gpgv plugins, which are launched when apt-get installs or upgrades, and the Erlang Port Mapper Daemon, which allows running arbitrary code under some scenarios.

The vulnerability has been patched in the networkd-dispatcher, although it wasn’t immediately clear when or in what version, and attempts to reach the developer weren’t immediately successful. People using vulnerable versions of Linux should patch their systems as soon as possible.« ([Goodin, 26 April 2022](#))

»Fixes for these vulnerabilities, now identified as CVE-2022-29799 and CVE-2022-29800, have been successfully deployed by the maintainer of the networkd-dispatcher, Clayton Craft.« ([Or, 26 April 2022](#))

- [ ] Used in this article and covered in this course:
  - boot process and root privileges
  - systemd and networkd daemon processes
  - OS Linux distro dependencies
  - UID, PID, GID
  - Linux file system /sbin /tmp /etc
  - Shell script
  - Symlinks
  - Environment variables
- [x] From the [Microsoft Defender for Endpoint](#) "real-time threat detection" platform commercial video (6 min):
  - Old news: "Allows only trusted applications to run" etc.
  - More stuff from the Microsoft software propaganda machine for the clueless and scared (= not for you)
  - Automatic threat detection and forensic evidence = do not (only) trust dashboards ("uses inspection algorithms that an analyst would use")
  - Service offering
  - "We encourage you to read the documentation on Microsoft docs to learn more."
- [ ] **What to do?** Embrace the BS - connect with Microsoft (subscribe to their channel, network with their experts) and continue to grow your Linux knowledge using Open Source technology (100% transparency).

## Networking 2 - wget, ssh, scp and sftp

### DOWNLOAD UPDATED FILE networking\_2.org FROM GDRIVE

- [x] Did anyone look at the "Bonus features" (`rclone / Tramp`)?
- [x] See also "[Remotely accessing your Raspberry Pi](#)" (networking)
- [x] Borrow a Pi if you must (but do tell me about it)
- [X] `wget` for non-interactive network downloads
- [X] `ssh` for remote shell login
- [ ]

#### Bonus challenge: what is "Raspbian" (technically)?

"Raspbian" was the former name of what is now called "Raspberry Pi OS". It is a Debian-based Linux Operating System for the Raspberry Pi single-board computer series. In turn, Debian is one of the oldest operating systems with a Linux kernel (first released in 1993). Raspberry Pi OS is Debian + the Raspberry Pi Desktop, which is based on Gnome (one of two popular desktop architectures - the other one is KDE<sup>2</sup>). (Source: [raspberrypi.com](#))

## Deadlines

- [ ] Final exam from **Thu-5-May 6 pm to Sat-7-May 8 am**
  - Random draw from all questions + minor modifications

- Quizzes and tests will be disabled before the exam

## Compiling from source with `make` - w16s28 (3-May)

### Content warning

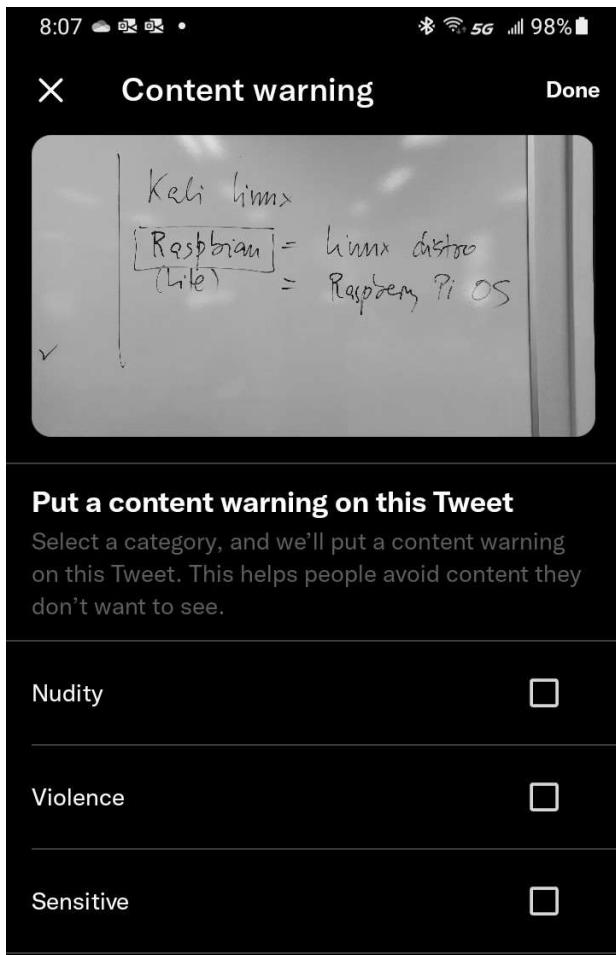


Figure 41: Content warning on my phone!

### Topics

- [X] See practice file `compiling.org` in GDrive
- [X] Works well with Cygwin (except the last step?)
- [X] What is compiling?
- [X] Source code compilation pipeline
- [X] Final exam review
- [X] Any outstanding work must be finished by Monday May 9
- [X]

Please take 10 minutes to complete the course evaluation now



## Final exam - w16s29 (Thu 5-May 6-8 pm)

COURSE NO	MEETS	TIME	ROOM
CSC 420	TR 9.30a	Thu 5-May 6.00p - 8.00p	Lyon 104
		Fri 6-May 1.00p - 3.00p	Derby 209
		Fri 6-May 3.30p - 5.30p	Derby 209
		Mon 9-May 1.00p - 3.00p	Derby 209
		Mon 9-May 3.30p - 5.30p	Derby 209

The final exam will take place **in room 104 of the Lyon building** (our usual classroom) on **Thursday, May 5 from 6.00 pm to 8.00 pm** ([see exam schedule](#)).

The exam will consist of 40 questions drawn at random from the pool of quiz and test questions. Completing the various quizzes and tests until you've reached 100%, and revisiting your past tests should enable you to pass this exam with flying colors!

Please reach out to me if you have difficulties taking the exam at this time. I'm going to offer a limited number of seats to take this exam on Friday May 6 or on Monday May 9. Graduating seniors must complete the exam no later than May 8.



Figure 43: penguin drop off

## References

- Couts (October 21, 2011). Google engineer who publicly bashed Google+: ‘Nothing bad happened’ [blog]. [URL: digitaltrends.com](#).
- element14 (n.d.). Identifying Your MOdel of Raspberry Pi [blog]. [URL: community.element14.com](#).
- Janssens (2021). Data Science at the Command Line (2e). O'Reilly. [URL: datascienceatthecommandline.com](#).
- Shotts (2019). The Linux Command Line (2e). NoStarch Press.
- Steve Yegge (Feb 21, 2022). Stevey's Tech Talk - sed -i 's/sorry/not sorry/g' [video]. [URL: youtube.com](#).
- Steve Yegge (Mar 7, 2022). Stevey's Tech Talk - A guided tour of Emacs [video]. [URL: youtube.com](#).
- Vaughan-Nichols (9 Feb 2022). Best desktop Linux for pros 2022: Our top 5 choices [blog]. [URL: www.zdnet.com](#).
- Wikipedia (25 Dec 2021). The Treachery of Images [website]. [URL: wikipedia.org](#).

## Footnotes:

<sup>1</sup> Unix manual pages are formatted with `troff`, the Unix "typsetter roff", a document processing system. "roff" stands for "I'll run off a document".

<sup>2</sup> An alternative (that takes much longer, because many files have to be downloaded, and the Pi 3B+ isn't the strongest networker) is to clone the entire `os420` repository with the command:

```
$ git clone https://github.com/birkenkrahe/os420
```

<sup>3</sup> A "distro" is a Linux distribution, like Ubuntu or Raspbian, or Kali. Distro design and distro news bind a lot of fanboy energy, see e.g. ["Best desktop Linux for pros 2022"](#) (Vaughan-Nichols, 2022).

<sup>4</sup> Windows likes to call directories 'folders' - another unnecessary dumbing down. A folder is just a container, but a directory (like the "telephone directory") has a data structure, an index, labels, and serves to search and find. Don't say "folder".

<sup>5</sup> Answer: `ls /` lists the root directory, because `/` is root's home. The particular ordering and naming of these directories is up to the Linux OS file administrator, but there are certain style rules that are usually obeyed by every professional.

<sup>6</sup> I have not tried this but I would have thought that the Google search bar permits use of wildcards.

<sup>7</sup> In the language R, both of these are combined as `|>` because they really are the same thing:

```
mtcars |> head(1)
```

```
mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21 6 160 110 3.9 2.62 16.46 0 1 4 4
```

Output:

```
: mpg cyl disp hp drat wt qsec vs am gear carb
: Mazda RX4 21 6 160 110 3.9 2.62 16.46 0 1 4 4
```

<sup>8</sup> (1) Virtualbox with a Linux image; (2) Cygwin utility tool bundle; (3) dual boot with Linux e.g. from a USB Stick or an SD-card; (4) Installing the Ubuntu app from the Microsoft store. (5) eshell (emulated Lisp shell in Emacs), (6) repl.it.com (online cloud Linux), (7) download and install a Docker image/container. If you're serious about learning OS, you should give all of these a try just for fun (Research + analysis skills)! Also, any of these will enable you to practice many of the commands you learn in this course. And when a command does not work, you'll learn why (troubleshooting skill!).

<sup>9</sup> The choice of visual desktop environment (Gnome or KDE) parallels other choice-wars, like for editors (Emacs vs. vi). KDE (the "Kool Desktop Environment") comes from Germany, Gnome comes from the Americas.

Author: Marcus Birkenkrahe

Created: 2022-05-03 Tue 15:21