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 (01/11/22)

- 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)

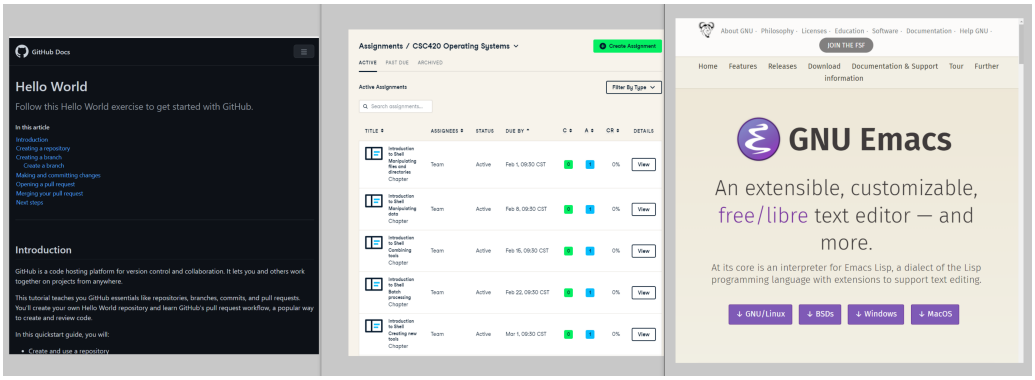
CLASS	CODE	TERM	Topics
Data Science Tools and Methods	DSC 101	Fall 2021	R, Basic EDA, Base R
Introduction to Advanced Data Science	DSC 205	Spring 2022	R, Advanced EDA, Tidyverse
Database Theory and Applications	CSC 330	Spring 2022	SQL, SQLite
Operating Systems	CSC 420	Spring 2022	Bash, awk, sed, regular expressions
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

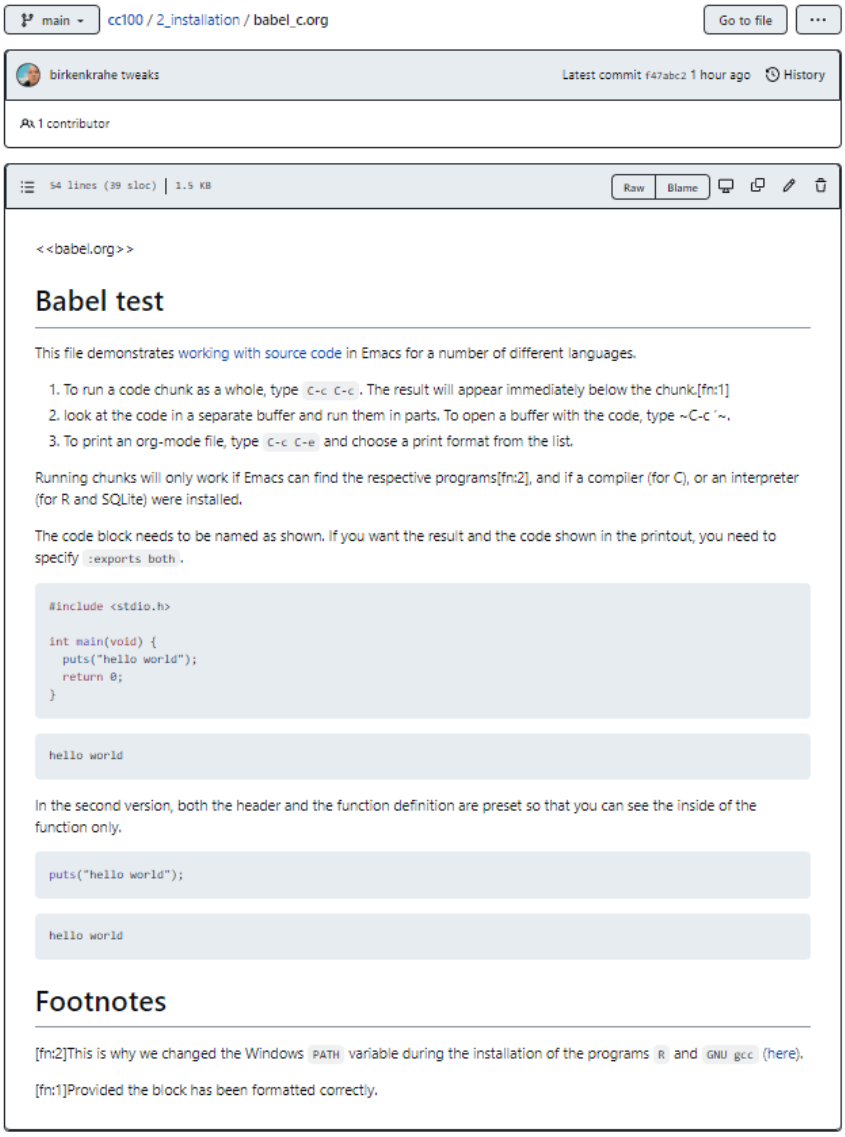
- What is it?
  - Software development platform
  - Built around Git by Linus Torvalds
  - Bought by Microsoft in 2018
  - AI support (e.g. GitHub Copilot)

  Watch: "What is GitHub?" (GitHub, 2016)

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

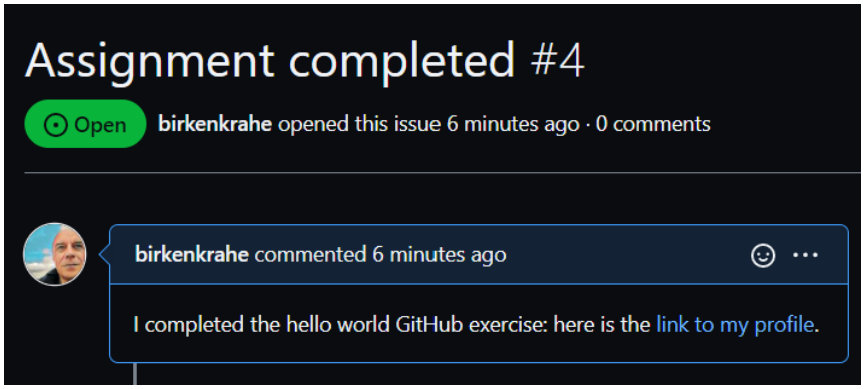
- Why are we using it?

  Image: Org-mode file in GitHub



  - 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).

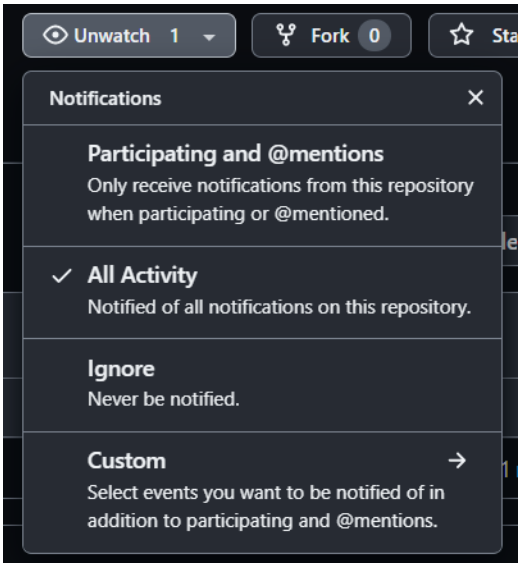
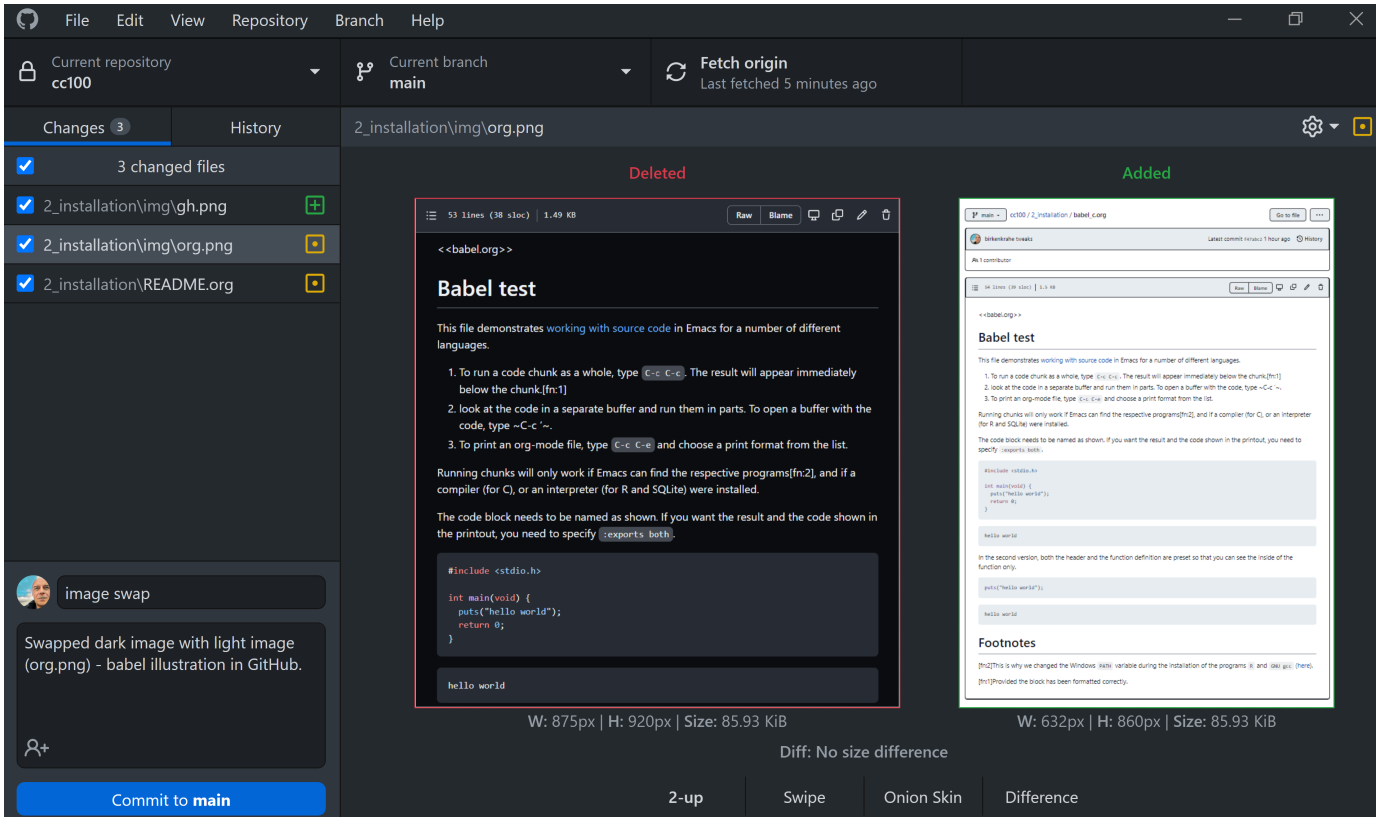


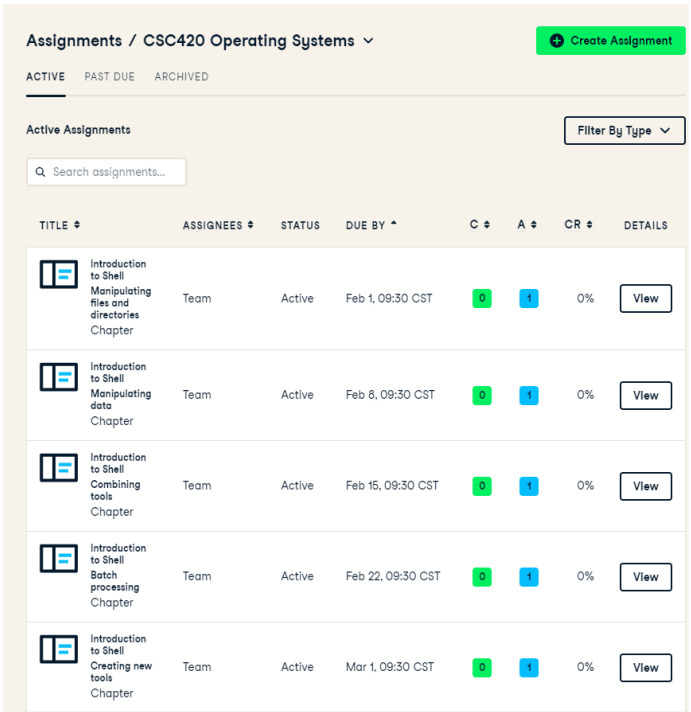
Image: Notifications settings when watching a repository

  - 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



- 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 (01/13/22)

Overview

HOW	WHAT
Review	GitHub Hello World exercise (FAQ)
Lecture	What operating systems do
Practice	Install GNU Emacs (FAQ)
Demo	Emacs guided tour
Self	Work through the Emacs onboard tutorial (18p)

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 (01/18/22)

Overview

HOW	WHAT
Review (S)	Quiz: course, OS foundations / GNU Emacs
Resource (S)	Fundamentals of Operating Systems YouTube playlist
Lecture (S)	Interrupts / I/O operations example / bootstrapping
Demo	Emacs guided tour
Self	Work through the Emacs onboard tutorial (18p)

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 (01/20/22)

Overview

HOW	WHAT
Lecture	Storage structure & OS management tasks
Practice	Emacs guided tour (tour)(tutorial)

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 (01/25/22)

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

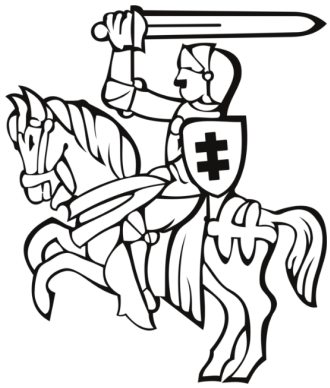


Figure 9: Unix the white knight of Operating Systems

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

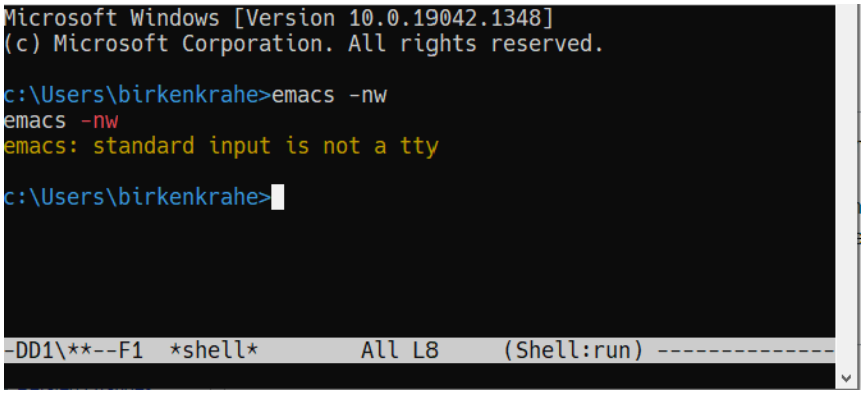
SHELL COMMAND	CHARACTERISTICS	MODELINE
`M-x shell`	Windows shell `CMD.exe`	`*shell*`
`M-x eshell`	Lisp simulated shell	`*eshell*`
`M-x term`	Terminal emulator	`*terminal*`

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

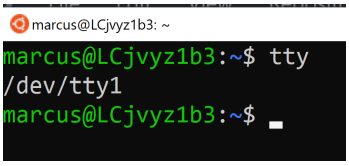
- Start Emacs from the terminal: `emacs -nw -q`
- Start all three shells in Emacs

- 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).



On Linux (Ubuntu App in Windows 10):



Assignment: hello world!

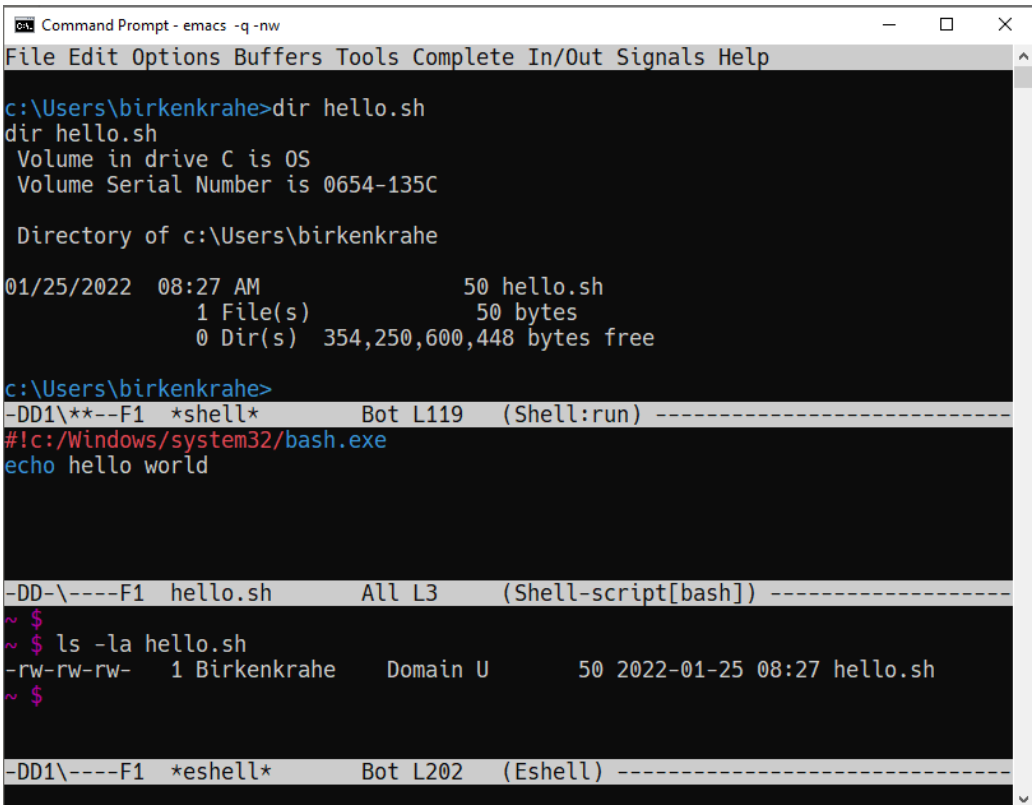


Figure 12: Windows CMD shell in Emacs and hello world pgm

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 (01/27/22)

Overview

HOW	WHAT
Review	`hello.sh` assignment
Fix	`.emacs` issue (https://tinyurl.com/lyonemacs)
	Find the file on the lab desktop
	Put it into the `$HOME` 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

1. Open the box in front of you
2. Take out the Pi and connect it to the KVM switch
3. Connect the Ethernet cable to the Pi
4. Plug in the power cord and then plug it into the Pi
5. Install using NOOBS
6. Get the password from me!
7. Open a terminal app on the Pi
8. In terminal, enter `sudo raspi-config`
9. In the configurator, change boot options > Desktop/GUI to `Console autologin`, then <TAB> to <Finish>
10. Reboot the Pi (`sudo reboot`)
11. Login as user `pi` with password
12. Check network connection with `ifconfig`
13. Update OS with `sudo apt update`

14. Upgrade OS with `sudo apt upgrade -y`
15. Install Emacs with `sudo apt install emacs`
16. Start emacs, check it and exit again
17. Finish session with `sudo shutdown now`
18. Unplug the power chord, then the KVM connections
19. Return Pi to box
20. 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 (02/01/22)

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 interpretes 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` [1].
- [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 - w4s8 (02/08/22)

Overview

HOW	WHAT
Test info	Test on Thursday, Feb 10 at 10.00-10.45 AM

HOW	WHAT
Setup	USB/HDMI > power > switch > `startx`
Review	Shell warm-up exercise (15 min)
Poll (vote)	Should you update your operating system?
Practice	`sudo apt update -y` [update OS]
	`sudo apt upgrade -y` [upgrade OS]
	`sudo apt autoremove -y` [remove old OS]
Lecture	The Linux File System ("Everything is a file")
Practice	Navigating the file system (DataCamp)
	Manipulate files and folders (DataCamp)
	Start an interactive notebook `bash.org`
Shutdown	`sudo shutdown now` > USB/HDMI > Power > switch

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. `-l`) 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[2]

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 (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
- Manipulating files and data
- Raspberry Pi history & hardware
- New DataCamp assignment due Feb 15 ("Combining tools"))

NEXT w4s9 (02/10/22)

Overview

HOW	WHAT

HOW	WHAT
Lecture/demo	Raspberry Pi - the hardware & the history 2
Test 1	10:00-10:45 AM

Objectives

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

Manipulating files and directories

- Files are organized in a hierarchical directory structure[3]
- Imagine you stand somewhere inside an upside down tree:
  - the `pwd` command tells you were 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 14: upside down tree

- [ ]

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

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

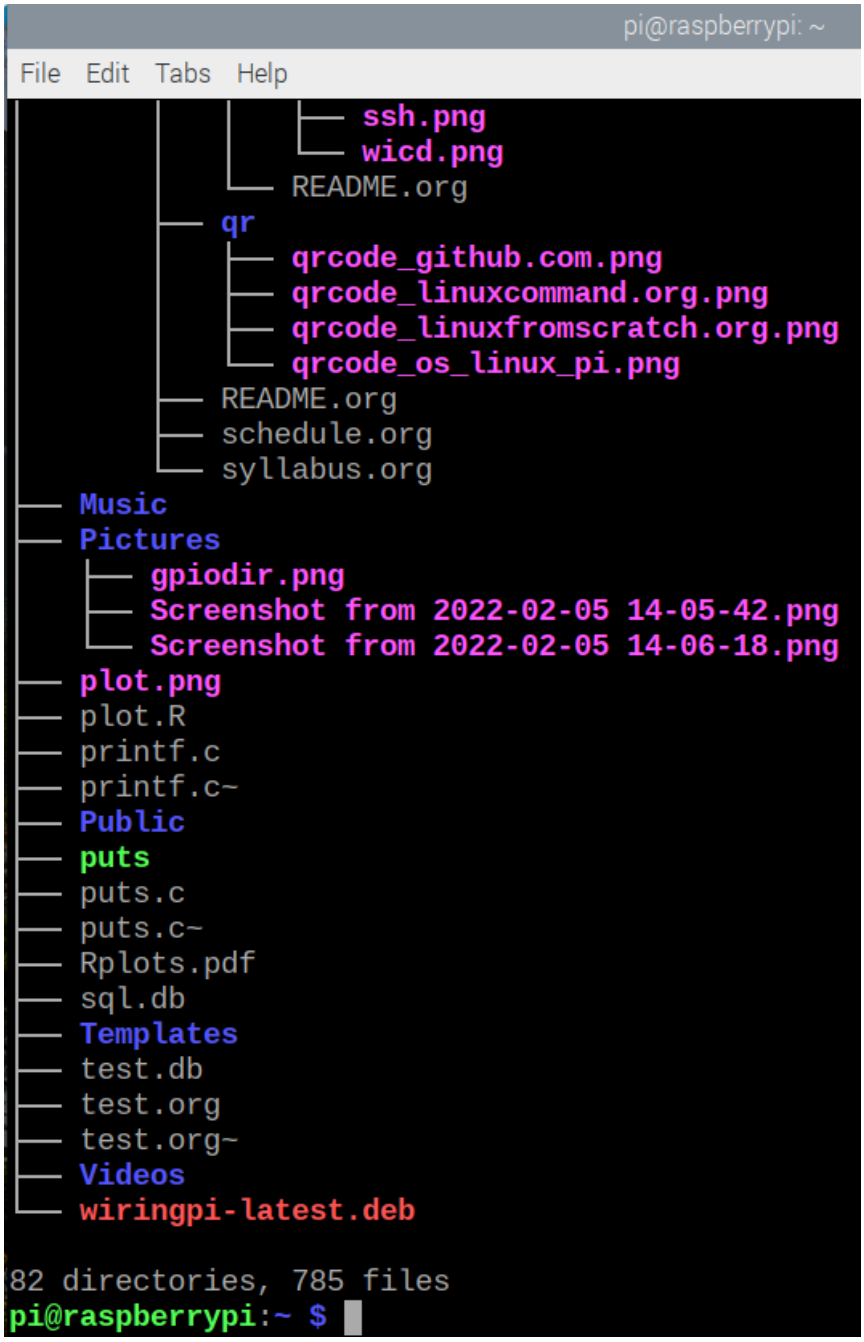


Figure 15: Linux tree command (screenshot)

- 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.
- `[ ]` Compare `ls`, `ls -l`, `ls -la`, `ls -lF`, and `ls -laF`
- `[ ]` Which command would show the top directories of the OS?[4]
- To create an empty file, use `touch`
- `[ ]` 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`

IN PROGRESS Manipulating data

Let's begin working inside the interactive notebook!

What's next

- Raspberry Pi hardware - GPIO pins and how to control them

TODO w5s10 (02/15/22)

Overview

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

Objectives

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

Raspberry Pi - hardware & history: hardware

- Raspberry Pi history

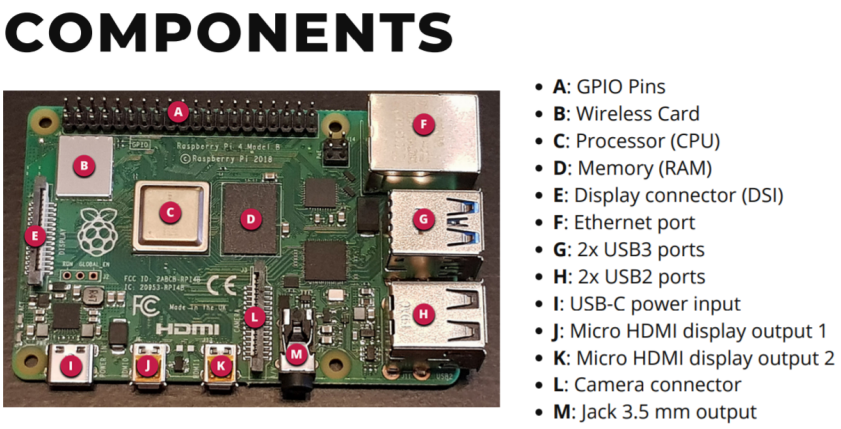


Figure 16: Raspberry Pi 4 Board (Source: raspberryptips.com)

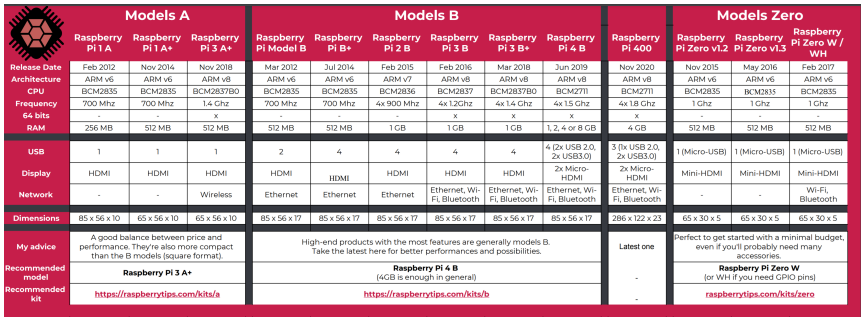


Figure 17: Raspberry Pi 4 Board (Source: raspberryptips.com)

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

TODO Combining shell tools

What's next

- Test 1, Thursday 10 February 2022, 10:00-10:45

References

- element14 (n.d.). Identifying Your MOdel of Raspberry Pi [blog]. URL: community.element14.com.
- Shotts (2019). The Linux Command Line (2e). NoStarch Press.

Footnotes:

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

[2] 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
```

[3] 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".

[4] 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.

Author: Marcus Birkenkrahe
Created: 2022-02-08 Tue 20:12
Validate