File management and command line basics

Getting to know your machine

NEH Institute “Advanced Digital Editing”
Week 1, day 2
July 12: 2.30pm - 3.30pm

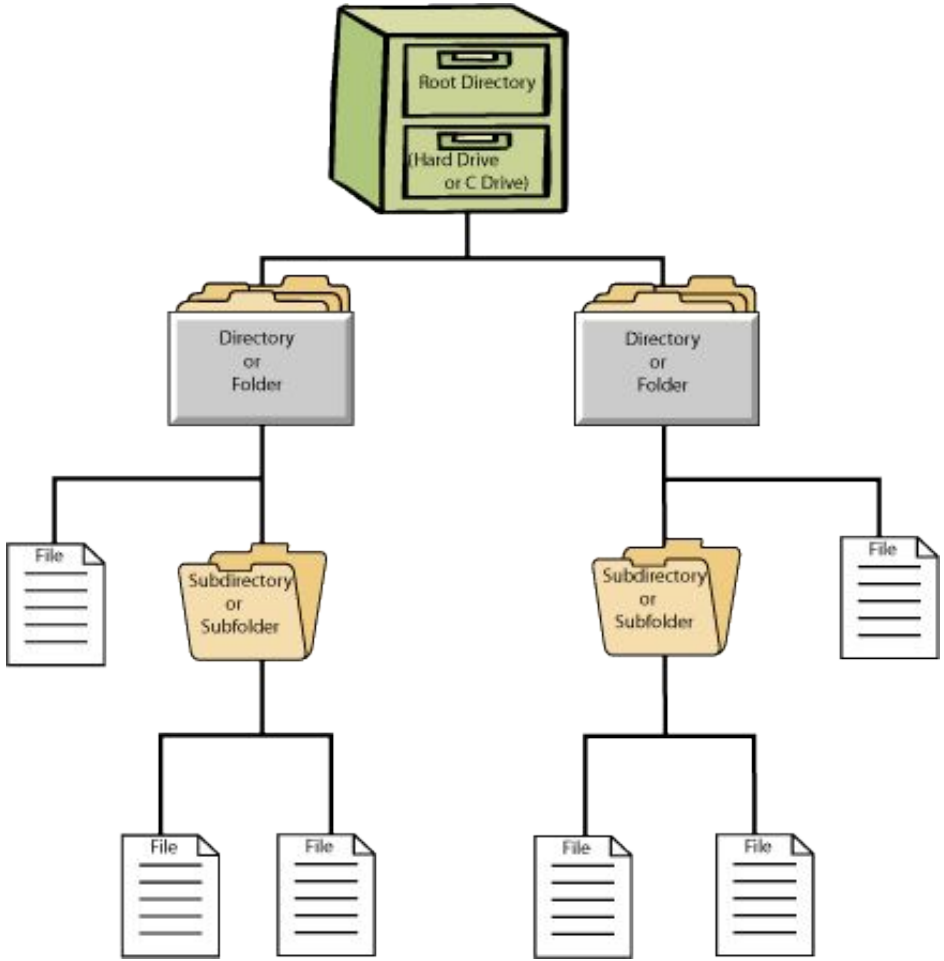
Outcomes

- File system hierarchy, file system hygiene principles
- Bash shell
- Moving around on your machine
- Troubleshooting, looking stuff up

1. File system

File system

- Responsible for managing information on your disk: naming, storing, retrieving
- Multi-level hierarchical structure
  - Information is stored in **files** which are stored in **directories** (or **folders**).
  - Directories can contain other directories.
  - Home directory (`Users/<your-username>` on Mac, `C:\Users\<your-username>` on Windows) functions as a root
- Every file has a unique path that is its location


Root Directory
(Hard Drive
or C Drive)
Directory
or
Folder
Directory
or
Folder
File
Subdirectory
or
Subfolder
Subdirectory
or
Subfolder
File
File
File
File
File

File system

Show/hide file name extensions

- Mac: Open Finder and select Preferences, click “Advanced”, and check the box next to “Show all filename extensions”.
- Windows: In any File Explorer window, click on “View” tab, and then “Options” on the right to open the “Folder Options” window. In the “View” tab, uncheck “Hide extensions for known file types” box.

Show/hide hidden files

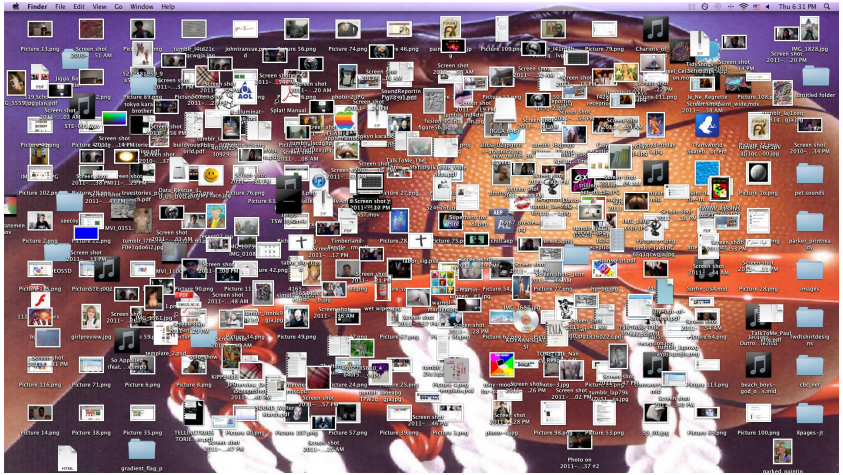
- Mac: Open Finder and hit Cmd+Shift+. Do the same thing to turn off hidden files
- Windows: In the same “Folder Options” window as before, Check “Show hidden files, folders, and drives”

File system hygiene

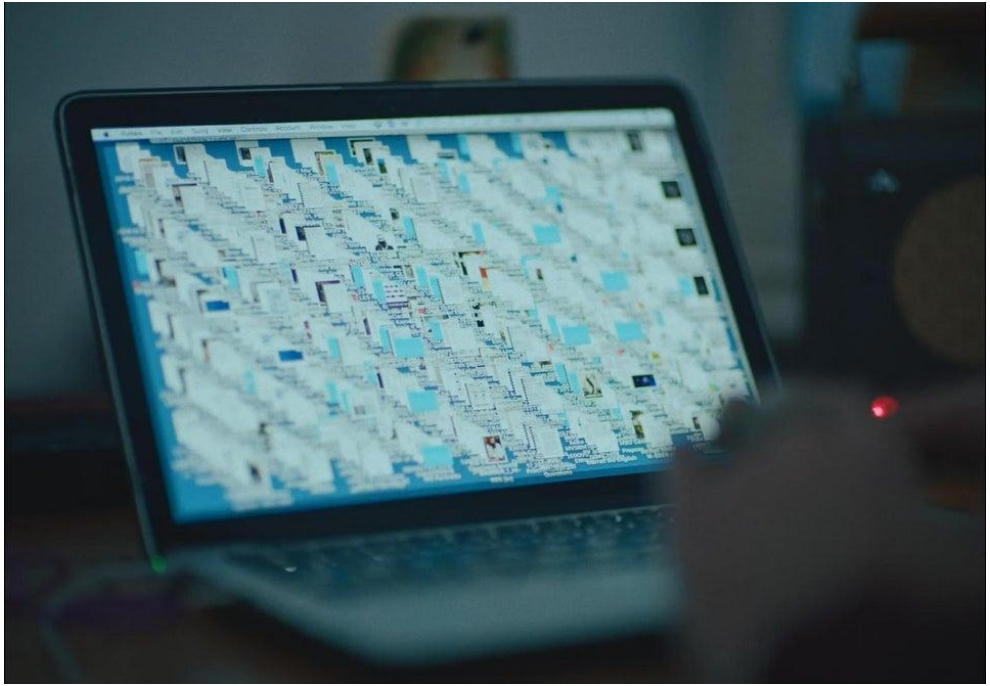
File system hygiene

Walt Whitman by Dr. William Reeder, 1891.
Image from the Walt Whitman Archive.

File system hygiene



“Messy desktop.” Image used in the Command Line Fundamentals workshop by David J. Birnbaum, Gabi Keane, and Emma Schwarz.



Warren Ellis’ desktop. Still from the documentary *This much I know to be true* by Andrew Dominik (2022).

File naming conventions

- Stay organized, quickly find files
- Human readable
  - Logical, original names that represents the content
  - Be nice to your future self

XKCD, https://xkcd.com/1459.

File naming conventions

```
01_marshal-data.md
01_marshal-data.r
02_pre-dea-filtering.md
02_pre-dea-filtering.r
03_dea-with-limma-voom.md
03_dea-with-limma-voom.r
04_explore-dea-results.md
04_explore-dea-results.r
90_limma-model-term-name-fiasco.md
90_limma-model-term-name-fiasco.r
Makefile
figure
helper01_load-counts.r
helper02_load-exp-des.r
helper03_load-focus-statinf.r
helper04_extract-and-tidy.r
tmp.txt
```

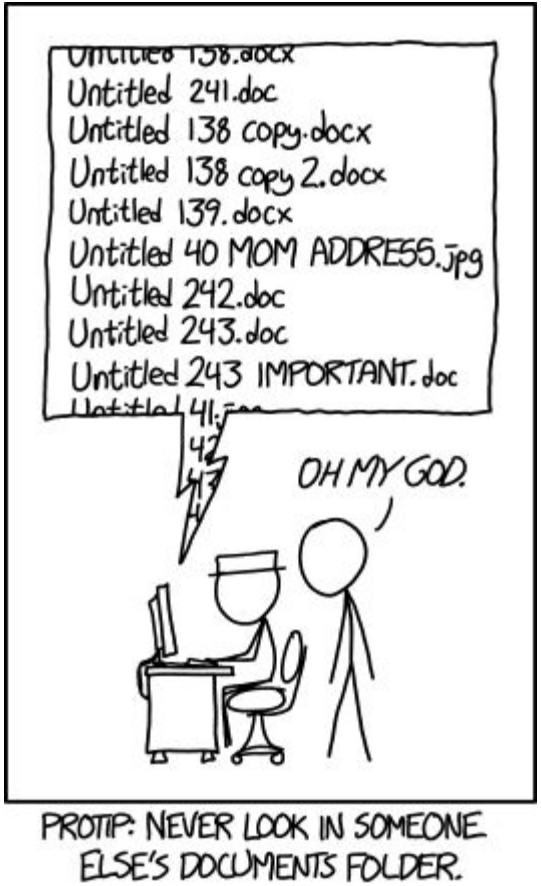
OR

```
01.md
01.r
02.md
02.r
03.md
03.r
04.md
04.r
90.md
90.r
Makefile
figure
helper01.r
helper02.r
helper03.r
helper04.r
tmp.txt
```

From data carpentry.

File naming conventions

- Stay organized, quickly find files
- Human readable
  - Represents the content
  - Be nice to your future self
- Machine readable
  - No spaces
  - No punctuation
  - No accented characters
  - Think about capitals vs. lower case
  - Deliberate use of delimiters (- and _)



XKCD, https://xkcd.com/1459.

https://github.com/Pittsburgh-NEH-Institute/Institute-Materials-2020/tree/5e783388dcf8011c 133%

Code | Issues 4 | Pull requests | Actions | Projects | Security | Insights | Settings

5e783388dc ▾ Institute-Materials-2020 / presentations / Go to file

ararebit Added session notes for Session 1-1 Edition Goals ✓ 5e78338 12 minutes ago History

..

sources	Created directory for for optional sources of all sessions' presentio...	2 days ago
1-1-presentation.md	session 1-1 presentation notes	2 hours ago
Day1_Session2_RonaldHD_Computational_...	Added the slides of the Computational Pipelines presentation of day 1...	7 hours ago
README.md	Created directory for pdf-versions of all sessions' presentions. READ...	2 days ago
day01_session02-ljo-transformation_and_...	Adding pdf for day01_session02_slot03.	2 hours ago
day1_session1_rowell_edition-goals.md	Added session notes for Session 1-1 Edition Goals	12 minutes ago

README.md

Presentations for the sessions

File naming conventions

- Spaces and punctuation are meaningful on the command line

  `Long program name`

- When trying to describe the location of a file, using a URL, spaces need to be escaped (`Long\ program\ name`)
- Spaces are encoded or converted to %20 in file names on the web

  `Long image name 1` **>>** `Long%20image%20name%201`

- Special characters like `/ ! | $ < > ? *` have a function on the command line

2. Shell

What is the shell?

- The shell is a program that runs other programs
- We use the shell to interact with the computer on the command line (CLI ~ GUI)
  - It gives you better control
  - You can pipe together small simple commands to do something complex
  - It works on every platform / OS
- A command is a software that is executed on the CLI and performs an action on the computer

Open your shell

- On Mac: terminal application
- On Windows: Git bash

3. Basic commands

Navigating the file system

Commands

`pwd`	print working directory
`ls`	list directory
`cd`	change directory
`less`	view a file
`whoami`	returns your username
`/`	file system root
`~`	user home directory

Navigating the file system

Commands and parameters (“flags”)

`cd ~`	change into your home directory
`cd ..`	move one level up
`cd Documents`	change to Documents folder

Also: tap completion!

Navigating the file system

Commands and parameters (“flags”)

`ls -a`	list directory including hidden files
`ls -l`	enhanced file information (date and time, owner and group, permissions)
`ls -g`	show colored output
`ls -d */`	list only directories
`ls -F`	decorate file names according to type

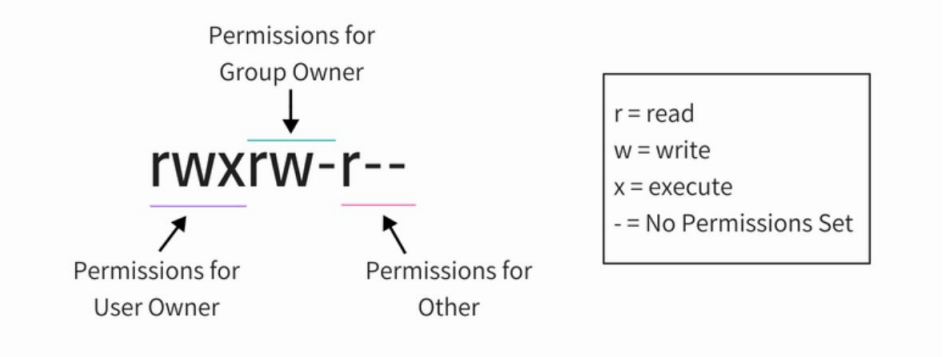
File permissions

```
$ ls -l
```

```
drwxr-xr-x   6 ellibleeker  staff   192B Jul  5 09:01 resources/
```

From left to right:

- File type (d for directory, - for file)
- Permissions (read, write, execute, no permission set)
- Hard link count
- User owner
- Group owner
- File size
- Date / time stamp
- Name



Source:Linux handbook

Modifying the file system

`touch` create a new file

```
$ touch myfile.txt
```

`mkdir` create a new directory

```
$ mkdir myfolder
```

`cp` copy a file

```
$ cp file1.txt  copy-of-file1.txt
```

Modifying the filesystem

`mv` rename/move a file to a new location

```
$ mv old-filename.txt new-filename.txt

$ mv file1.txt new-directory/file1.txt

$ mv file1.txt   new-drectory/new-filename.txt
```

Modifying the file system

Removing files on the CLI should be done with caution!

- There is no “undo remove”
- There is no bin from which you can retrieve your files

➢ Once it’s gone, it’s gone.

Safety measures

- Always check in which directory you are before you remove something
- Have bash ask you for a confirmation

Modifying the file system

`rm` remove a file

```
$ rm copy-of-file1.txt
```

`rm -r` remove directory

```
$ rm -r myfolder
```

`rm -i` check before removing

Useful commands

`man`	gives you the command “manual”
	`$ man rm`
`cat`	dump complete file to the screen
	`$ cat myfile2`
`history`	show history of the last ten commands
`clear`	clears your CLI interface (but scroll up, it’s all still there!)
`!!`	repeat last command

Chaining commands together

`echo` prints some text to the screen (standard output

```
$ echo hello world
$ echo hello        world
```

`echo ~` prints home dir

`echo "<text>"` preserves the text

```
$ echo "~"
```

Chaining commands together

`>` redirect output to a new file

```
$ echo hello world > myfile2.txt
```

`>>` redirects output to an existing file, appending to the end

```
$ echo goodbye you >> myfile2.txt
```

Looking stuff up

- Man pages: `$ man <command>`
- Explain shell. Will dissect any shell command you type in and display help text for each piece. Especially useful if on Windows Git Bash and the `man` page doesn't work
- TLDR pages, to explain `man` pages with practical examples: https://tldr.sh/
- Software carpentry overview of shell commands: https://swcarpentry.github.io/shell-novice/reference.html

3. Exercises

Your turn!

- Navigate (on your CLI) to the `pr-app` folder you created this morning
- Explore the folder with commands like `ls` and `cd` , moving up and down the file directory structure
- Experiment with different flags of `ls`

Your turn!

- Create a new directory in the `pr-app` folder (using the `mkdir` command)
- Navigate into the directory (using the `cd` command)
- Create a new file (using the `touch` command)
- Add some text to the file (with `echo` command and the `>` redirect)
- Check if you succeeded (with `cat`)

Your turn!

- Check the content of the file (with the `cat` command)
- Rename the file (with the `mv` command, inside the same directory)
- Check the result with `ls`
- Move the file (with the `mv` command, to a new directory)
- Change into that directory (with `cd`)
- Remove the file (with the `rm -i` command, be careful!)
- Check the result with `ls`

References

- Explain shell: https://explainshell.com/. Will dissect any shell command you type in and display help text for each piece. Especially useful if on Windows Git Bash and the man page doesn't work;
- TLDR pages, to explain man pages with practical examples: https://tldr.sh/;
- Software carpentry, basic shell commands: https://swcarpentry.github.io/shell-novice/reference.html;
- Course on command line by David J. Birnbaum, Gabi Keane, Emma Schwarz: https://github.com/djbpitt/command-line-fundamentals;
- Matthew Panzarino. 2012. "How Pixar's Toy Story 2 was deleted twice, once by technology and again for its own good", *The Next Web*, May 21, 2012. Online.
- Linux handbook: https://linuxhandbook.com/