UNIVERSITY OF LUCERNE

The ABC of Computational Text Analysis

#4 Introduction to the Command-Line

Alex Flückiger
Faculty of Humanities and Social Sciences
University of Lucerne

14 March 2025

Lectures via Zoom

Date	Topic
22 February 2025	Introduction + Where is the digital revolution?
29 February 2025	Text as Data
07 March 2025	Setting up your Development Environment
14 March 2025	Introduction to the Command-line
21 March 2025	Basic NLP with Command-line
28 March 2025	Introduction to Python in VS Code
04 April 2025	*no lecture (Osterpause)*
11 April 2025	Working with (your own) Data
18 April 2025	Data Analysis of Swiss Media
25 April 2025 (Zoom)	Ethics and the Evolution of NLP
02 May 2025 (Zoom)	NLP with Python
09 May 2025	no lecture (Christi Himmelfahrt)
16 May 2025	NLP with Python II + Working Session
23 May 2025	Mini-Project Presentations + Discussion
30 May 2025	*no lecture (Fronleichnam)*

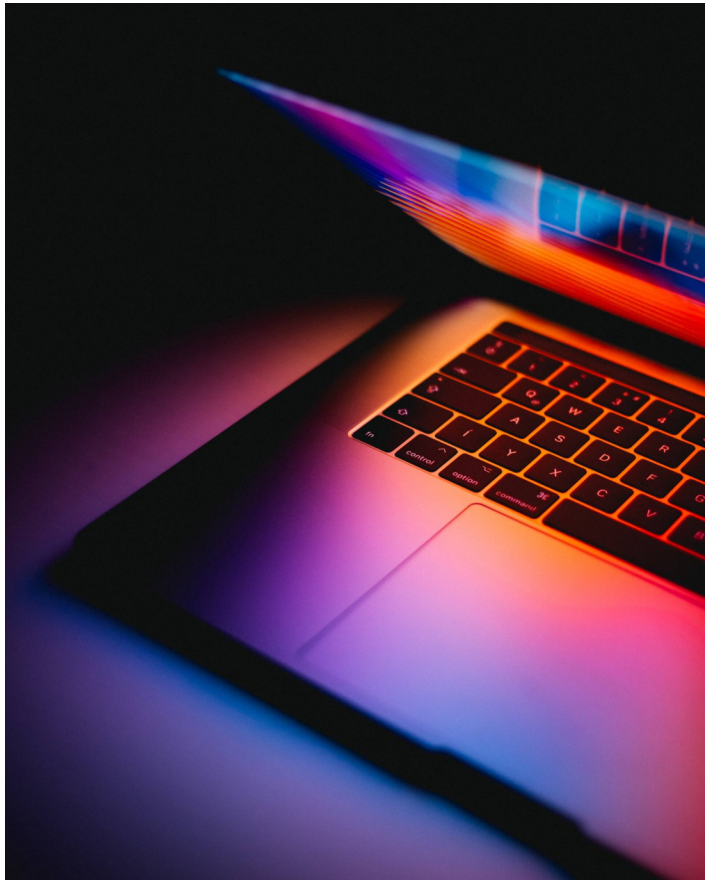
Recap last lecture

- installation successful? ⚙️
- engineering approach 🤓
  - instructions vs clicks, packages, open-source
- any questions ❓

Outline

- learn principles of the shell 🏛️
- perform shell commands ▶️
- get practice by solving exercises 🏗️

What is a computer actually?



Your computer stores files and runs commands

How to get started

Open a Shell

macOS

- open `Terminal`
- shell type: `zsh`

Windows

- open `Ubuntu 22.04 LTS`
- shell type: `Bash`
- ~~open Windows Command Prompt~~

The black window: Run commands

Say hello!

```
echo "hello world"      # print some text
man echo                # get help for any command (e.g., echo)
```

Bourne-again Shell

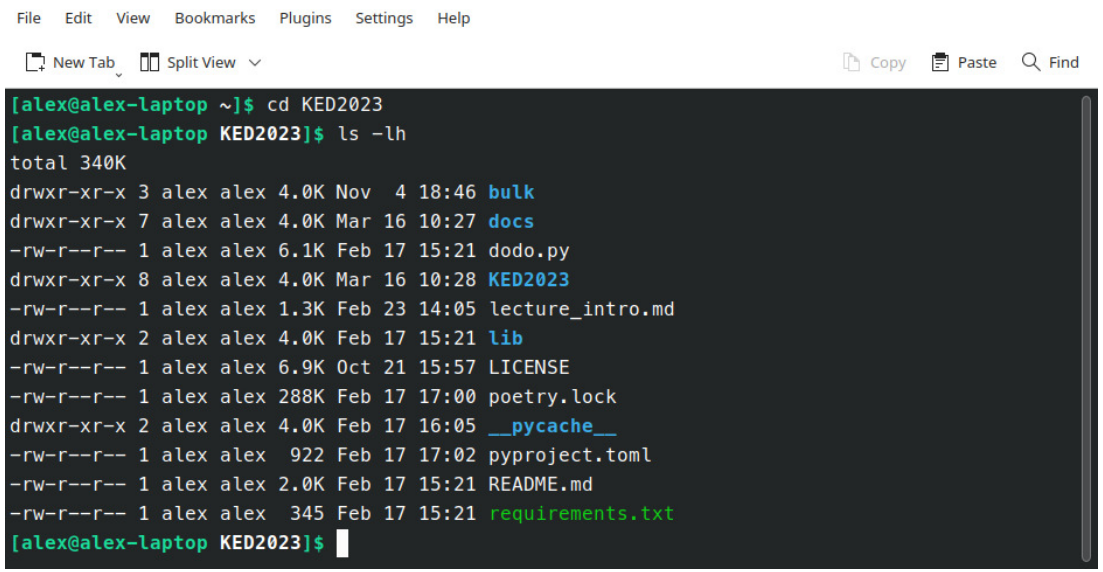
Bash

- offers many built-in tools
- shell prompt

  `USER@HOSTNAME:DIRECTORY$`
- home directory

  `~` refers to `/home/USER`
- case-sensitive 🔤
- no feedback 😶

  unless there is an issue



Unix philosophy

Build small programs that *do one thing*
and *do it well*. 🤓

General structure of commands

Example parts of a command

```
command -a --long_argument FILENAME     # non-working example command
```

Storing files

An analogue equivalent

Cabinet: Old-fashioned and, likely, you have never used one.

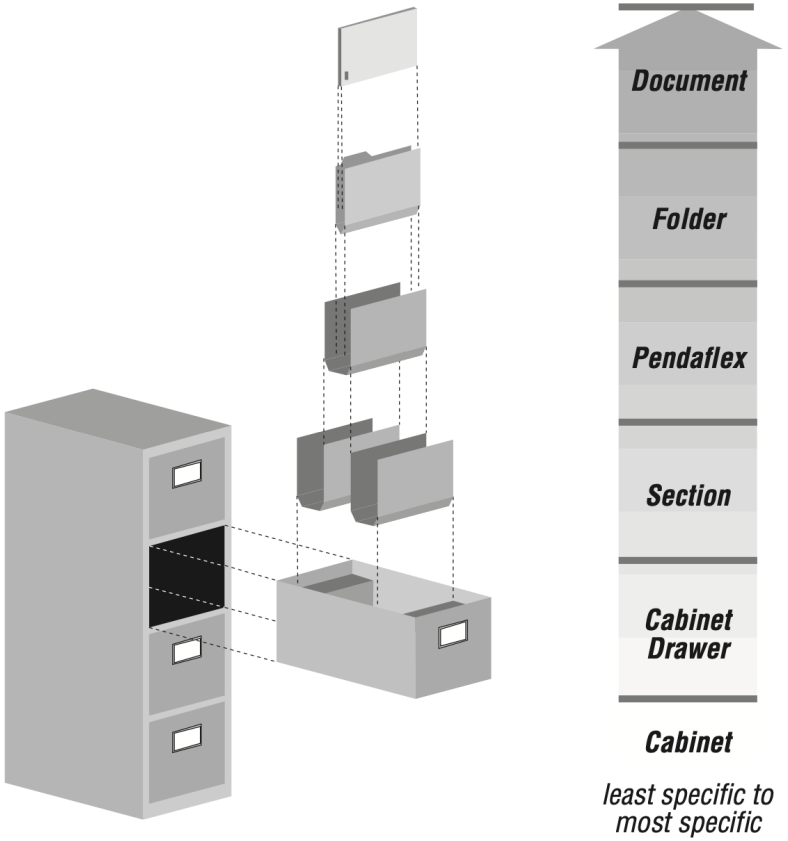


Illustration of a file cabinet (Powers et al. 2002)

Where to find files?

A filesystem is hierarchical contains 🌲

- folders/directories
- files with a suffix (e.g. `.jpg`)

```
.
├── README.md
└── lectures
    ├── images
    │   └── ai.jpg
    ├── html
    │   ├── KED2025_01.html
    │   └── KED2025_02.html
    └── md
        ├── KED2025_01.md
        └── KED2025_02.md
```

How to describe the location of a file?

- **absolute paths** start from top-level directory

  begins with `/` (uppermost folder)

  e.g. `/home/alex/KED2025/slides/KED2025_01.html`

- **relative paths** when looking from current directory

  begins with the name of a folder or file

  e.g. `KED2025/slides/KED2025_01.html`

What is the path?

You are in `/home/myuser/documents` containing the subfolders `pictures` and `texts`.

- What is the **absolute path** to `texts`?
- What is the **relative path** to `texts`?

⚠️ Only relative paths work across systems

Important places in your file system

- shortcut names of directories

  `.` current dir

  `..` parent dir

  `~` home dir (e.g. `/home/alex`)

- find your files on Windows

  `/mnt/c/Users/USERNAME/` (replace with your USERNAME)

  shortcut via `documents`

Navigating in a file system

```
pwd                 # show absolute path of current directory

ls                  # list content of current directory
ls -lh              # list with more information
ls dirname          # list content of directory dirname

cd ..               # change directory to go folder up
cd dir/subdir       # go to folder dir/subdir (two folders down)
```

When you are lost, open the file manager (GUI)

```
open .          # open path in Finder (macOS)
explorer.exe .  # open Explorer in WSL Ubuntu (Windows)
```

Open text files

Show within Shell

```
more text.txt           # print content (spacebar to scroll)

head text.txt           # print first 10 lines of file
tail -n 5 text.txt      # print last 5 lines of file
```

Useful key actions

- autocompletion: `TAB`
- history of used commands: ⬆️
- scrolling: `SPACEBAR`
- cancel: `CTRL + C`
- quit: `q` or `CTRL + D`

Create files and directories

```
touch test.txt          # create a new file

mkdir data              # make a new directory
mkdir -p data/1999      # make a new directory with a subfolder
```

Copy and move files

```
cp test.txt other_folder/      # copy file into other folder
mv test.txt new_name.txt       # rename a file
mv test.txt other_folder/      # move file into other folder
```

Remove files

Watch out, there is no recycle bin. No way back!

```
rm old.txt          # remove a file
rm -r old_data      # remove a folder with all its files
```

In-class: Exercises I

1. Create a new directory called `tmp` in your home directory.
2. Change into that directory using `cd` and print its absolute path using `pwd`.
3. Use `touch` to create a new file called `magic.txt` in `tmp`.
4. Rename the file from `magic.txt` to `easy_as_pie.txt`.
5. Find the `easy_as_pie.txt` file using your graphical file manager (Windows: Explorer, Mac: Finder)
6. Check out the helper page of `mv` command.
7. Look around in the filesystem using `cd` and `ls`. Where are your personal files located?

Follow conventions 🙏

- no spaces/umlauts in names

  only: alphanumeric, underscore, hyphen, dot

- files have a suffix, folders don't

  `text_1.txt` vs. `texts`

- descriptive file names

  `SOURCE/YEAR/speech_party_X.txt`

How is that useful? 🤔

We are getting there!

Wildcards

Placeholders to match ...

- any single character: ?
- any sequence of characters: *

```
mv data/*.txt new_data/.    # move txt-files from to another subfolder
cp *.txt files/.            # copy all txt-files in a single folder
```

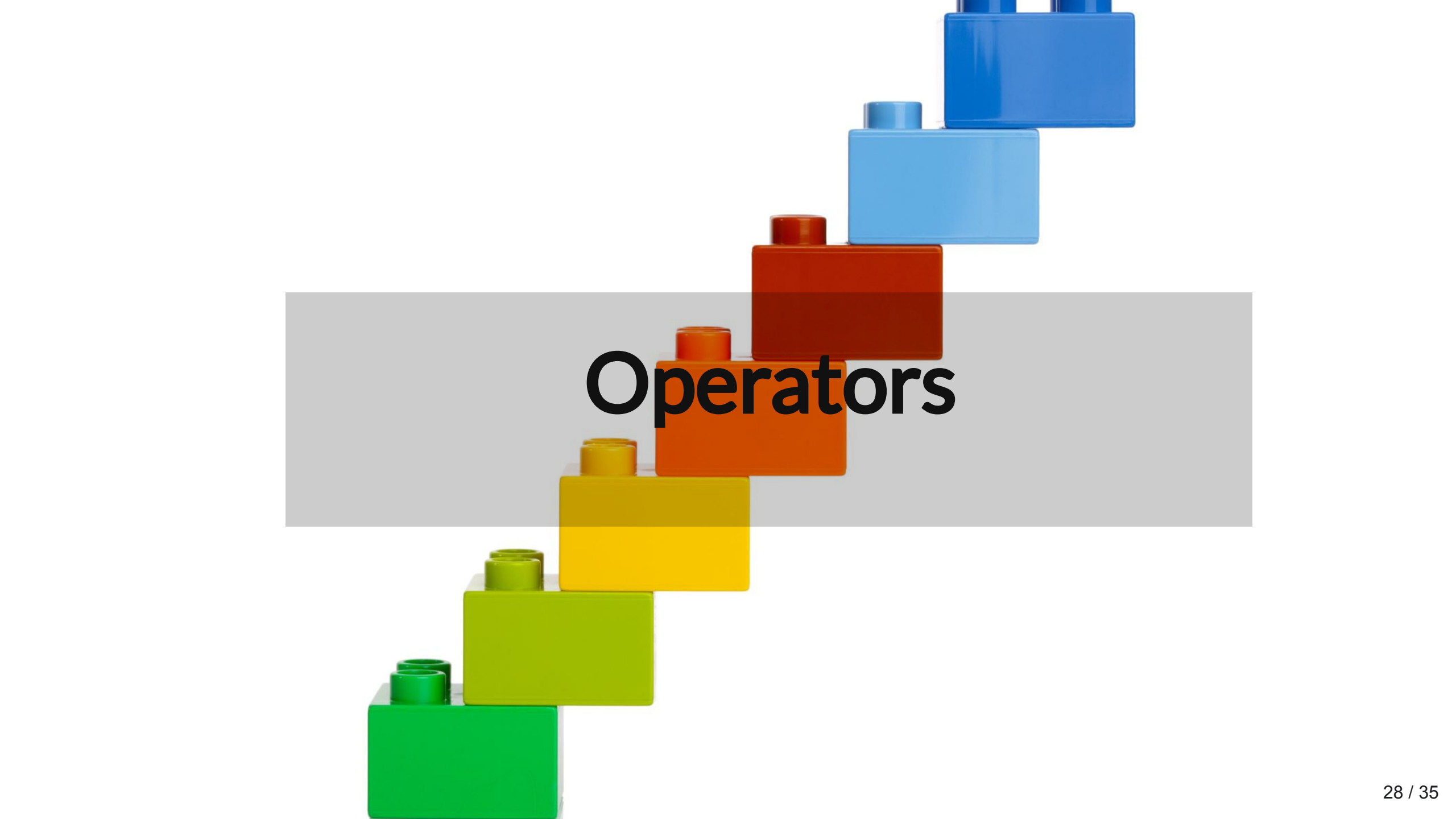
Searching

List certain files only

```
# list all files with the suffix .txt (in current directory)
ls *.txt
```

Find term across files

```
# find all files containing X in provided directory
grep -r "Europe" /path/to/dir
```



Operators

Combining commands

Use shell operators to ...

- redirect output into file (overwrite): `>`
- append to existing file: `>>`
- stream to next command: `|` (pipe)

```
echo 'line 1' > test.txt    # write into file
more test.txt | tail -n 1   # pass output to next command
```

Learn more about operators ⚙️

Merging files

```
cat part_1.txt part_2.txt       # concatenate multiple files
cat *.txt > all_text.txt        # merge all txt into a single one
```


Questions?

In-class: Exercises II

1. Create a new file with `touch`.
2. Write the following content into that file, one line at a time using the append operator:

```
How about making programming a little more accessible? Like:
from human_knowledge import solution
```

3. Make sure that the content was written into that file using `more`.

In-class: Exercises III

1. Navigate up and down in in your filesystem using `cd` and list the respective files per directory with `ls`. Where can you find your personal documents? Print the absolute path with `pwd`.
   Windows users may have a look at `/mnt/c/Users` since they are working on a Ubuntu subsystem.
2. Read `man ls` and write an `ls` command that lists your documents ordered
   - by recency (time)
   - by size
3. Use the `|` and `>` operators to write the 3 “last modified” files in your documents folder into a file called `last-modified.txt` on your desktop (desktop is also a directory). Write a single command performing multiple operations using operators.

Additional resources

Useful intros to Bash

- Cheatsheet for this course
- Introduction to the Bash Command Line by The Programming Historian
- An Introduction to the Linux Terminal by DigitalOcean
- The Unix Shell by Software Carpentry

References

Powers, Shelley, Jerry Peek, Tim O'Reilly, and Mike Loukides. 2002. *Unix Power Tools, Third Edition*. 3rd edition. Sebastopol, CA: O'Reilly Media.