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

16 March 2024

Recap last Lecture

- Successful installation? ✅
- Scripting 🎯

  automate, document, reproduce
- Any questions ❓

Outline

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

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~~

Bourne-again Shell

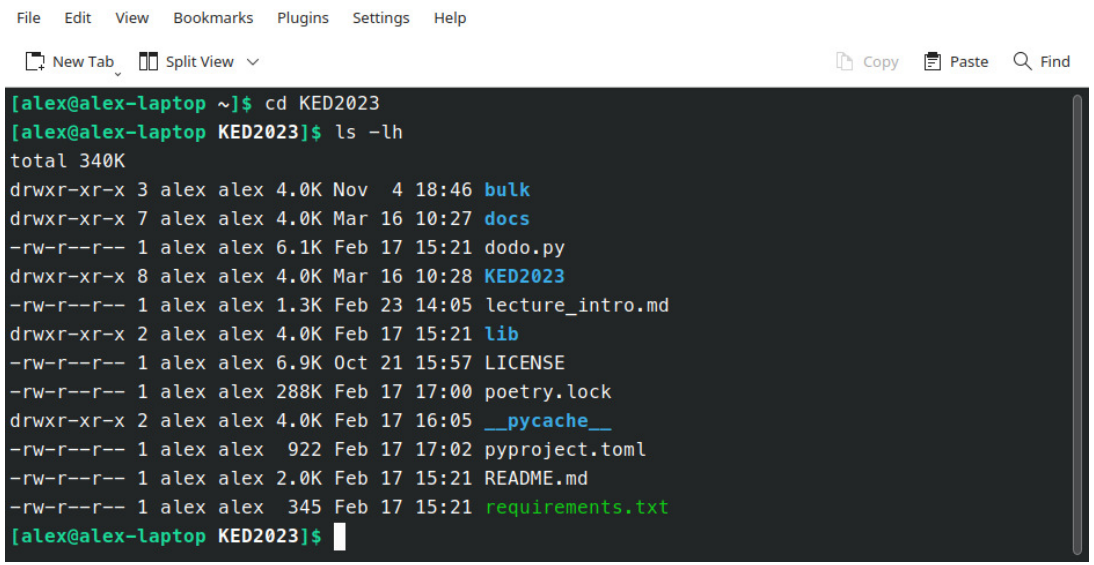
Bash

- offers many built-in tools
- shell prompt

  `USER@HOSTNAME:~$`
- 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*. 🤓

Basic Commands in Shell

Say hello!

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

General Structure of Commands

Example parts of a command

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

Storing Files

… and how to find them

- hierarchical filesystem 🌲

  folders/directories

  files with a suffix (e.g. `.jpg`)

```
.
├── README.md
└── lectures
    ├── images
    │   └── ai.jpg
    ├── html
    │   ├── KED2024_01.html
    │   └── KED2024_02.html
    └── md
        ├── KED2024_01.md
        └── KED2024_02.md
```

Relative vs. Absolute Paths

- absolute paths start from top-level directory

  begins with `/` (uppermost folder)

  e.g. `/home/alex/KED2024/slides/KED2024_01.html`

- relative paths when looking from current directory

  begins with the name of a folder or file

  e.g. `KED2024/slides/KED2024_01.html`

⚠️ Only relative paths work across systems

Important Places in your Filesystem

- 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 Files

Show within Shell

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

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

Creating, Moving and Copying

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/.              # copy file into other folder, keep
mv test.txt other/new_name.txt   # move or rename a file
```

Removing 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`.
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. Check out the helper page of `mv` command.
6. Look around in the filesystem using `cd` and `ls`.
7. Find the created file using your graphical file manager (Windows: Explorer, Mac: Finder)

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 subfol
cp *.txt files/.             # copy all txt-files in a single folder
```

Searching

List certain files only

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

Find specific files

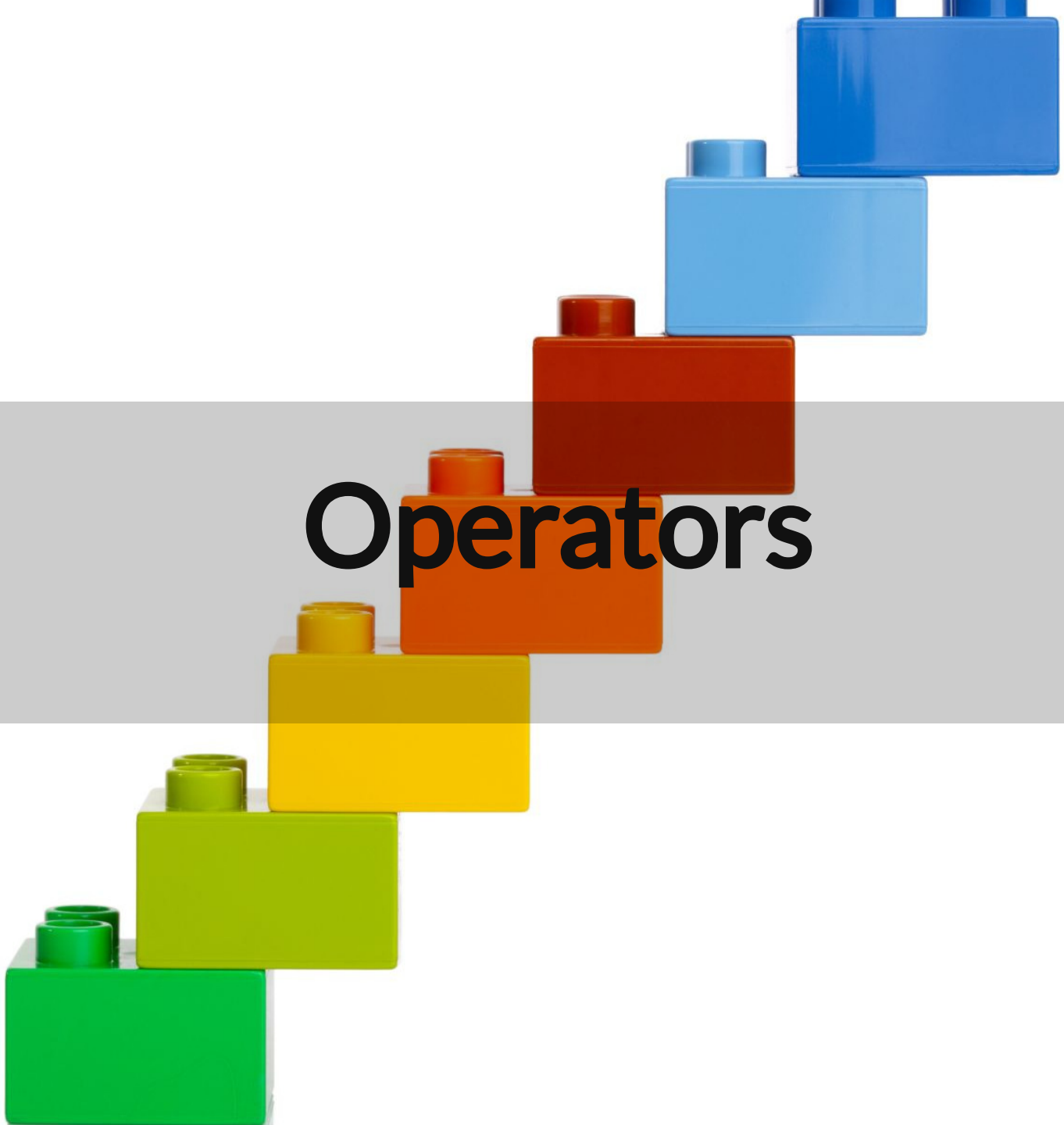
```
# search on content
grep -r "Europe" /path/to/dir    # find all files containing X in a
```

Expansion

Batch processing with expansion

```
touch text_{a..c}.txt
# is equivalent to
touch text_a.txt text_b.txt text_c.txt

mkdir {2000..2005}{a..c}
# is equivalent to
mkdir 2000a 2000b 2000c 2001a 2001b 2001c ...
```



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 -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
```

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`

- don't modify the raw data

Writing a runnable Script

Example script: `TODO.sh`

```
#!/bin/sh

echo "This is my homework."
```

- file with suffix `.sh`

  one command per row

  `#` precedes comments

- start script with Shebang `#!/bin/sh`
- execute with `bash SCRIPTNAME.sh`

The beauty of scripting is automation. ⚡

Assignment #1 ✍️

- get/submit via OLAT
  - starting tonight
  - deadline: 23 March 2024, 23:59
- discuss issues on OLAT forum
- ask friends for support, not solutions


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, one after another.

Additional Resources

Useful primers on Bash

- Cheatsheet for this course
- The Programming Historian
- DigitalOcean