The ABC of computational Text Analysis

04: Introduction to the Command-line

Alex Flückiger
19 March 2020

Action Plan #COVID-19

- interacting through chat + (audio + video)

  mute mic unless you speak

- slides already online
- weekly live-lectures
- recorded sessions on SWITCHtube

Let's resist! 😷

Recap last Lecture

- Was the installation successful?
- Any questions?

Outline

- learn principles of the shell
- perform shell commands
- solving exercises

Unix Philosophy

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

Starting a Shell

macOS

- open `Terminal`
- shell type: zsh

Windows

- open `Ubuntu 18.04 LTS`
- shell type: Bash
- ~~open Windows Command Prompt~~

Bourne-again Shell

- offers many built-in apps
- shell prompt

  `USER@HOSTNAME:~$`
- home directory `~`

  refers to `/home/USER`
- case-sensitive
- no feedback

Getting started in a Shell

running commands

```
command -a --long_argument FILE      # generic components
echo "hello world"                   # print a hello
```

getting help

```
man echo        # get help for any command (e.g.; echo)
echo --help     # get help for any command (e.g.; echo)
```

Structure of a File System

- hierarchical file system

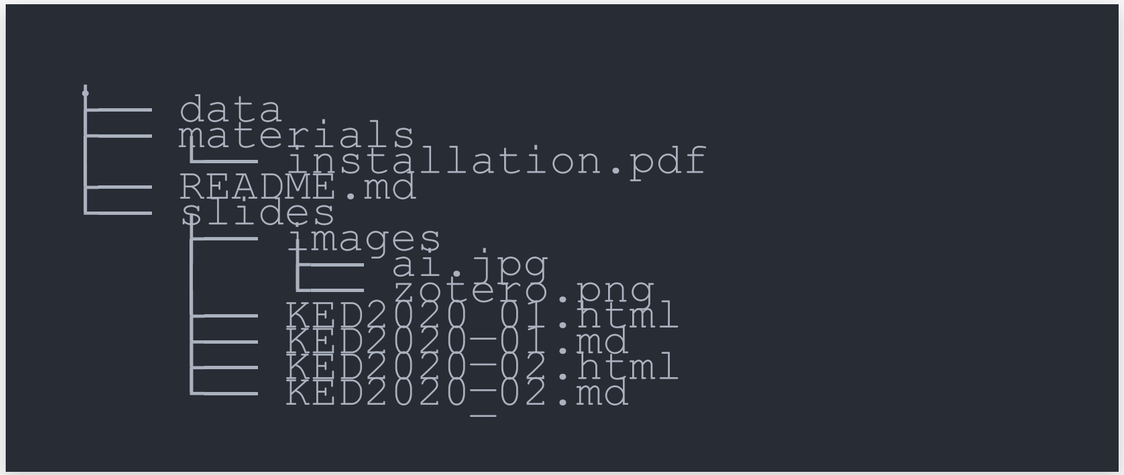
  tree-like

- relative vs. absolute path

  relative works across systems
  data/slides

- common directories

  `.` *current*
  `..` *parent*
  `~` *home directory*



Navigating in a File System

list content

```
pwd                     # show absolute path to 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)
```

open in a window

```
open .      # open path in finder (macOS)
nautilus .  # same for Ubuntu (Windows)
```

Reading/Modifying Files

```
more text.txt                # print content (space to scroll)
head text.txt                # print first 10 lines of file
tail -5 text.txt             # print last 5 lines of file
# Ubuntu Linux (Windows)
xdg-open text.txt            # open file in default application
# macOS
open text.txt                # open file in default application
```

Useful Key Actions

- autocompletion: *tab*
- get last command: ↑
- scrolling: *space*
- interrupt *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
```

change their location

```
cp test.txt /other/.                    # copy file, keep its name
mv test.txt /other/new_name.txt         # move or rename a file
```

Removing Files

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

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

Wildcards

placeholders to match ...

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

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

Searching

collect certain files only

```
ls *.txt        # list all files with the suffix .txt
ls */*.txt      # list all txt-files in any subfolder
```

find specific files

```
# concerning names
find /path/to/dir -name "fname"   # find a file in specific directory
locate -i pattern_1 pattern_2     # global search of files/folders
# concerning content
grep -r 'x'                       # find files containing x
```

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 ...
```

Combining Commands

shell operators to ...

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

```
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
```

Course Repository

background

- Git version control software
- GitHub hosting platform

```
# get an initial copy of the course material
git clone https://github.com/aflueckiger/KED2020.git
# update your local copy continiously
cd KED2020
git pull
```

Scripting

- all commands in single script

  one command per row

- start script with Shebang

  `#!/bin/sh`

- execute with

  `bash scriptname`

```
#!/bin/sh
# example script located at: scripts/find_all_pdf.sh
echo "This is a list of all PDFs on my computer:"
locate -i /home/*.pdf
```

Conventions

- no spaces/umlauts in names

  alphanumeric, underscore, hyphen, dot

- files have a suffix, folders not

  `text_1.txt` *vs.* `texts`

- descriptive file organization

  `SOURCE/YEAR/speech_party_X.txt`

- separate data from scripts

- never change the raw data

Questions

Assignment #1

- get/submit via OLAT

  starting tonight

  deadline: 26 March 2020, 23:59

- ask friends for support, not solutions

In-class: Exercises I

1. If you have not cloned the course repository from Github, do this now.
2. Create a new directory called `tmp` in the course directory `KED2020`.
3. Look up the `touch` program. The `man` program is your friend.
4. Use `touch` to create a new file called `advice_for_programmers.txt` in `tmp`.
5. Write the following content into that file, one line at a time using operators:

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

In-class: Exercises II

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`.
   Hinting Windows users as they are working in a Ubuntu subsystem, check out: `/mnt/c/Users`
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). It is a single command performing multiple operations, one after another.

Additional Resources

useful primers on Bash

- The Programming Historian
- DigitalOcean