Cheatsheet Shell Commands

Seminar KED2023

Alex Flückiger

30 March 2023

Contents

1 Basic Shell Commands **1**

2 Regular Expressions **2**

1 Basic Shell Commands

Shell Command	**Explanation**
`cd` *filepath*	**c**hange **d**irectory aka move into a different folder
`ls -lh` *folder*	**lis**t the files and folders in your current **dir**ectory
`pwd`	show **p**ath of **w**orking **d**irectory aka the folder that you're in right now
`touch` *fname*	make a new file
`mkdir` *dirname*	**mak**e a new **dir**ectory aka a folder
`rm` *fname*	**rem**ove aka delete a file or directory
`cp` *original-fname copied-fname*	**cop**y a file or directory
`mv` *original-fname new-fname*	**mov**e or rename a file or directory
`cat` *fname*	show all the contents of a file
`more` *fname*	show snippet of a file that allows you to scroll through the entire thing
`head` *fname*	show the first 10 lines of a file (change number of lines by adding a flag, e.g. `head -100`)
`tail` *fname*	show the last 10 lines of a file (change number of lines by adding a flag, e.g. `tail -100`)
`wc -w -l` *fname*	show how many **w**ords or lines in a file
`man` *command*	show the **man**ual aka the documentation that tells you what a particular command does
`echo`	print text to the command line
`egrep "search pattern"` *fname* or *dirname*	search for lines that include search term in file. See below for the arguments of egrep.
`wget` *url*	**get** a file from the **w**eb

This cheatsheet is based on this resource. Please also refer to this resource for a more in-dept explanation in prose. You should follow the guide for macOS and Unix even as a Windows user as we have installed a Unix environment.

1.1 Searching with egrep

`egrep` allows pattern-based search (i.e., searching with regular expressions). The most common arguments of `egrep` are:

- `-i` search case insensitive
- `-r` search recursively in folder
- `-o` show exact matches only instead of entire lines with matches
- `-h` suppress the file path where the match occurred

1.2 Operators

- `|`: A pipe takes the output of one command and passes it as the input to another.

  ```
  echo "pass this text to next command" | cat
  ```

- `>`: This operator redirects the output to a file (overwrites if it already exists). Example:

  ```
  echo "first line of file1" > file1
  ```

- `>>`: This operator redirects and appends the output to an *existing* file: Example:

  ```
  echo "line following existing content of file1" >> file1
  ```

2 Regular Expressions

2.1 Counting words across Files

It is common to quantify words across files. The example command

- searches for a word starting with `eco` and continuing with any letters
- count the number of occurrences
- sorts the words according to their frequency.

```
egrep -roh "\beco[a-z]*" **/*.txt | sort | uniq -c | sort -h
```

`\b` matches the boundary of a word.

2.2 Example Patterns

```
# alle Kleinbuchstaben
echo "Das ist ein Satz mit der Zahl 1000" | egrep --colour "[a-z]"

# alle Grossbuchstaben
echo "Das ist ein Satz mit der Zahl 1000" | egrep --colour "[A-Z]"

# das Wort "ist" und das nächste Wort
echo "Das ist ein Satz mit der Zahl 1000" | egrep --colour "ist [a-z]*"

# das Wort "Zahl" gefolgt von einer Ziffer
echo "Das ist ein Satz mit der Zahl 1000" | egrep --colour "Zahl [0-9]"

# das Wort "Zahl" gefolgt von beliebig vielen Ziffern
echo "Das ist ein Satz mit der Zahl 1000" | egrep --colour "Zahl [0-9]*"
```

2.3 Pattern Equivalence

```
a+ == aa*                 # "a" once or more than once
a? == (a|_)               # "a" once or nothing
```

```
a{3} == aaa               # three "a"
a{2,3} == (aa|aaa)        # two or three "a"
[ab] == (a|b)             # "a" or "b"
[0-9] == (0|1|2|3|4|5|6|7|8|9)  #any digit
```