

**The 19th International Conference on Web Engineering (ICWE-2019)**

**June 11 - 14, 2019 - Daejeon, Korea**

# **Powerful Unix-Tools - grep**

**Andreas Schmidt**

**Department of Informatics and  
Business Information Systems  
University of Applied Sciences Karlsruhe  
Germany**

**Institute for Automation and Applied Informatics  
Karlsruhe Institute of Technologie  
Germany**

## grep (1)

- Purpose:

**grep searches the named input FILEs for lines containing a match to the given PATTERN.** If no files are specified, or if the file "-" is given, grep searches standard input. **By default, grep prints the matching lines.**

- Advantages/Disadvantages:

- + Easy to use from command line
- + Small footprint
- + Supports regular expressions
- - Slow (~2 GB/sec.)

## grep (2) - useful options

- -E, -P option: Supports extended regular expressions
- -i: case insensitive match
- -o: only prints matching word/pattern
- -w: print line/pattern only, if match forms a whole word
- -m <num>: stop reading file, after <num> matches
- -v: output lines, which does not contain pattern
- -c: Count the number of matches for each file
- -l: print name of file, instead of matching lines/matches

## grep (3) - useful options

- -L: print files without match
- -n: add line number for each match
- -b: add byte offset for each match
- -h option: suppress output of filename (if multiple files are given)
- -A<num>, -B<num>, -C<num>, -<num>: print <num> context lines around match
- --color=auto: print match in red
- --label=<file-name>: Display input coming from STDIN, as input coming from file <file-name>.
- -a: Also handle binary files as text-files

## Simple grep Examples

- Print lines containing the term<sup>1</sup> „Column Store“ (or **colUmn sTOre**, ...). Each output-line is preceded with the filename, in which the pattern was found

```
$ grep 'Column Store' papers/*.txt
```

```
$ grep -i 'column store' papers/*.txt
```

- Print the **names of all documents**, containing lines with term 'column store'

```
$ grep -i -l 'column store' papers/*.txt
```

- Print for each document, **the number of times, the word 'didactic' appears**

```
$ grep -i -l -c 'didactic' papers/*.txt
```

---

1. in one line

## grep Examples with Regular Expressions

- Print all IP-adresses in the document office-net.txt in red)  
`grep --color=auto -E '([0-9]{1,3}\.){3}[0-9]{1,3}' office-net.txt`
- Print all lines, **not** starting with an '#' character  
`grep -E -v '^ *#' ../src/build-index.php`
- Print all files, not containing one of the two words (teaching, didactic)  
`grep -L -E '(didactic|teaching)' papers/*.txt`

## Further grep Examples

- Match all 4 digit numbers, if they are preceded by „in“ and followed by a dot (.) (lookbehind/lookahead)

```
grep -P -o '(?<=in ) [0-9]{4} (?=\.)' papers/*.txt --color=auto
```

- Print not only matching line, but also a context of 3 lines before/after

```
grep -3 'Emanuel Macron' news/*.txt
```

## Output of Filename

- Rule for output of filenames:
  - If multiple files are given (or wildcards used), each match is preceded with the filename
  - if only a single file is given (or input from STDIN), no filename is printed out (see also --label, -H options)

```
$ grep -a -n function Bruch.php
8:  function __construct($z=0, $n=1) {
16: private function kuerze() {
23: function add($b) {
30: function neg() {
34: function __toString() {
41: static function ggt($m, $n) {
```

```
$ grep -a -n function B*.php
Bruch.php:8:  function __construct($z=0, $n=1) {
Bruch.php:16: private function kuerze() {
Bruch.php:23: function add($b) {
Bruch.php:30: function neg() {
Bruch.php:34: function __toString() {
Bruch.php:41: static function ggt($m, $n) {
```