

Betriebssysteme und Softwaretechnik

Sample Sheet
Deadline: 04.05.2017

Max Mustermann Matr.Nr. 1701
Autor Zwei Matr.Nr. 4711
Dritter Kollaborateur Matr.Nr. 4242

# 1	# 2	Σ

`-n, --line-number`
Prefix each line of output with the
↪ 1-based line number
within its input file.

Aus `man fortune`:

HISTORY
This version of fortune is based on the
↪ NetBSD fortune 1.4,
but with a number of bug fixes and
↪ enhancements.

Problem 1

Problem 3

Problem 2

Problem 2.1

a)

```
$ uname -a
Linux BuS1337 4.9.16-gentoo #1 SMP Fri Apr 28
↪ 16:18:12
CEST 2017 x86_64 Intel(R) Core(TM) i7-6600U
↪ CPU @ 2.60GHz
GenuineIntel GNU/Linux
```

```
$ cat /etc/issue

This is \n.\O (\s \m \r) \t
```

```
$ gcc --version
gcc (Gentoo Hardened 5.4.0-r3 p1.3, pie-0.6.5)
↪ 5.4.0
Copyright (C) 2015 Free Software Foundation,
↪ Inc.
This is free software; see the source for
↪ copying
conditions. There is NO warranty; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR
↪ PURPOSE.
```

```
$ w
21:53:q25 up 3:36, 4 users, load average:
↪ 0.00, 0.02, 0.09
USER      TTY      LOGIN@  IDLE   JCPU   USER      TTY      LOGIN@  IDLE   JCPU
↪ PCPU WHAT
root      tty2          18:45   11:57   0.31s  0.31
↪ s -bash
root      tty1          18:25   2:31m  0.60s  0.33
↪ s links
georg     pts/0        19:29   1:59m  0.03s  0.03
↪ s -bash
georg     pts/1        20:01   1.00s  0.07s  0.00
↪ s w
```

b)

Aus `man man`:

```
man -k printf
Search the short descriptions and
↪ manual page names for the
keyword printf as regular expression
↪ . Print out any
matches. Equivalent to apropos printf.
```

Aus `man grep`:

Problem 2.2

a)

```
echo "Bus_2016:_Abgabe_der_1._Uebung_am_6.5."
↪ | sed s/6/7/
Bus 2017: Abgabe der 1. Uebung am 6.5.
```

b)

Der Befehl `cut -d ' ' -f 1 d*` gibt jeweils aus allen Dateien im aktuellen Ordner, die mit `d` beginnen den Beginn jeder Zeile bis zum ersten Leerzeichen aus. `-d ' '` setzt das Trennzeichen auf ein Leerzeichen. `-f 1` legt fest, dass nur das erste Feld was mit diesem Trennzeichen gefunden wurde ausgegeben wird. Durch `d*` wird der Befehl auf alle Dateien angewendet, deren Namen mit `d` beginnt.

c)

```
$ grep -B 19 -ne '[0-9]\{5\} [a-zA-Z]\{1,\} [
↪ a-zA-Z]\{1,\}' emails
19017-Message-ID: <32509bcc5b1a43c@posteo.de>
19018-Date: Tue, 26 Oct 2010 15:26:51 +0200
19019-From: Arthur Dent <realArthurDent@posteo.
↪ de>
...
19022-To: Emily Saunders <emily.
↪ saunders@mostlyharmless.com>
...
19034-Arthur Dent
19035-Galaxy 7
19036:74369 Third Orbit
```

Die durch “...” markierten Teile der Ausgabe wurde aufgrund von fehlender Relevanz für die Aufgabenstellung ausgelassen

Problem 2.3

a)

Der erste aufruf von `tr` im Befehl

```
$ tr -d '?!,:;+&'"' < wotw.txt | tr -s " "
```

entfernt alle der spezifizierten Sonderzeichen aus der eingabe, welche aus der Datei `wotw.txt` gepiped wird.

Die Ausgabe des Befehles wird in den zweiten **tr** aufruf gepiped, in welchen (durch das flag **-s**) jedes doppelte Leerzeichen entfernt wird.

b)

Durch den Befehl

```
$ grep "road" -i -v -c wotw.txt
6330
```

erfährt man, dass exakt 6330 Zeilen des Dokumentes das Wort "road" nicht enthält. Das flag **-i** sorgt dafür, dass **grep** die Groß-/Kleinschreibung ignoriert, **-v** sorgt dafür, dass nur Zeilen ohne Match ausgegeben werden, **-c** Zählt die von **grep** ausgegebenen Zeilen.

c)

Der **grep** Befehl wandelt zunächst die Datei in eine Liste ihrer Wörter um. Danach werden die Wörter mit **sort** gruppiert damit sie mit **uniq -c** zusammengefasst werden können und die Anzahl ihrer Vorkommnisse bestimmt werden kann. Um die häufigsten 10 zu ermitteln wird dann mit **sort -n -r** nach den von **uniq** hinzugefügten Anzahlen absteigend sortiert und schlussendlich werden mit **head -n 10** nur die ersten 10 ausgegeben.

```
grep -o '\<[[:alpha:]]*\>' wotw.txt | sort |
↪ uniq -c | sort -n -r | head -n 10
4417 the
2373 and
2284 of
1554 a
1300 I
1160 to
924 in
853 was
754 that
568 had
```

Problem 2.4

a)

```
$ git init
```

b)

```
$ git add A.txt B.txt C.txt
$ git commit -m "Initial"
```

c)

```
$ git diff
```

Gibt die alle Änderungen seit dem letzten commit aus.

```
$ git diff $FILE
```

Vergleicht die Datei **\$FILE** mit der Version aus dem letzten Commit, und gibt alle Unterschiede aus.

d)

```
$ git commit -m "Change_files"      # commit
$ git log                          # Anzeigen
↪ der Commits
```

Problem 2.5

a)

```
$ convert B.png C.jpg BC.pdf && pdftunit A.pdf
↪ BC.pdf ABC.pdf
```

b)

```
$ pdftk A.pdf cat 5-9 23 240-242 output
↪ Relevant.pdf
```