

## Übung 1

---

### Aufgabe 1.1 **Korrekt**

a)

```
echo "BuS 2016: Abgabe der 1. Uebung am 6.5" | sed -e s/6/7/
```

b)

Der Befehl gibt von jeder Zeile der Datei  $d^*$  das erste Wort, also alles was vor dem ersten Leerzeichen jeder Zeile steht, aus.  $-d$  ' ' beschreibt dabei, welches Zeichen als Trennzeichen verwendet wird, der Text wird also als eine Tabelle aufgefasst wobei jedes Leerzeichen in einer Zeile für eine weitere Spalte steht. Nun beschreibt  $-f 1$ , dass die erste Spalte ausgegeben werden soll, also das erste Wort. Das  $d^*$  am Ende beschreibt die Datei, aus welcher die Information gelesen werden soll.

c)

Der Befehl :

```
grep -A3 -E ^[[[:alpha:]]\+[[[:space:]]\+[[[:alpha:]]\+ $ emails  
| grep -B1 -A1 -E ^[[[:alpha:]]\+[[[:space:]]\+[[[:digit:]]\+ $
```

Findet die Adresse:

Arthur Dent  
Galaxy 7  
74369 Third Orbit

Welche sich in den Zeilen 10034 bis 19036 befindet. Die E-Mail wurde von realArthurDent@posteo.de verschickt und von emily.saunders@mostlyharmless.com erhalten.

## Aufgabe 1.2 0,5/2,5

a)

```
tr -s " " < wotwNeu.txt | tr -s " " > wotwNeu.txt
```

Der Befehl speichert das angepasste Dokument als wotwNeu.txt

falsch -0,5

b)

```
cat wotw . txt | tr -d "'\"?\\.\!\\:; ,+& ' | tr -s ' ' > wotw_nospecial . txt
```

```
tr ' ' '\n' < wotw.txt | grep road | wc -l
```

Das Ergebnis ist 122. Falsch -1

c)

```
tr -c '[:alnum:]' '\n*' < wotw.txt | sort | uniq -c | sort -nr | head -10
```

Das Ergebnis ist 12223 absolute Häufigkeit, 4417 the, 2373 and, 2284 of, 1554 a, 1300 I, 1160 to, 924 in, 853 was und 754 that. Wortreihenfolge richtig, aber falsche Werte - 0,5

## Aufgabe 1.3 3/5,5

a)

Ein Systemcall bzw. Syscall ist eine Methode um vom Betriebssystem bereitgestellte Funktionalitäten auszuführen, wie zum Beispiel das Schreiben einer Datei. Dabei wird die Kontrolle vom Programm an den Kernel übergeben.

b)

*execve* = Ausführen einer ausführbaren Datei (z.B. .out Datei) oder interpretieren der Datei durch in der Datei definierten Interpreter.

*open* = Aufruf einer Datei (File Management )

*stat* = Status einer Datei oder eines Dateisystems anzeigen (Speicherverbrauch, Zugriffszeitpunkt... ).

*mmap* = Bildet eine Datei oder ein Gerät im virtuellen Speicher ab. (ka ob das so stimmt...)  
= mappt eine Datei in den Prozessadressraum -0,5

c)

*strace* = diagnostic tool welchen den Prozess mit dem es gestartet wurde überwacht und alle syscalls die es aufruft speichert. zu rudimentär -0,5

d)

Es gibt zwischen `ls` und `ls -l` bzgl. der Systemcall nur Unterschiede in den Speicheradressen und deren Werte. Beide Kommandos nutzen demnach gleich viele Systemcalls. Bei der Ausgabe gibt `ls -l` im Vergleich zu `ls` neben den Datei und Ordernamen auch noch Besitzer, Rechte usw. an.

**Keine Analyse der `strace`-Ausgaben. 0,5/2**

#### Aufgabe 1.4

**1,5/3,5**

a)

`str[1] = '0';`  $\implies$  setzt den zweiten Eintrag des char Arrays `str` auf 0. `str` hat nun die Werte 103.

b)

`i = *(list + 3);`  $\implies$  setzt den Wert von `list[3]` auf 7

c)

`pi = &list[i];`  $\implies$  Durch den Befehl wird versucht den Zeiger `pi` auf den int Wert von `list[7]` zu zeigen. Da `list` aber nur 4 Elemente hat ist dieser Befehl nicht möglich. **Doch ist möglich; zeigt dann auf n.-def. Speicherbereich -0,5**

d)

`*pi = 42;`  $\implies$  Auf den Speicherbereich auf den `pi` zeigt soll der int Wert 42 geschrieben werden. Da `pi` aber noch nicht initialisiert ist, könnte der Compiler einen Fehler werfen.

**Begründung falsch. N. wohldef. Speicherbereich kann zu OS gehören... -0,5**

e)

`list = pi;`  $\implies$  Durch diesen Befehl soll die Basisadresse von `list` auf die Adresse auf die der Zeiger von `pi` zeigt gelegt werden, da das Array nun aber keinen zusammenhängenden Speicherbereich mehr hat ist das ein Fehler.

f)

`i = *(pi + 2);`  $\implies$  Da `pi` immer noch nicht initialisiert ist, ist es unklar worauf `pi` zeigt. Demnach ist der Zuweisung auf den 2. Speicherbereich nach `pi` nicht möglich. **Falsch**

g)

`str[3]='4';`  $\implies$  Das char-Array `str` wird an der Stelle mit dem Index 3 um das Zeichen 4 erweitert. `str` hat jetzt den Wert 1034. **\o-Terminierung wird überschrieben. Hätte erwähnt werden sollen**

**-0,5**

## Aufgabe 1.5 Korrekt

a)

Das Hauptproblem des Quelltextes ist ein Rundungsproblem zwischen float zu int.

**Seid was präziser!**

b)

---

```
1 #include <stdio.h>
2
3 float celsius_zu_fahrenheit(float temperatur) {
4     return (float) 9/(float) 5 * temperatur + (float) 32;
5 }
6
7 int main()
8 {
9     float celsius, fahrenheit; /* Gleitkommavariablen */
10
11     printf("Temperatur in Celsius: ");
12     scanf("%f", &celsius);
13
14     fahrenheit = celsius_zu_fahrenheit(celsius);
15
16     printf("Temperatur in Fahrenheit: %f\n",fahrenheit);
17 }
```

---

**Code IMMER als separate .c-Datei abgeben!**

c)

Es ist möglich das Programm ohne explizite Typkonvertierung zu lösen indem man 9/5 durch 1.8 ersetzt.

## Aufgabe 1.6 Korrekt

3. b zeigt vor dem Aufruf auf a[0]. Durch b++; zeigt b nun auf a[1].
4. t1 wird der Wert zugewiesen der in der Speicheradresse von b steht. Also wir t1 = 11. (Dereferenzierung)
5. Der Pointervariable c wird die Speicheradresse von b zugewiesen. (Referenzierung)
6. c erhält den Wert der in der Speicheradresse steht und erhöht ihm um zwei. Nun hat das Array folgende Einträge a[] = 2, 13, 23, 42, 13, 37. (2x Dereferenzierung)
7. c erhält den wert 13, demnach erhält t2 den Wert 14. (2x Dereferenzierung)
8. \*\*c zeigt auf a[1]=13 und a[3]=42, demnach ist t3 = 42&13. Der & operator ist in diesem Fall ein bitweises UND. Somit ist t3 = 8.
9. b zeigt nun auf a[5], weil b = a+5  $\iff$  b = &a[5].
10. Da b auf a[5] zeigt wird a[5] auf 10 gesetzt. Nun hat das Array folgende Einträge a[] = 2, 13, 23, 42, 13, 10.