Gruppe **20**
Betriebssysteme und Systemsoftware

Patrick Arens, 377536
Georg Manthey 376854
Benedikt Gerlach, 376944

Übung 1

Aufgabe 1.1

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

a)

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

Der Befehl speichert das angepasste Dokument als wotwNeu.txt

b)

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

Das Ergebnis ist 122.

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.

Aufgabe 1.3

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, Zugriffzeitpunkt...).

mmap = Bildet eine Datei oder ein Gerät im virtuellen Speicher ab. (ka ob das so stimmt...)

c)

strace = diagnostic tool welchen den Prozess mit dem es gestartet wurde überwacht und alle syscalls die es aufruft speichert.

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.

Aufgabe 1.4

a)

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

b)

*i = *(list + 3);* $\Longrightarrow$ setzt den wert von list[3] auf 7

c)

pi = &list[i]; $\Longrightarrow$ 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.

d)

**pi = 42;* $\Longrightarrow$ 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.

e)

list = pi; $\Longrightarrow$ 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);* $\Longrightarrow$ 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.

g)

str[3]='4'; $\Longrightarrow$ Das char-Array str wird an der Stelle mit dem Index 3 um das Zeichen 4 erweitert. str hat jetzt den Wert 1034.

Aufgabe 1.5

a)

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

b)

```c
#include <stdio.h>

float celsius_zu_fahrenheit(float temperatur) {
    return (float) 9/(float) 5 * temperatur + (float) 32;
}

int main()
{
  float celsius, fahrenheit; /* Gleitkommavariablen */

  printf("Temperatur in Celsius: ");
  scanf("%f", &celsius);

  fahrenheit = celsius_zu_fahrenheit(celsius);

  printf("Temperatur in Fahrenheit: %f\n",fahrenheit);
}
```

c)

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

Aufgabe 1.6

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.