Übungen zu Computergrundlagen WS 2017/2018

Übungsblatt 3: Unixgrundlagen 3

3. November 2017

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 10.11.2017, 11:00 Uhr**
- Schickt die Lösungen bitte per Email an Euren Tutor:
  - Montag 11:30 – 13:00: Julian Zeller (`julian.zeller@icp.uni-stuttgart.de`)
  - Montag 14:00 – 15:30: Miriam Kohagen (`mkohagen@icp.uni-stuttgart.de`)
  - Dienstag 14:00 – 15:30: Ingo Tischler (`itischler@icp.uni-stuttgart.de`)
  - Dienstag 15:45 – 17:15: Konrad Breitsprecher (`konrad@icp.uni-stuttgart.de`)
  - Donnerstag 09:45 – 11:15: Ashreya Jayaram (`ashreyaj@icp.uni-stuttgart.de`)
- Die Übungen sollen von Gruppen von jeweils *zwei bis drei* Leuten bearbeitet werden. Bitte gebt *nur eine Lösung pro Gruppe* ab und nennt in eurer Abgabe alle Mitglieder eurer Gruppe!
- Wie in den vorherigen Übungsblättern sollen die Lösungen der Aufgaben in eine Textdatei eingetragen werden, welche ihr dann per E-Mail an euren Tutor schickt.

Aufgabe 3.1: Dateirechte (5 Punkte)

In einem Terminal hat der Benutzer `zeman` folgenden Dialog:

```
$ groups zeman kai cgl17-001
zeman : icp klausur cgl www-data cpp pc
kai : icp asm sysguru www-data klausur stud
cgl17-001 : cgl www-data stud

$ ls -la
total 8
drwxr-xrwx 5 zeman cgl  98 Oct 30 19:22 .
drwxr-xr-x 3 zeman icp 302 Oct 30 19:14 ..
-rw-r----- 1 zeman cgl 142 Oct 30 19:21 bar.txt
dr-xrwxr-x 2 zeman cgl  21 Oct 30 19:13 cglstuff
----rw---- 1 zeman cgl 142 Oct 30 19:22 foo.txt
drwxr--rwx 2 zeman icp   6 Oct 30 19:22 private
drwx------ 2 zeman icp   6 Oct 30 19:22 public
----r-x--- 1 zeman cgl   0 Oct 30 19:19 script.sh
```

- **3.1.1** (3 Punkte) Welche der Benutzer `zeman`, `kai` und `cgl17-001` können welchen der folgenden Befehle erfolgreich ausführen? Gebe für jeden Befehl an, welche der Benutzer ihn ausführen können und welche nicht. *Begründe* deine Antworten!
  - `cat foo.txt` (Lesen von `foo.txt`)
  - `cp bar.txt cglstuff/`
  - `./script.sh` (Ausführen von `script.sh`)

Hinweis: Bei manchen Befehlen kann es wichtig sein, zu wissen, welche Rechte Vorrang gegenüber anderen haben (Beispiel: Benutzerrechte sind „stärker“ als Gruppenrechte).

- **3.1.2** (2 Punkte) Sind die Bezeichnungen `private` und `public` für die Verzeichnisse sinnvoll gewählt? Welche Zugriffsrechte sollten die Verzeichnisse sinnvollerweise haben? Welche Befehle müsste `zeman` ausführen, um die Zugriffsrechte entsprechend zu setzen?

Aufgabe 3.2: Automatisierte Verarbeitung von Textdateien (5 Punkte)

Diese Aufgabe soll wie üblich in einem Terminal bearbeitet werden.
Im Verzeichnis `/group/cgl/2017/03` befindet sich eine Datei `RDF_Na-Cl_ger.dat` . Erstelle in deinem Heimatverzeichnis ein Verzeichnis `cgl_03` und kopiere die oben genannte Datei dorthin. Anschließend wechsle in das neu erstellte Verzeichnis und stelle sicher, dass sich die kopierte Datei auch dort befindet.
Bei der Datei handelt es sich um eine Textdatei, welche Daten aus der Molekulardynamik-Simulation einer wässrigen Kochsalzlösung ($NaCl_{(aq)}$) enthält. Die Daten stehen in der Datei in zwei Spalten, wobei die erste Spalte die x-Werte und die zweite Spalte die dazugehörigen y-Werte eines Schaubilds darstellt. Lass dir den Inhalt der Datei mit dem Befehl `less RDF_Na-Cl_ger.dat` anzeigen.

Hinweis: Das Programm `less` lässt sich wieder schließen, indem man die Taste `q` drückt (q steht hier für “quit”).

- **3.2.1** (1 Punkt) Die Daten sind in wissenschaftlicher Notation (“scientific notation”) abgespeichert. Finde heraus, wie diese Notation definiert ist und beschreibe, was die Bestandteile einer so dargestellten Zahl bedeuten.

Wie du wahrscheinlich bereits bemerkt hast, sind die Zahlen leider in *deutscher* Schreibweise dargestellt, wobei ein Komma als Dezimaltrennzeichen verwendet wird. Viele Programme, mit der sich die Daten einer solchen Datei grafisch darstellen lassen, erwarten standardmäßig aber die *englische* Darstellung mit einem Punkt als Dezimaltrennzeichen. Im Folgenden sollen mit Hilfe eines Kommandos im Terminal alle Zahlen der Datei in die englische Darstellung umgewandelt werden.

Das Programm `sed` kann dazu verwendet werden, um Textdateien automatisiert zu verarbeiten.
Zum Beispiel kann mit dem Befehl

```
sed 's/foo/BAR/g' file.txt
```

der Inhalt der Datei `file.txt` ausgegeben werden, wobei jedes Vorkommen der Zeichenkette „foo“ durch „BAR“ ersetzt wurde.

- **3.2.2** (2 Punkte) Verwende das Programm `sed` um alle Kommata in der Datei `RDF_Na-Cl_ger.dat` durch Punkte zu ersetzen. Die Ausgabe soll dabei in eine neue Datei `RDF_Na-Cl_en.dat` umgeleitet werden. Trage den verwendeten Befehl in deine Lösungsdatei ein.

  Hinweis: Um `sed` einen Punkt schreiben zu lassen, muss man dem Punkt einen Backslash voranstellen, also `\.` verwenden. Das liegt daran, dass ein einfacher Punkt ohne vorangestellten Backslash für das Programm eine spezielle Bedeutung hat. Falls du wissen möchtest, welche Bedeutung das ist, kannst du zunächst den Punkt (ohne Backslash) verwenden ohne die Ausgabe des `sed`-Kommandos in eine Datei umzuleiten. Wenn du nicht alleine darauf kommst, welche Funktion der Punkt erfüllt, kannst du deinen Tutor (oder alternativ das Internet) befragen.

Verwende den Befehl

```
echo "plot \"RDF_Na-Cl_en.dat\" with lines" | gnuplot -p &
```

um dir die Daten der in der vorigen Aufgabe erstellten Datei als Schaubild anzeigen zu lassen. Wie du sehen wirst, gehen die Daten auf der x-Achse von null bis fünf. Da das Verhalten der dargestellten Funktion aber für $x \geq 2$ recht uninteressant ist, sollen in den folgenden Teilaufgaben alle Zeilen aus der Datei `RDF_Na-Cl_en.dat` ausgelesen und wieder abgespeichert werden, welche die interessanten Werte ($x < 2$) enthalten.

- **3.2.3** (1 Punkt) Lass dir die Anzahl der Zeilen der Datei `RDF_Na-Cl_en.dat` mit einem geeigneten Befehl ausgeben. Trage den Verwendeten Befehl und seine Ausgabe in deine Lösungsdatei ein.

Da du nun sowohl den Bereich der x-Werte (null bis fünf) der Daten als auch die Gesamtanzahl der Datenpunkte (Zeilen) kennst, sollst du dir nun überlegen, bis zur wievielten Zeile die Datei Datenpunkte im interessanten Bereich ($0 \leq x < 2$) enthält.

- **3.2.4** (1 Punkt) Verwende den Befehl `head` mit geeigneten Optionen um die Datei bis zu dieser Zeile einzulesen und in eine neue Datei mit dem Namen `RDF_Na-Cl_en_cropped.dat` abzuspeichern. Trage den verwendeten Befehl in deine Lösungsdatei ein.

  Hinweise: Verwende `man head` um herauszufinden, wie du das anstellen kannst.
  Um zu überprüfen, ob das auch geklappt hat, kannst du anschließend wieder den Befehl von weiter oben verwenden, welcher dir das Schaubild anzeigt (allerdings mit dem *neuen* Dateinamen!). Das Schaubild sollte dann auf der x-Achse nur von null bis zwei gehen.
  Falls du dich dafür interessierst, was die Daten eigentlich zu bedeuten haben, kann dir das dein Tutor erklären!