Linux Kommandozeile: Einfache Skripte

AST, Wintersemester 2013/2014

1 Eine erste einfache Schleife

Eine einfache for-Schleife sieht folgendermaßen aus:

```
for i in 1 2 3 4 5
do
    echo $i
done
```

Eine for-Schleife erlaubt es etwas mehrfach auszuführen: der Schleifenkörper (alles zwischen "do" und "done") wird in diesem Beispiel insgesamt 5 mal ausgeführt, bei jedem Durchgang nimmt die Variable "i" einen anderen Wert an, zuerst 1, dann im nächsten Durchgang 2, usw. bis sie im letzten Durchgang den Wert 5 annimmt.

Alternativ kann man alles auf eine Zeile schreiben (praktischer wenn man alles direkt in der Kommandozeile eingibt), man muss dann mit Semikolons anzeigen wo normalerweise ein Zeilenende wäre:

```
for i in 1 2 3 4 5; do echo $i; done
```

Speichert man sein Skript in einer Datei ab empfiehlt sich die erstere Schreibweise wegen der besseren Leserlichkeit.

2 Shellskripte

Mit der Shell lassen sich nicht nur Befehle direkt eingeben, wir können auch Befehlsabfolgen die wir öfter ausführen wollen in einer Datei abspeichern, einem sog. "Shellskript". Dann können wir das Skript mit einem kurzen Befehl ausführen anstatt alles immer erneut eintippen zu müssen. Fangen wir mit einem kleinen Beispiel an, unserer Schleife aus dem vorherigen Abschnitt.

Öffnet mit dem Editor `kate` eine neue Datei, und gebt als erste Zeile

```
#!/bin/bash
```

ein. Diese Zeile sagt eurem Betriebssystem, dass es zum Ausführen der Datei den Bash-Interpreter `/bin/bash` mit dem Inhalt der Datei starten soll. Danach (in der nächsten Zeile, oder besser der übernächsten Zeile weil es besser aussieht) kommt die for-Schleife aus dem vorherigen Abschnitt. Speichert nun euer Skript unter einem Dateinamen der in ".bash"

endet, und merkt euch das Verzeichnis in dem ihr das Skript gespeichert habt. Geht nun mit der Konsole in das Verzeichnis.

Versuchen wir nun das Skript zu starten. Heisst das Skript "foo.bash" (im folgenden müsst ihr statt "foo.bash" den Namen den ihr eurem Skript gegeben habt verwenden) wird es mit

```
./foo.bash
```

gestartet. Leider bekommen wir beim ersten Versuch eine Fehlermeldung, da das Skript nicht ausführbar ist, d.h. die "execute permission" ist auf der Skriptdatei nicht gesetzt. Unter Linux (und Unix im allgemeinen) bestimmt nicht die Dateiendung ob eine Datei ausführbar ist, sondern die "execute permission". Sehen können wir dies wenn wir uns mit `ls -l` die genauen Informationen zu "foo.bash" anschauen:

```
ls -l foo.bash
```

Die Ausgabe wird in etwa so aussehen:

```
-rw-r--r-- 1 matthies users 33 2012-10-29 15:40 foo.bash
```

Der erste Eintrag `-rw-r--r--` sind die Dateirechte. Machen wir nun die Datei für alle Nutzer ausführbar:

```
chmod a+x foo.bash
```

Eine erneutes `ls -l` zeigt uns die Änderung:

```
-rwxr-xr-x 1 matthies users   33 2012-10-29 15:40 foo.bash
```

Die Ausführungsrechte sind die "x". Jetzt funktioniert das Starten des Skripts, probiert es aus!

3 Beispielskript: Umformatieren von vielen Daten

Angenommen wir haben ein Verzeichnis mit vielen Daten welche wir umformatieren möchten. Wir schreiben uns ein Skript um nicht mühsam alles per Hand machen zu müssen.

Als erstes kopieren wir die Daten und wechseln in das Verzeichnis:

```
cp -r ~matthies/uebung-linux-intro/data-for-reformat/ .
cd data-for-reformat
```

Schaut euch die Daten mit `less` an. Jede Datei hat den gleichen Aufbau, einen Kopf- und Fussbereich sowie Daten die zwischen zwei Markern START und END liegen. Wir wollen jetzt aus jeder Datei die zweite und fünfte Spalte der Daten extrahieren.

Wir legen wir uns ein Skript an um die Daten umzuformatieren, der Inhalt könnte in etwa so aussehen:

```
for f in data*
do
    sed -n '/START/,/END/p' "$f"    \
        | tail -n +2 | head -n -1   \
        | awk '{print $2 " " $5}'   \
        > "new-$f"
done
```

Das Skript legt dann für jede Datei eine neue Datei mit dem Präfix "new-" an. Der Backslash "\" sorgt dafür das ein Zeilenumbruch *keinen* neuen Befehl markiert, d.h. er erlaubt das leserliche Schreiben eines Befehls über mehrere Zeilen.

Erläuterung zu den Programmen: `sed` (der "stream editor") und `awk` sind recht komplexe Werkzeuge die hier zu erklären den Rahmen sprengen würde. Insbesondere `sed` ist nicht unbedingt einfach zu erlernen, häufig findet man aber im Internet mit etwas Suchen den richtigen Befehl. Die Stärken von `sed` liegen darin, wenn man einfache Textsubstitutionen machen will oder wie hier einfach bestimmte Bereiche herausfiltern will. Zwar kann man theoretisch alles mit `sed` machen (es enthält quasi eine vollständige Programmiersprache), für viele kompliziertere Sachen empfiehlt sich dann aber ein anderes Werkzeug. Das Programm `awk` ist ebenfalls eine vollständige Programmiersprache, wir benutzen hier vor allem die in `awk` eingebaute Fähigkeit, schnell auf einzelne durch Leerzeichen getrennte Elemente einer Zeile zugreifen zu können (`$2` und `$5` entsprechen dem 2. und 5. Element einer Zeile).