Linux Kommandozeile: Einfache Skripte

AST, Wintersemester 2016/2017

1 Wiederhohlung

Hier sind ein paar Befehle, die ihr letzte Woche schon kennen gelernt habt und heute benutzt.

	Befehl	Parameter	Funktion
Dateien und Verzeichnisse	`cp`	*Datei Ziel*	Datei kopieren
	`cp -r`	*Verzeichnis Ziel*	Rekursives kopieren von Verzeichnissen
	`rm`	*Datei*	Datei löschen
	`rm -r`	*Verzeichnis*	Rekursives löschen von Verzeichnissen
	`mkdir`	*Verzeichnis*	Neues Verzeichnis erstellen
	`wget`	*url*	Datei aus dem Internet laden
Navigation und Pfade	`.`		Aktuelles Verzeichnis
	`..`		Übergeordnetes Verzeichnis ("Vorgänger")
	`cd`		Wechsel in dieses Verzeichnis
	`ls`		Zeigt Inhalt des aktuellen Verzeichnis
	`pwd`		Gibt den aktuellen Pfad an
Ausgabe	`echo`	*Variable/Zeichenkette*	Ausgabe in der Konsole
	`grep`	*Zeichenkette Datei*	Zeichenkette oder Pattern suchen
	`|`		Ausgabe weiterleiten

2 Einfache Skripte

Auch wenn es beim Erlernen des Umgangs mit neuen Technologien nicht immer den Eindruck macht: Computer sollen uns Arbeit abnehmen - so viel wie möglich! Um nicht immer wieder die selben Befehle einzugeben, können Abläufe in Skripten zusammengefasst werden. Bash-Skripte sind Dateien mit der Endung .sh. Diese Skripte können dann jederzeit wieder aufgerufen werden um den Prozess automatisch ablaufen zu lassen.

Aufgabe 2.1
Erzeuge ein neues Verzeichnis und nutze `cd` um in dieses Verzeichnis zu wechseln. Erzeuge eine Datei mittels `touch helloWorld.sh`. Öffne diese mit dem Texteditor kate. Schreibe `echo hello world`, schliesse kate und führe das Skript mit `bash helloWorld.sh` aus.

Um nicht jedes mal `bash` zu schreiben, kann man in die erste Zeile des Skriptes

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

schreiben. Diese Zeile sagt dem Betriebssystem, dass diese Datei mit dem Bash-Interpreter `/bin/bash` ausgeführt werden soll. Versucht man jetzt das Skript mit dem Befehl

```
./helloWorld.sh
```

auszuführen, bekommt man eine Fehlermeldung. Das liegt daran, dass in Linux eine "execute permission" gesetzt sein muss, damit eine Datei als Programm ausführbar ist. Mit `ls -l` können wir uns die Rechte für die Datei genauer ansehen. Die Ausgabe sollte ungefähr so aussehen:

```
-rwxr--r-- 1 petersen users 30 Oct 25 13:29 helloWorld.sh
```

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

```
chmod a+x helloWorld.sh
```

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

```
-rwxr-xr-x 1 petersen users    30 Oct 25 13:29 helloWorld.sh
```

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

Aufgabe 2.2
Schreibe `#!/bin/bash` in die erste Zeile von `helloWorld.sh`. Ändere die Nutzungsrechte und führe die Datei mittels `./helloWorld.sh` aus. Probiere auch einmal den gesamten Pfad anstelle des relativen Pfades (`./`). (Den Pfad des aktuellen Verzeichnisses erhaltet ihr mit pwd.)

Noch ein Tipp: Beim Schreiben von Skripten es oft sinnvoll einzelne Befehle erst in der Konsole zu testen.

3 Ein zweites Skript

In der letzten Übung wurden Umgebungsvariablen beschrieben, z.B. `$HOME` und `$HOST`. `$USER` ist eine weitere Umgebungsvariable.

Aufgabe 3.1
Schreibe ein Skript, welches den Namen deines Computers (den "host") ausgibt.

Umgebungsvariablen lassen sich einfach mit weiteren Zeichenketten (aka. Wörtern) kombinieren. Probier doch zum Beispiel mal folgenden Befehl aus:

```
echo Home: $HOME
```

Aufgabe 3.2
Schreibe ein Skript, welches das Home-Verzeichnis, den Computernamen und den Nutzernamen folgendermaßen ausgibt:

```
User: petersen
Host: weissensee
Home: /home/petersen
```

Die genaue Ausgabe unterscheidet sich natürlich abhängig von Nutzer und Computer!

4 Atome zählen

Wir wollen wissen, wie viele Atome sich in einer Proteinstruktur befinden. Die Struktur befindet sich jedoch nicht auf dem eigenen Rechner, sondern in der PDB (Protein Data Bank).

Folgende Befehle wurden in der letzten Übung vorgestellt:

- Mit `wget` kann man eine Datei aus dem Internet laden.
- `gunzip` entpackt eine `.gz` Datei.
- `|` leitet die Ausgabe eines Programms in ein anderes weiter.
- Mit `grep` kann man Zeilen, welche eine bestimmte Zeichenkette enthalten, aus einer Datei filtern.
  - Beispiel: `grep ATOM 1igt.pdb` zeigt alle Zeilen mit dem Schlüsselwort "ATOM".

Ein neuer Befehl ist `wc` *Dateiname*. Damit zählt man Zeichen, Wörter und Zeilen in einer Datei. Mit `wc -l` werden nur die Zeilen gezählt. Probier es doch einmal mit einer beliebigen Datei aus!

Aufgabe 4.1

- Lade die Datei `1igt.pdb` aus dem Internet.
  - Die url ist `http://www.pdb.org/pdb/files/1igt.pdb.gz`.
- Entpacke die Datei.
- Nutze `grep`, `|` und `wc -l` um alle Zeilen in `1igt.pdb` mit dem Schlüsselwort `ATOM` zu zählen.

Aufgabe 4.2

Nun wollen wir ein Skript schreiben, dass die Zeilen mit dem Wort `ATOM` in der Datei `1igt.pdb` zählt ohne Spuren zu hinterlassen. Schreibe dafür ein Skrip. Das Skript soll die Datei herunterladen, die Atome zählen und anschließend aufräumen. Folgende Schritte sollen automatisch durchgeführt werden:

- Ein neues Verzeichnis erstellen.
- Mit `cd` in dieses Verzeichnis zu wechseln.
- Die Datei `1igt.pdb.gz` in diesem Verzeichnis herunterladen und entpacken.
- Wie zuvor die Atome zählen.
- Das Verzeichnis verlassen (`cd ..`).
- Das Verzeichnis löschen.

Tipp: Probiere die einzelnen Schritte zunächst in der Konsole aus. Füge sie dann im Skript zusammen.

Noch ein Tipp: Wenn du willst, dass `wget` und `grep` keine Ausgabe erzeugen, kannst du jeweils die Option `-q` nutzen.

5 Schleifen

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.

Aufgabe 5.1
Teste die beschriebene Schleife in der Konsole und schreibe ein Skript mit dieser Schleife. Ersetze die Zahlen durch Wörter, z.B. ”Aubergine”, ”Quadratwurzel” und ”Sousaphon”.

Aufgabe 5.2
Implementiere einen Countdown. Tausche dafüer die Reihenfolge der Zahlen, so dass von 10 nach 0 gezählt wird. Nach jedem `echo $i` nutze `sleep 1s` um den computer eine Sekunde warten zu lassen.

6 Die Umgebungsvariable PATH

Die Umgebungsvariable `$PATH` enthält die Pfade, in denen automatisch nach Programmen gesucht wird. Mit

```
echo $PATH
```

kann man sich den Inhalt ansehen. Befindet sich ein Programm in einem Verzeichnis, welches in `$PATH` aufgelistet ist, kann es ohne Angabe des Pfades ausgeführt werden. Will man ein Verzeichnis der `$PATH`-Variable hinzufügen, kann man das folgendermaßen tun:

```
PATH=$PATH:/home/petersen
```

Hier wurde das Homeverzeichnis des Nutzers petersen hinzugefügt. Das `$PATH` rechts vom `=` ist notwendig, um nicht alle bereits gelisteten Pfade aus `$PATH` zu verlieren.

Aufgabe 6.1
Erweitere `$PATH` um ein Verzeichnis, in dem sich ein selbst geschriebenes Skript befindet. Navigiere in ein beliebiges anderes Verzeichnis und führe das Skript aus ohne einen Pfad anzugeben (als auch kein `./`).

Öffnet man eine neue Konsole und gibt `echo $PATH` ein, stellt man fest, das `$PATH` wieder auf den unveränderten Zustand zurückgesetzt ist. Probier es mal aus! Man muss also `$PATH` jedes mal neu setzen.

Um das zu vermeiden, kann man die Pfaderweiterung in die Datei `.bashrc` eintragen. `.bashrc` ist eine Konfigurationsdatei die sich im Homeverzeichnis des Nutzers befindet. Beim starten der Konsole werden alle darin befindlichen Befehle ausgeführt.

Aufgabe 6.2
Navigiere in dein home-Verzeichnis. Prüfe mit `ls -a` ob die Datei `.bashrc` existiert. Falls nicht, erzeuge die Datei mit dem Befehl `touch`. Öffne `.bashrc` mit `kate` und erweitere `$PATH` in `.bashrc` wie zuvor für die Konsole beschrieben.

Auch nützlich: Mit `env` kann man sich alle Umgebungsvariablen ansehen.