Shell für Anfänger

Kurzüberblick über die häufigsten Befehle

Jens Dietze, Evgeni Golov, Janine Haas, Markus Brenneis

02.08.2019

Inhaltsverzeichnis

1 Shell? Was ist eine Shell?

„Die Unix-Shell oder kurz Shell (englisch für Hülle, Schale) bezeichnet die traditionelle Benutzerschnittstelle unter Unix oder unixoiden Computer-Betriebssystemen. Der Benutzer kann in einer Eingabezeile Kommandos eintippen, die der Computer dann sogleich ausführt. Man spricht darum auch von einem Kommandozeileninterpreter.“ [1]

Eine Shell wird später der wichtigste Begleiter im Linux[2]-Alltag sein, jetzt ist sie aber erst mal nur eine Kommandozeile, in die man ein paar Befehle eintippen kann. Du wirst an einigen Stellen auch das Wort *Terminal* oder *Konsole* lesen. Meist ist hier dasselbe gemeint, wenn auch die Wörter Unterschiedliches bedeuten: Terminal bzw. Konsole ist das Fenster, das die Darstellung übernimmt, wohingegen die Shell das Programm ist, dass die einzelnen Befehle in der Eingabezeile entgegennimmt.

Später wirst du feststellen, dass man mithilfe der Shell sehr viele coole Sachen machen (insb. kleinere Aufgaben automatisieren) und die Arbeit mit dem Computer sehr vereinfachen kann. Bis dahin ist aber noch ein langer Weg und du findest hier erst mal eine kleine Übersicht für den Einstieg.

Eine Sache möchten wir Dir schon sagen, bevor Du anfängst: Wenn die Shell keine Fehlermeldung ausgibt, ist alles gut gelaufen. Eine Bestätigung gibt es im Allgemeinen nicht.

1.1 Wie navigiert man?

Das oberste Verzeichnis heißt `/`, man erreicht es durch

```
cd /
```

Ansonsten navigiert man:

- ins Benutzer-Verzeichnis („Home-Verzeichnis“)

  ```
  cd
  ```

[1] Wikipedia, https://de.wikipedia.org/wiki/Unix-Shell, CC-by-sa-3.0, abgerufen am 02.08.2019

[2] oder auch macOS – Das meiste, was in diesem Dokument beschrieben ist, trifft auf alle Unix-ähnlichen Systeme zu; dazu zählen insbesondere Linux (auch Android) und macOS.

- zurück ins letzte Verzeichnis, in dem man war

  ```
  cd -
  ```

- ins Verzeichnis oberhalb vom aktuellen Verzeichnis

  ```
  cd ..
  ```

- Kurzform für das Home-Verzeichnis

  ```
  cd ~/<unterverzeichnis>
  ```

  ist äquivalent zu

  ```
  cd /home/<benutzername>/<unterverzeichnis>
  ```

- ins Verzeichnis <verzeichnisname>:

  ```
  cd <verzeichnisname>
  ```

- ins Verzeichnis <verzeichnis>/<unterverzeichnis>

  ```
  cd <verzeichnis>/<unterverzeichnis>
  ```

Tipp: Mit der Tabulatortaste[3] kann man sich Befehlsnamen, Parameter und Dateinamen vervollständigen lassen. Wenn es keine eindeutige Vervollständigung gibt, zeigt zweimaliges Drücken der Tabulatortaste alle Möglichkeiten an.

Die meisten Shells zeigen am Anfang der Eingabezeile an, in welchem Verzeichnis man sich aktuell befindet. Falls dies nicht der Fall ist, kann man sich mit dem Befehl `pwd` den Pfad des aktuellen Verzeichnisses anzeigen lassen.

1.2 Inhalt eines Ordners anzeigen

Damit man weiß, wohin man navigieren will, muss man den Inhalt eines Ordners kennen:

```
ls
```

Oder mit mehr Informationen:

[3] Auf deutschen Tastaturen links neben dem Q

```
ls -l
```

Auch versteckte Dateien (unter Linux[4] sind das solche, die mit einem . (Punkt) anfangen):

```
ls -a
```

Beides kombiniert:

```
ls -la
```

Auszug:

```
$ ls -la
drwxr-xr-x  22 root root  4096  6. Jul 12:30 .
drwxr-xr-x  22 root root  4096  6. Jul 12:30 ..
drwxr-xr-x   2 root root  4096  6. Jul 12:25 bin
drwxr-xr-x   3 root root  4096  6. Jul 12:41 boot
```

Erklärung:

1. Die Rechte:
   `d rwx rwx rwx`
   - `d`: es handelt sich um ein Verzeichnis (sonst „-“ an erster Stelle)
   - drei Gruppen mit „rwx“ bzw. „-“ an manchen Stellen
     - `r` = Leseberechtigung
     - `w` = Schreibberechtigung
     - `x` = Ausführberechtigung
     - `-` = diese Berechtigung ist nicht gesetzt
     - erste Gruppe („`rwx`“): gilt für den Besitzer der Datei
     - zweite Gruppe: gilt für die Gruppe der Datei
     - dritte Gruppe: gilt für alle anderen
2. der Besitzer:
   `root root` bedeutet: Besitzer: root, Gruppe: root
3. danach die Größe (4096 Byte)
4. Veränderungsdatum
5. Name

[4] und auch macOS

1.3 Wie erstellt man Ordner/Dateien?

- im aktuellen Ordner den Ordner <ordner> erstellen:

  ```
  mkdir <ordner>
  ```

- die komplette Ordnerstruktur /<ordner1>/<ordner2>/<ordner3> erstellen:

  ```
  mkdir -p /<ordner1>/<ordner2>/<ordner3>
  ```

- eine (leere) Datei erstellen, wenn sie bereits vorhanden ist, das Änderungsdatum auf die aktuelle Zeit setzen:

  ```
  touch <dateiname>
  ```

1.4 Wie löscht man diese?

Achtung: „Löschen“ heißt hier wirklich „löschen“ und nicht „in den Papierkorb verschieben“[5]. Es erfolgt standardmäßig keine Sicherheitsabfrage bei `rm`.

- eine Datei löschen:

  ```
  rm <dateiname>
  ```

- mehrere Dateien löschen:
  - alle jpg-Dateien im aktuellen Ordner:

    ```
    rm *.jpg
    ```

  - alle Dateien, die mit „a“ anfangen:

    ```
    rm a*
    ```

  - einen Ordner inklusive Unterordner und aller Dateien im Ordner:

    ```
    rm -r <ordnername>
    ```

[5] Dafür gibt es einen anderen Befehl: `trash`

1.5 Wie verschiebt oder kopiert man diese?

- Kopieren einer Datei:

  ```
  cp <pfad>/<dateiname> <ziel>/<dateiname>
  ```

- Kopieren eines Ordners (mit komplettem Inhalt)

  ```
  cp -r <pfad>/<dateiname> <ziel>/
  ```

- Verschieben einer Datei:

  ```
  mv <pfad>/<dateiname> <ziel>/<dateiname>
  ```

2 Spezielle Ordner und Partitionen

Man sollte vorher wissen, dass unter Linux alles ein eigener Ordner oder eine eigene Datei ist. Es gibt kein Laufwerk C: und D:, alles ist an einer sinnvollen Stelle unter / eingehängt („gemountet“). Das heißt, Dateisysteme werden dem Benutzer an bestimmten Stellen zugänglich gemacht.[6] Man hat also eine klare Baumstruktur mit einer Wurzel.

- `/`
  Das Wurzelverzeichnis („root-Verzeichnis“), der Anfang von allem sozusagen, die Wurzel
- `/bin`
  Wichtige Binärdateien (Programme), die für den Systemstart benötigt werden
- `/boot`
  Betriebssystem-Kernel und Bootloader-Konfiguration
- `/dev`
  Auch Geräte sind unter Linux Dateien, solche speziellen findet man hier; z. B. ist `/dev/sda` die erste Festplatte des Systems.
- `/etc`
  Globale Konfigurationsdateien des Systems (benutzerspezifische liegt in `/home`)

[6] Wikipedia, http://de.wikipedia.org/wiki/Mounten, 14.08.2011

- `/home`
  Benutzerdaten: Dokumente, Bilder, Konfigurationen – alles, was von einem Benutzer generiert wurde

- `/lib`
  Wichtige Bibliotheken, die für den Systemstart benötigt werden

- `/media`
  Externe Datenträger werden hier eingehängt

- `/mnt`
  Interne Datenträger gehören hier hin[7]

- `/proc`
  Auch Prozesse sind Dateien, die man hier manipulieren kann

- `/root`
  Das Homeverzeichnis des Administrators (root), immer außerhalb von `/home`

- `/sbin`
  Weitere Programme, die für den Systemstart nötig sind (und nur vom Administrator (root) ausgeführt werden dürfen)

- `/sys`
  Hier kann man Einstellungen des Kernels zur Laufzeit verändern

- `/tmp`
  Hier kann jeder temporäre Daten ablegen, die beim nächsten Neustart automatisch gelöscht werden

- `/usr`
  Programme, Bibliotheken und Dokumentation

- `/var`
  Programmdaten (z. B. die Dateien eines Webservers)

- `swap`
  Das Äquivalent zur Auslagerungsdatei unter Windows.

[7] Je nach verwendetem System fällt die Unterscheidung zwischen `/media` und `/mnt` weg.

3 Rechte

3.1 Wer ist Wurzel?

„root“ wird der Administrator genannt. Er darf überall Dateien und Ordner anlegen, ändern, löschen, etc. Dabei ist es egal, wem die Dateien/Ordner gehören. Um ein Programm mit root-Rechten auszuführen, schreibt man:

```
sudo <programmname>
```

Alternativ kannst Du mit

```
sudo su
```

eine ganze Session als root starten.

`sudo` ist ein Programm, welches in vielen Linux-Distributionen standardmäßig enthalten ist. Ansonsten kann man es nachträglich installieren (siehe Abschnitt 4).

Befehle sollte man nur dann als root ausführen, wenn man genau weiß, was die Befehle bewirken! Wenn man Befehle aus dem Internet kopiert, sollte man sich *immer* darüber informieren, was die Befehle tun (siehe Abschnitt 5).

3.2 Dateirechte ändern

root kann außerdem die Dateirechte von beliebigen Dateien ändern, also ändern, wer eine bestimmte Datei lesen, schreiben oder ausführen darf.

- den Besitzer einer beliebigen Datei/eines Ordners ändern:

  ```
  chown <benutzername>[:<gruppenname>] <dateiname>
  ```

- den Besitzer von allen Dateien in einem Ordner ändern:

  ```
  chown -R <benutzername>[:<gruppenname>] <ordnername>
  ```

- Rechte für beliebige Dateien ändern (alle dürfen diese Datei lesen):

  ```
  chmod a+r <dateiname>
  ```

  Es gibt für diesen Befehl viele Parameter. Hier daher die wichtigsten:

`a`	alle
`g`	Gruppe
`u`	Besitzer
`+`	Recht gewähren
`-`	Recht entziehen
`r`	Leseberechtigung
`w`	Schreibberechtigung
`x`	Ausführungsberechtigung

Zusammengefasst:

```
chmod {a,g,u}{+,-}{r,w,x} <dateiname>
```

oder auch mehrere Operationen auf einmal, z.B.

```
chmod a-w,u+wx <dateiname>
```

um in einem Befehl allen die Schreibberechtigung zu entziehen (`a-w`) und dem Besitzer Schreib- und Ausführrechte zu geben (`u+wx`).

3.3 Was darf ein Benutzer normalerweise?

Ein normaler Benutzer darf

- in Ordnern, die ihm gehören, Dateien und Ordner erstellen
- eigene Ordner/Dateien ändern oder löschen (bei Ordnern nur, wenn keine fremden Dateien darin enthalten sind)
- fremde Dateien nur dann ändern, wenn die Berechtigungen entsprechend gesetzt sind
- Rechte eigener Dateien setzen
- Besitzer eigener Dateien verändern

4 Software

4.1 Pakete

Unter Linux wird Software meistens in sogenannten Paketen angeboten. Ein Paket enthält neben der eigentlichen Software noch weitere Informationen

(z. B. welche andere Software zur Benutzung benötigt wird) und integriert die Software bestmöglich in das System. Die meisten Anbieter (Distributionen) fassen Paketsammlungen als Repositories zusammen. Das sind (meistens) Webserver mit vielen Paketen und einem Index dieser, mit dessen Hilfe man dann das passende (oder gewünschte) für sein System installieren kann.

4.2 Paketverwaltung

Damit der Nutzer nicht selber nachgucken muss, welches Paket welches andere braucht und wo genau man es herunterladen kann, gibt es eine Reihe von Tools zur Paketverwaltung.
Hier eine kleine Übersicht (für Debian, Ubuntu und darauf basierende Distributionen):

- `dpkg`
  Ist für das eigentliche Installieren (Entpacken, Kopieren der Dateien etc.) eines einzelnen Pakets. `dpkg` wird vom normalen Nutzer meist nicht benötigt, außer er möchte explizit ein Paket von Hand installieren und nicht aus dem Repository.

- `apt`
  Ist ein einfaches Tool zur Paketverwaltung. Man kann damit Pakete suchen, installieren, deinstallieren, Paketlisten verwalten etc.

- `aptitude`
  Ist ein fortschrittlicherer Paketverwalter der auf `apt` aufsetzt (arbeitet also nicht ohne `apt`). `aptitude` kann Paketabhängigkeiten besser auflösen und räumt beim Deinstallieren von Paketen besser auf als `apt`.

- Diverse grafische Programme
  Neben der Installation über die Konsole kann man Software unter Linux auch über grafische Programme installieren, die ähnlich einfach zu bedienen sind wie die App-Stores auf Handys. Meistens ist ein solches Programm schon vorinstalliert und im Anwendungsmenü unter Begriffen wie „Softwarecenter“ oder „Programmverwaltung“ zu finden. Einzelne, von Websites heruntergeladene Pakte[8] können in der Regel einfach durch Doppelklick installiert werden.

[8] Im Linux-Umfeld im Gegensatz zum Windows-Umfeld eher selten gemacht.

4.3 Was man mit Paketen so alles anstellen kann

In der Konsole kannst du Pakete mit `apt` oder `aptitude` (de)installieren und einiges mehr. Zum Installieren von Software benötigst du Rootrechte. Bei den hier vorgestellten Befehlen kannst du, je nach persönlicher Vorliebe, `apt` durch `aptitude` ersetzen.

- Paket, das <Suchbegriff> in Name oder Beschreibung beinhaltet, suchen:

  ```
  apt search <Suchbegriff>
  ```

- Paket <Paketname> installieren:

  ```
  sudo apt install <Paketname>
  ```

- Paket <Paketname> deinstallieren:

  ```
  sudo apt-get remove <Paketname>
  ```

- Aktualisieren von Paketen
  Im Normalfall wird dein Betriebssystem dich durch ein auffälliges Symbol irgendwo in der Kontrollleiste darauf hinweisen, dass es Aktualisierungen für bestimmte Pakete gibt. Willst du diesen Vorgang von der Shell aus aufrufen, benutzt du die Befehle:

  ```
  sudo apt update
  sudo apt upgrade
  ```

  Anders als man vielleicht zunächst vermutet installiert der erste Befehl keine Updates, sondern aktualisiert nur die

- Müllbeseitigung
  Hin und wieder finden sich „Karteileichen“ auf eurem System, die man mit

  ```
  sudo apt autoremove
  sudo apt clean
  ```

  beseitigen kann. Der erste Befehl entfernt alle Softwarepakete, die automatisch als Abhängigkeit von anderen Paketen installiert wurde. Der zweite Befehl leert den Paket-Zwischenspeicher.

4.4 Software entwickeln

Im ersten und zweiten Semester wirst du programmieren müssen. Viele von euch haben das vielleicht noch gar nicht oder nur innerhalb von grafischen Entwicklungsumgebungen, wie Eclipse oder IntelliJ, gemacht. Wenn du in der Shell deine Programme kompilieren und ausführen willst, benötigst du die folgenden Befehle. Beide Compiler („Übersetzer“) unterstützen eine Vielzahl von weiteren Parametern, die du jedoch zunächst nicht benötigen wirst.

4.4.1 Java

- kompilieren

  ```
  javac <Programmname.java>
  ```

- ausführen

  ```
  java <Programmname>
  ```

4.4.2 C

- kompilieren

  ```
  gcc -o <Programmname> <Dateiname.c>
  ```

- ausführen

  ```
  ./<Programmname>
  ```

5 manpages

Du wirst feststellen, dass wir euch bei weitem nicht alle Befehle und Parameter hier aufzeigen, die es so gibt. Zum Beispiel haben wir hier nichts zum Thema „Dateien suchen“ gesagt. Wann immer du nicht weißt, wie du einen Befehl benutzen solltest oder was er vielleicht noch alles kann, lohnt es sich, die manpage (Gebrauchsanleitung, engl. manual) zu diesem Befehl zu lesen. Die manpage eines Programms rufst du mit

```
man <Programmname>
```

auf. Um dir das genauer anzuschauen, ruf doch einfach mal

```
man man
```

auf.

Typische Befehle, die sehr, sehr nützlich sind, i. d. R. auf allen Systemen zur Verfügung stehen und über die man sich informieren sollte:

- `echo`
  Text auf der Konsole ausgeben
- `find`
  Suchen von Dateien
- `grep`
  Suchen innerhalb einer Datei
- `sed`
  Ersetzen von Zeichen in Text anhand
- `cat`
  Inhalte mehrere Dateien ausgeben
- `less`
  Seitenweises lesen von Textdateien
- `file`
  Anzeigen des Dateityps
- `curl`
  Herunterladen von Dateien
- `wc`
  Zählen von Zeichen, Wörtern und Zeilen
- `tar`
  tar-Archive erstellen
- `zip`
  zip-Archive erstellen
- `nano`
  Einfacher Texteditor

- `vi`
  Sehr mächtiger Texteditor
- `alias`
  Kurze Aliase für Befehle anlegen
- `xdg-open`[9]
  Dateien mit dem Standardprogramm öffnen

Diese Befehle lassen sich in kleinen Skripten kombinieren, um Aufgaben automatisiert ausführen zu können. Dabei kann man z. B. Variablen, Verzweigungen und Schleifen benutzen.

Außerdem gibt es für einige spezielle Aufgaben gute Programme, die in der Regel nicht standardmäßig installiert sind:

- `ffmpeg`
  Konvertieren und Manipulation von Audio- und Video-Dateien
- `pdftk`
  Manipulation von PDF-Dokumenten
- `csvcut`
  Spalten von CSV-Tabellen ausschneiden
- ...

6 Verbindung von Befehlen und Schreiben in Dateien

Wenn ein Befehl eine „normale“ Ausgabe auf der Konsole macht, landet diese in der sogenannten Standard-Ausgabe (stdout). Die Standardausgabe kann man mit `>` in eine Datei umleiten. Mit `ls > /tmp/dateien` wird beispielsweise die Ausgabe von `ls` in der Datei `/tmp/dateien` gespeichert.

Die Standard-Ausgabe kann man als Eingabe für einen anderen Befehl benutzen. Dazu benutzt man das Pipe-Symbol `|`[10]. Zum Beispiel sucht folgende

[9] unter macOS: `open`

[10] Auf deutschen Tastaturen mit AltGr+< eingebbar, bei Apple-Tastaturen mit Alt+7

Befehlsfolge zuerst alle Zeilen in der Datei `foo.txt`, die „Beispiel“ enthalten, und ersetzt dann alle Vorkommen von „schlecht“ durch „gut“: `grep Beispiel foo.txt | sed ß/schlecht/gut/"`

Mehrere Befehle können mit `&&` nacheinander ausgeführt werden. Der nachfolgende Befehl wird nur ausgeführt, wenn es beim vorhergegangenen Befehl keinen Fehler gab.

Darüber hinaus gibt es noch die Standard-Eingabe und die Fehler-Ausgabe, mit der man ähnlich arbeiten kann. Informationen gibt es u. a. mit `man stdout` und `man sh` (Abschnitt „Redirections“) und in diversen Internetquellen.

7 Editoren

Es gibt viele Konsoleneditoren. Diese sind jedoch nicht immer intuitiv benutzbar. An dieser Stelle möchten wir daher nur darauf hinweisen, dass es sie gibt. Beispiele für weit verbreitete Konsoleneditoren sind der einfache Editor *nano* und die sehr mächtigen Editoren *vim* und *emacs*, die aber eine sehr steile Lernkurve haben.

Darüber hinaus gibt es – wie unter Windows – auch grafische Editoren, die sich ohne größere Vorkenntnisse bedienen lassen. Je nach Distribution sind unterschiedliche Editoren bereits vorinstalliert. Beispiele sind **kate**, **gedit**, **kedit**, **kwrite**, **geany**, ... Egal welchen Editor man am Ende verwendet: Es ist wichtig, dass man mit seinen Funktionen vertraut ist. Typische Funktionen eines Texteditors für Programmierer sind Syntax-Highlighting, automatisches Einrücken, Tastenkombinationen für das Auskommentieren von Codeblöcken, ein integriertes Terminal usw.

8 Lustiges

Weil du bis ganz zum Ende durchgehalten hast, gibt es nun noch einen Tipp von uns, was du bei Gelegenheit mal ausprobieren kannst.

- `cowsay`
- `sl`

- `apt-get moo`
- `aptitude moo`

Wir wünschen dir nun viel Spaß beim Ausprobieren.