# Linux Cheatsheet

## Table of Contents

# 1. Basics

## 1.1. Filesystem-Navigation

Das Filesystem beginnt bei `/`, also der Root. Die User Directories, auch Home-Directory genannt, befinden sich bei `/home/<username>`. Das Home-Directory des eigenen Users kann auch über `$HOME` angesprochen werden.

### 1.1.1. Tilde

Die Tilde `~` führt meistens ins Home-Directory des jeweiligen Users. Es gibt allerdings oft noch weitere Funktionen:

- `~<username>` Führt zum Home-Directory von `<username>`
- `~-` Vorherige Working-Directory
- `~+` Momentanes Working-Directory

Weitere Infos: stackoverflow.com

### 1.1.2. Files und Directories

Es gibt einige Namenskonventionen, um das Arbeiten im Filesystem zu erleichtern:

- `file.txt` Normales File
- `dir/` Normales Directory
- `.hidden.txt` Mit einem Punkt als Prefix, wird angezeigt dass ein File oder Directory versteckt ist
- `.hidden_dir/` Verstecktes Directory
- `.` Momentanes Directory
- `..` Übergeordnetes Directory

### 1.1.3. Wichtige Files und Directories

Folgende Files und Directories können im Home-Directory gefunden werden. Sie

sind meist auch in `/etc/` gespeichert.

- `.bashrc` Dieses ist ein normales Bash-Script, welches beim Start eines neuen Terminal ausgeführt wird Es dient dazu Environment-Variablen, Aliases zu setzen oder andere Einstellungen vorzunehmen. Es wird auch häufig dazu genutzt eine Nachricht auszugeben, z.B.:

  ```
  echo "Hello World"
  curl wttr.in
  ```

- `.zshrc` Hat ähnliche Funktionalität, wie das `.bashrc`
- `.config/` Hier werden Config-Files für verschiedene Programme gespeichert, z.B. Google Chrome oder Virtual Box
- `.profile` Dieses File wird meist von der Shell als auch von Desktop-Environments ausgeführt

## 1.2. Wichtige Commands

### 1.2.1. Adminrechte

Mithilfe von `sudo` kann ein User sich als Admin anmelden oder auch nur ein einzelnes Command als Admin ausführen.

```
sudo <Command>
sudo rm file ①
```

① Nutzen von sudo um ein File zu löschen

### 1.2.2. Navigation

Mit `cd` kann man Directories wechseln.

```
cd path/to/dir/ ①
cd ②
cd - ③
```

① In ein Directory wechseln

② Wechselt ins Home-Directory

③ Ins vorherige Directory wechseln

`ls` listet die Inhalte eines Directories auf.

```
pwd ①
ls ②
ls <directory> ③
ls -l ④
ls -a ⑤
ls -h ⑥
```

① Zeigt absoluten Pfad zum momentanen Directory

② Zeigt Files/Directories im momentanen Directory an

③ Zeigt Files/Directories im `<Directory an>`

④ Files/Directories in Listenstruktur anzeigen

⑤ **Alle** Files/Directories anzeigen, inklusive der Versteckten

⑥ Nur von Menschen lesbare Files anzeigen

### 1.2.3. Erstellen von Files und Directories

```
touch <path/to/file.txt> ①
touch {file1,file2,file3} ②
mkdir <directory> ③
mkdir {dir1,dir2,dir3} ④
mkdir -p parent/dir ⑤
```

① Erstellt ein File

② Erstellt `file1, file2, file3`

③ Erstellt ein Directory

④ Erstellt `dir1, dir2, dir3`

⑤ Erstellt Directories rekursiv

### 1.2.4. Löschen von Files und Directories

```
rm <file> ①
rmdir <dir> ②
rm -r <dir> ③
rm -f <file> ④
```

① Löscht ein File

② Löscht ein leeres Directory

③ Löscht ein Directory und dessen Inhalt

④ Löscht zwingend ein File

### 1.2.5. Kopieren und Verschieben von Files und Directories

```
cp <from> <to> ①
cp -r <from> <to> ②
mv <from> <to> ③
```

① Kopiert File von A nach B

② Kopiert Directory rekursiv

③ Verschiebt File/Directory

### 1.2.6. Alias

Mit einem Alias kann man häufig genutzte Commands ersetzen.

```
alias ll='ls -lh'
```

### 1.2.7. Text-Editing Tools

Mit folgenden Tools können Files manipuliert werden, sie werden allerdings nicht verändert. Viele dieser Tools funktionieren nur mit CSV-Files.

#### wc

Tool zum Zählen von Zeichen, Zeilen, Wörtern, etc.

```
wc -l ①
wc -w ②
```

① Zählen von Zeilen

② Zählen von Wörtern

### head/tail

`head` gibt die ersten `n`-Zeilen eines Inputs, z.B. File, aus.

`tail` gibt die letzten `n`-Zeilen aus.

```
head -3l file.txt ①
tail -3l file.txt ②
```

① Ersten drei Zeilen

② Letzten drei Zeilen

### cut

Entfernt bestimmte Felder eines CSV-Files.

```
cut -d'|' -f1,3 file.txt
-d'|' ①
-f1,3 ②
```

① Gibt als Trennzeichen eine Pipe an

② Es werden nur die erste und dritte Spalte ausgegeben

### sort

Sortiert ein File zeilenweise.

```
sort -t'|' -k1 -r -n file
-t'|' ①
-k1 ②
-r ③
-n ④
```

① Als Trennzeichen wird eine Pipe angegeben

② Es wird nach der ersten Spalte sortiert

③ In umgekehrter Reihenfolge sortieren

④ Nummerisch sortieren

### uniq

Doppelte Werte werden gelöscht. **Funktioniert nur bei sortierten Files!**

```
uniq -c file ①
```

① Lösche doppelte Werte und gibt die Anzahl der gleich vorkommenden Werte aus

### grep

Mit `grep` kann ein File zeilenweise gefiltert werden. Es werden nur Zeilen ausgegeben, welche mit einem vorher gegebenem Pattern übereinstimmen.

```
grep -c 'Hello' test.txt ①
grep -E 'H[ea]llo' test.txt ②
```

① Mit `-c` werden die gefundenen Zeilen gezählt

② Um Extended-Regex nutzen zu können muss die Option `-E` gegeben sein

#### grep Regex

Extended Regex von `grep` ist unterschiedlich zum normalen Regex.

- `[[:alnum:]]` Alphanumerische Characters
- `[[:alpha:]]` Alphabetische Characters
- `[[:blank:]]` Space und Tab
- `[[:digit:]]` Ziffern
- `[[:lower:]]` Kleinschreibung
- `[[:space:]]` Tab, Newline, Vertical Tab, Page Break, Carriage Return, und Space
- `[[:upper:]]` Großschreibung

### sed

Der Streaming Editior kann genutzt werden, um Text zeilenweise zu manipulieren.

```
sed 's/<regex>/<replace>/' file ①
sed -r 's/<regex>/<replace>/<flag>' file ②
sed 'n,ms/<pattern>/replace/' ③
sed '/<line_pattern>/s/<find>/<replace>/' <file> ④
sed '/<regex>/d' <file> ⑤
sed -n '<n>,/^$/p' <file> ⑥
sed 's#<regex>#<replace>#' <file> ⑦
sed -r 'n,m {<commands>}' ⑧
```

① Ersetzt pro Zeile einmal den `<regex>` durch `<replace>`

② Mit `-r` wird der extended Regex genutzt. Am Ende kann auch eine Flag angegeben werden, z.B. die `g` Global Flag, mit dieser wird nicht pro Zeile, sondern alles gematched

③ Ersetze das pattern in den Zeilen `n` - `m`

④ Ersetzt nur die Matches in den Zeilen, welche das `<line_pattern>` erfüllen

⑤ Löscht die Zeilen, welche einen Match haben

⑥ Gibt alle Zeilen von `<n>` bis zur nächsten leeren Zeile aus

⑦ Ein Slash `/` als Trennzeichen kann auch durch ein anderes ersetzt werden, z.B. eine Raute `#`

⑧ Führe `command` beim `n` ten auftreten bis Zeile `m` aus

# 2. Packagemanager

Auf Linux Systemen wird das Installieren von Software von einem Packagemanager übernommen. Im Fall von Debian basierten Distros, dazu gehört auch Ubuntu, ist das `apt`. Meist muss zum Installieren und Entfernen von Paketen `sudo` genutzt werden.

```
apt install <package> ①
apt search <query> ②
apt remove <package> ③
```

① Installiere ein Paket

② Suche ein Paket

③ Entferne ein Paket

# 3. Scripting

## 3.1. Piping

Mithilfe der Pip | kann die Ausgabe eines Befehls zum nächsten weitergeleitet werden.

```
cat file.txt | grep "hello"
```

## 3.2. Ausgabeumleitung

Output kann auch in Files umgeleitet werden, diese werden, sollten sie nicht existieren, automatisch angelegt

```
echo "Hello" > hello.txt ①
echo "Hello" >> hello.txt ②
ls dir/ 1> file.txt ③
ls not/extisting/dir 2> error.txt ④
```

① File wird komplett überschrieben

② Ausgabe wird am Ende des Files angefügt

③ Standard-Output(stdout) wird umgeleitet

④ Error-Output(stderr) wird umgeleitet

## 3.3. Environment-Variablen

Environment-Variablen sind Variablen, welche einen oder keinen Wert beinhalten. Diese Variablen sind allerdings nicht überall verfügbar.

```
HELLO="Hello, "$USER ①
echo $HELLO ②
export $HELLO ③
```

① Die `$USER` Variable wird beim Login gesetzt und enthält den momentan aktiven. `$HELLO` ist nur für die momentane Shell verfügbar

② Gibt den Inhalt von `$HELLO` aus

③ Nun ist `$HELLO` auch für die Subprozesse der Shell verfügbar.

Um eine permanente Variable zu erzeugen, muss diese ins `.profile` File geschrieben werden. Dadurch ist sie weitgehend verfügbar.

Soll die Variable nur für Shells verfügbar sein, muss diese ins `.bashrc` geschrieben werden. Dadurch ist sie nur für die Bash verfügbar.

## 3.4. awk

`awk` ist eines der mächtigsten Tools für Shell Scripting. Es beinhaltet eine eigenen Scripting Sprache, welche Touring Complete ist. `awk` ist gedacht für Oneliner-Scripts, es können aber auch größere Scripts entwickelt werden. Am besten wird es mit CSV Files genutzt, da `awk` meist zeilenweise arbeitet.

### 3.4.1. Syntax

Der Anfang eines Scripts enthält eine Condition, so wie ein `if`, der zweite Teil eine Aktion, welche ausgeführt werden soll. Man kann einzelne Spalten eines Files mit `$n` ansprechen, wobei `n` durch die Spaltennummer ersetzt wird. `$0` ist die gesamte Zeile, welche übergeben wird.

```
$2 > 2 { print $0} ①
$2 > 2 { print $0 > bigger_than_two.txt} ②
```

① Es wird überprüft ob der Wert in Spalte 2 Zeile `n` größer als 2 ist, wenn dies zutrifft, wird die gesamte Zeile `0` ausgegeben

② Gleich wie oben, allerdings werden die Zeilen in ein File gespeichert

# 4. Rechte

| | |
|---|---|
| U | User Rechte |
| G | Group Rechte |
| O | Other Rechte |
| A | All Rechte |

## 4.1. File/Folder Rechte

### 4.1.1. Folder

- Read(4): Inhalt eines Ordners ansehen
- Write(2): Inhalt eines Ordners löschen
- Execute(1): In einen Ordner wechseln
- Sticky Bit(1): Nur der Besitzer eines Files/Dir darf dieses entfernen/verschieben, z.B. `/tmp`
- SGID(2): Alle Unterverzeichnisse/Files werden der Gruppe zugeordnet, welcher das Parentdirectory gehört
- SUID(4): Alle Unterverzeichnisse/Files werden dem Owner des Parentdirectorys zugeordnet

### 4.1.2. Files

- Read(4): Inhalt eines Files ansehen
- Write(2): Inhalt eines Files ändern
- Execute(1): File ausführen, z.B. Script
- Sticky Bit(1): Wird ignoriert
- SGID(2): Der User hat temporär die Rechte der Gruppe, wenn ausgeführt
- SUID(4): Ein File wird mit den Rechten des Owners ausgeführt, z.B. `passwd`

### 4.1.3. Owner/Group ändern

```
chown <user> <file/dir> ①
chgrp <group> <file/dir> ②
```

① Ändert den Owner, mit `-R` rekursiv

② Ändert die Gruppe

## 4.2. User Actions

```
useradd <username> ①
userdel <username> ②
```

① Fügt einen User hinzu. Mit der Option `--create-home` wird auch das Home-Directory erstellt

② Löscht User. Die Option `-r` entfernt alle Files/Dirs, welche initial erstellt wurden

## 4.3. Group Actions

```
groupadd <group> ①
groupdel <group> ②
adduser <user> <group> ③
delgroup <group> ④
```

① Fügt eine Gruppe hinzu

② Löscht eine Gruppe

③ Fügt einen User zu einer Gruppe

④ Entfernt User aus Gruppe

# 5. Hilfe!

## 5.1. Man

Mithilfe von `man` kann man sich die Manual-Page anschauen. Die meisten Commandline-Tools besitzen eine.

```
man <command>
man ls
```

## 5.2. TL;DR

Ein Tool mit Zusammenfassungen und Beispielen für verschiedene Commands: https://tldr.sh

### 5.3. Regex

https://regexr.com

## 6. Fancy Shell

Funktioniert nur mit Debian-basierten Distros.

Folgende Befehle sind nicht als komplettes Script zu sehen!

### 6.1. Einfache Variante

```
sudo apt install fish ①
chsh -s /usr/bin/fish ②
```

① Installiert die Fish-Shell

② Setze die Standard-Shell auf Fish-Shell

### 6.2. Komplexe Variante

Im gegensatz zur einfachen Variante, wird hier das Framework oh-my-zsh genutzt. Mit diesem können Plugins und Themes verwaltet werden.

```
sudo apt install zsh git curl ①
chsh -s /usr/bin/zsh ②
sh -c "$(curl -fsSL
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)" ③
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ④
echo "source ${(q-)PWD}/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh" \
    >> ${ZDOTDIR:-$HOME}/.zshrc
git clone https://github.com/zsh-users/zsh-autosuggestions \
    ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions ⑤
nano ~/.zshrc ⑥
```

① Installiere die ZSH-Shell, Git und curl

② Setze die die Standard-Shell auf ZSH-Shell

③ Installiere OH-MY-ZSH

④ Installiere das Syntax-Highlighting Plugin

⑤ Installiere das Auto-Suggestions Plugin

⑥ Nun müssen die Plugins im `plugins` Array hinzugefügt werden