

# Arbeiten in der Unix-Kommandozeile

```
[ismo@it ~]$ _
```

Was ist das?

Muss das sein?

Ist das nicht völlig veraltet?

Das sieht nicht so schick aus...

## Motivation

- Die meisten Geräte basieren auf Unix
  - Server, Cluster, Supercomputer
  - Smartphones
  - Router, Drucker, ...
- Wissenschaftliche Programme werden für Unix geschrieben
  - Bedienung über Kommandozeile
  - Wichtige Programme haben keine GUIs
  - z.B. bei der Bachelorarbeit...

## Motivation

- Kommandozeile ist überlegenes Bedienkonzept
  - Die meiste Zeit beim Arbeiten verbringen wir im CLI
- GUI versteckt die Details
  - Details werden wichtig
  - GUI kann versagen
- GUIs sind nicht böse oder schlecht, man muss nur wissen, was dahinter steckt
- In der Kommandozeile ist alles automatisierbar
  - Wenn man etwas zum dritten Mal tut, sollte man ein Skript dafür schreiben

## Dateisystem

- bildet *einen* Baum
  - beginnt bei / (root)
  - / trennt Teile eines Pfads
  - auf Groß-/Kleinschreibung achten!
- es gibt ein aktuelles Verzeichnis
- relative vs. absolute Pfade
- spezielle Verzeichnisse:
  - . das aktuelle Verzeichnis
  - .. das Oberverzeichnis
  - ~ das Homeverzeichnis

man, pwd, cd

<code>man <i>topic</i></code>	„manual“: zeigt die Hilfe für ein Programm
<code>pwd</code>	„print working directory“: zeigt das aktuelle Verzeichnis
<code>cd <i>directory</i></code>	„change directory“: wechselt in das angegebene Verzeichnis

## ls

- `ls [directory]` „list“: zeigt den Inhalt eines Verzeichnisses an
- `ls -l` „long“: zeigt mehr Informationen über Dateien und Verzeichnisse
- `ls -a` „all“: zeigt auch versteckte Dateien (fangen mit `.` an)



## mkdir, touch

<code>mkdir <i>directory</i></code>	„make directory“: erstellt ein neues Verzeichnis
<code>mkdir -p <i>directory</i></code>	„parent“: erstellt auch alle notwendigen Oberverzeichnisse
<code>touch <i>file</i></code>	erstellt eine leere Datei

## cp, mv, rm, rmdir

<code>cp source destination</code>	„copy“: kopiert eine Datei
<code>cp -r source destination</code>	„recursive“: kopiert ein Verzeichnis rekursiv
<code>mv source destination</code>	„move“: verschiebt eine Datei (Umbenennung)
<code>rm file</code>	„remove“: löscht eine Datei (Es gibt keinen Papierkorb!)
<code>rm -r directory</code>	„recursive“: löscht ein Verzeichnis rekursiv
<code>rmdir directory</code>	„remove directory“: löscht ein <i>leeres</i> Verzeichnis

## cat, less, grep, echo

<code>cat file</code>	„concatenate“: gibt Inhalt einer (oder mehr) Datei(en) aus
<code>less file</code>	(besser als more): wie cat, aber navigabel
<code>grep pattern file</code>	g/re/p: sucht in einer Datei nach einem Muster
<code>grep -i pattern file</code>	„case insensitive“
<code>grep -r pattern directory</code>	„recursive“: suche rekursiv in allen Dateien
<code>echo message</code>	gibt einen Text aus

## Ein- und Ausgabe

<code>command &gt; file</code>	überschreibt Datei mit Ausgabe
<code>command » file</code>	fügt Ausgabe einer Datei hinzu
<code>command &lt; file</code>	Datei als Eingabe
<code>command1   command2</code>	Ausgabe als Eingabe (Pipe)

## Tastaturkürzel

**Ctrl-C** beendet das laufende Programm

**Ctrl-D** EOF (end of file) eingeben, kann Programme beenden

**Ctrl-L** leert den Bildschirm

## Globbering

- \* wird ersetzt durch alle passenden Dateien
- {a, b} bildet alle Kombinationen

### Beispiele:

\*.log → foo.log bar.log

foo.{tex,pdf} → foo.tex foo.pdf

## Shell-Skripte

- Datei enthält Befehle
- Selbe Syntax wie Kommandozeile
- Endung: keine oder `.sh`
- Ausführung:
  - `bash skript`
  - `./skript` (mit Shebang)
- Shebang: erste Zeile enthält Pfad des Interpreters (muss absolut sein)
  - `#!/bin/bash`