Arbeiten in der Unix-Kommandozeile

PeP et al. Toolbox Workshop

PEP ET AL. E.V.
PHYSIKSTUDIERENDE UND
EHEMALIGE PHYSIKSTUDIERENDE
DER TU DORTMUND

2015

```
[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

`man` *`topic`*	„manual“: zeigt die Hilfe für ein Programm
`pwd`	„print working directory“: zeigt das aktuelle Verzeichnis
`cd` *`directory`*	„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`

`mkdir` *`directory`*	„make directory“: erstellt ein neues Verzeichnis
`mkdir -p` *`directory`*	„parent“: erstellt auch alle notwendigen Oberverzeichnisse
`touch` *`file`*	erstellt eine leere Datei, falls sie noch nicht existiert
	ändert Bearbeitungsdatum auf „jetzt“

cp, mv, rm, rmdir

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

cat, less, grep, echo

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

Ein- und Ausgabe

`command > file`	überschreibt Datei mit Ausgabe
`command >> file`	fügt Ausgabe einer Datei hinzu
`command < file`	Datei als Eingabe
`command1 | command2`	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

Globbing

`*`	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`

Umgebungsvariablen

- steuern viele Einstellungen und Programme
- Ausgabe mit `echo $NAME`
- wichtiges Beispiel: `PATH` (auch unter Windows):
  - enthält alle Pfade, in denen nach Programmen gesucht werden soll
  - wird von vorne nach hinten gelesen
  - erster Treffer wird genommen
  - `which` *`program`* zeigt den Pfad eines Programms
  - Shebang, das den ersten Treffer im `PATH` nutzt, statt festem Pfad: `#!/usr/bin/env python`
- Änderung über `export`:

```
export PATH=/home/maxnoe/anaconda3/bin:$PATH
```

Config-Files

- Einstellungen für viele Programme werden in Textdateien gespeichert
- Üblicherweise versteckte Dateien im `HOME`-Verzeichnis
- Einstellungen für die Konsole an sich: `.bashrc` (Linux) bzw. `.bash_profile` (Mac)
- Bash-Befehle die beim Start jeder Konsole ausgeführt werden
- Umgebungsvariablen setzen
- Sehr nützlich: `alias`, definiert Alternativform für Befehle

```
alias ll='ls -lh'
alias gits='git status -s'
alias ..='cd ..'
```

- Müssen nach Änderungen neugeladen werden:

```
source ~/.bashrc
```