

# Arbeiten auf der Commandozeile

Björn, Milan & Co.

2012-10-18 Thu

# Outline

- 1 Einleitung
- 2 Grundlagen
- 3 Beispiele

# Wozu eigentlich?

- Bedienung eines Rechners, ggfs. nichtgrafisch ueber ssh ("Server").
- Szenario: Man will auf einem Server den von allen Nutzern verwendeten Speicherplatz zusammenrechnen und das ganze als Textdatei abspeichern.

```
du -s * | sort -nr > $HOME/user_space_report.txt
```

# Warum eine GUI Umgebung nicht ausreicht

- GUIs sind nuetzlich und sinnvoll fuer viele verschiedene Anwendungsgebiete und Aufgaben.
- Symbolischer zugriff auf vorgefertigte Ablaeufe per Icon und Mausclick sind zwar bequem, aber unflexibel und koennen zu ungeniessbaren Workflows fuehren.
- RSI (Tennisarm) durch Mausschubsen.
- Repetitives stupides Arbeiten im GUI; Beispielaufgabe: Finde alle Textdateien in einer mehrfach verschachtelten Verzeichnisstruktur die eine gewisse Zeichenkette beinhalten und ersetze diese Zeichenkette durch eine andere. -> Typischer Fall fuer ein Script.
- GUIs und Desktopumgebungen aendern sich sehr oft und sind alles andere als uniform.

# Die Vorteile der Commandozeile

- Einmal ausgeführtes bleibt in der History (C-r fuer Rueckwaertssuche)
- Man kann in einer Sprache (Syntax & Semantik) interaktiv mit dem Computer “reden” und ihm sehr praezise mitteilen was man will und ist nicht auf eine “Babysprache” beschraenkt.
- Scriptbarkeit “fuer spaeter”

# Ein paar oft benutzte Kommandos (1)

- pwd
- ls
- cd
- mv
- cp
- rm
- mkdir
- which
- cat
- less
- tar
- man / info
- bash
- env
- chmod / chown

# Ein paar oft benutzte Kommandos (2)

- top
- grep
- locate
- find
- cat
- echo
- alias
- free
- ps
- df
- fg
- jobs
- kill
- nohup
- su / sudo

# Verzeichnisstruktur

- Hierarchie(!) Wurzel “/” ist “ganz oben” im Dateisystem (und nicht “My Computer” oder “Arbeitsplatz”).
- “/home/\$USER” bzw “~” ist das Zuhause fuer persoenliche Nutzerdateien und Einstellungen (dotfiles)
- “.” ist immer das aktuelle Verzeichnis, “..” bedeutet “eine Ebene hoeher”
- Andere Verzeichnisse unter / beinhalten z.b. Systemweite Einstellungen, geteilte Dateien, Dokumentationen, Quellcodes, Binaerdateien (Pedant zu .exe unter Windows).
- Verzeichnisse, Geraete und anderes Dinge koennen beliebig in die Hierarchie eingebunden werden (mount).
- Umbenennen und Verschieben ist ein und dasselbe (denn nur der inode-eintrag im Dateisystem geaendert wird)!
- Symbolische und “Harte” Verlinkung ist moeglich mit “ln”.



# Ein paar Dinge zu Dateinamen

- Ein mit einem Punkt “.” beginnender Dateiname steht fuer eine “versteckte” Datei oder “dotfile” (z.b. lokale Systemeinstellungen)
- Gross-/Kleinschreibung ist WICHTIG.
- Dateiendungen sind rein optional!
- Leerstellen sind moeglich, sollten aber vermieden oder “escaped” werden!
- Bitte bitte bitte keine Minuszeichen als erstes Zeichen im Dateinamen! (geht zwar, tut aber weh. Loesung “-” als Trenner)

# Was fuer Arten von Kommandos gibt es?

- Ausfuehrbare (chmod +x) dateien (im \$PATH)
- Shell-builtins (z.b. cd, echo, alias)
- (selbstdefinierte) Shell-funktionen
- Aliase
- Was und wo ist was?
  - type
  - which
  - whereis
  - echo \$PATH

# Hilfe zur Selbsthilfe

- Immer ausprobieren: `-help` Kurzhilfe
- `manpages` und `infopages(!!!)` (erfordert oft `*-doc` pakete)
- `apropros` (`== man -k`)
- `help fuer shell buildins`

# grep (Suchfunktion fuer Dateiinhalte mit oder ohne regex)

- `grep -i "linuxparty" meinedatei`
- `grep -A 3 -i "pizza" meinedatei`
- `grep -r "hunger" *` (oder "rgrep")

# find und locate (Suchfunktion fuer Dateinamen)

- locate suchstring
- updatedb
- find -iname "\*party"
- find . -type f -iname \*.sh -print0
- find -iname "MyCProgram.c" -exec md5sum {

```
find / -iname *.jpg -type f -print | xargs \  
tar -cvzf images.tar.gz
```

- `chmod ug+rwx meinedatei`
- `chown meinuser:meinegruppe meinedatei`

# Liste meistbenutzter Kommandos

```
history | awk '{CMD[$2]++;\  
count++;}END \  
{ for (a in CMD)print CMD[a]\  
" " CMD[a]/count*100 "% " a;}' \  
\| grep -v "./" | column -c3 -s \  
" " -t | sort -nr | nl | head -n10
```

# I/O umleiten

```
ls > dateilisting.txt
sort < dateilisting.txt > sortiertes_dateilisting.txt
ls -la | less
ls -lt | head
du -h | sort -nr
find . -type f -print | wc -l
echo "test" > neuedatei
echo "muhalol" >> bereitsvorhandenedatei
```