HowTo: Linux

Ruben 14felgenh, Hauke 14stieler

6. Dezember 2022

Terminal Historie

- 19. Jahrhundert Drucker für Telegramme
- Später **T**ele**ty**pe-writer (TTY)
  - Gerät für Ein-/Ausgabe an Computern = Terminal
  - Auch per Telefonkabel
  - UNIX besaß TTY Treiber
- Dumb Terminals (bis 1970er)
- Smart Terminals (ab Mitte 1960er)
- Virtuelle/Pseudo Terminals
  - **P**seudo-Tele**ty**pe – PTY
  - ab Mitte 1990er
  - Terminal-Emulatoren (z.B. `konsole`, `xterm` oder `gnome-terminal`) verbinden sich mit PTY

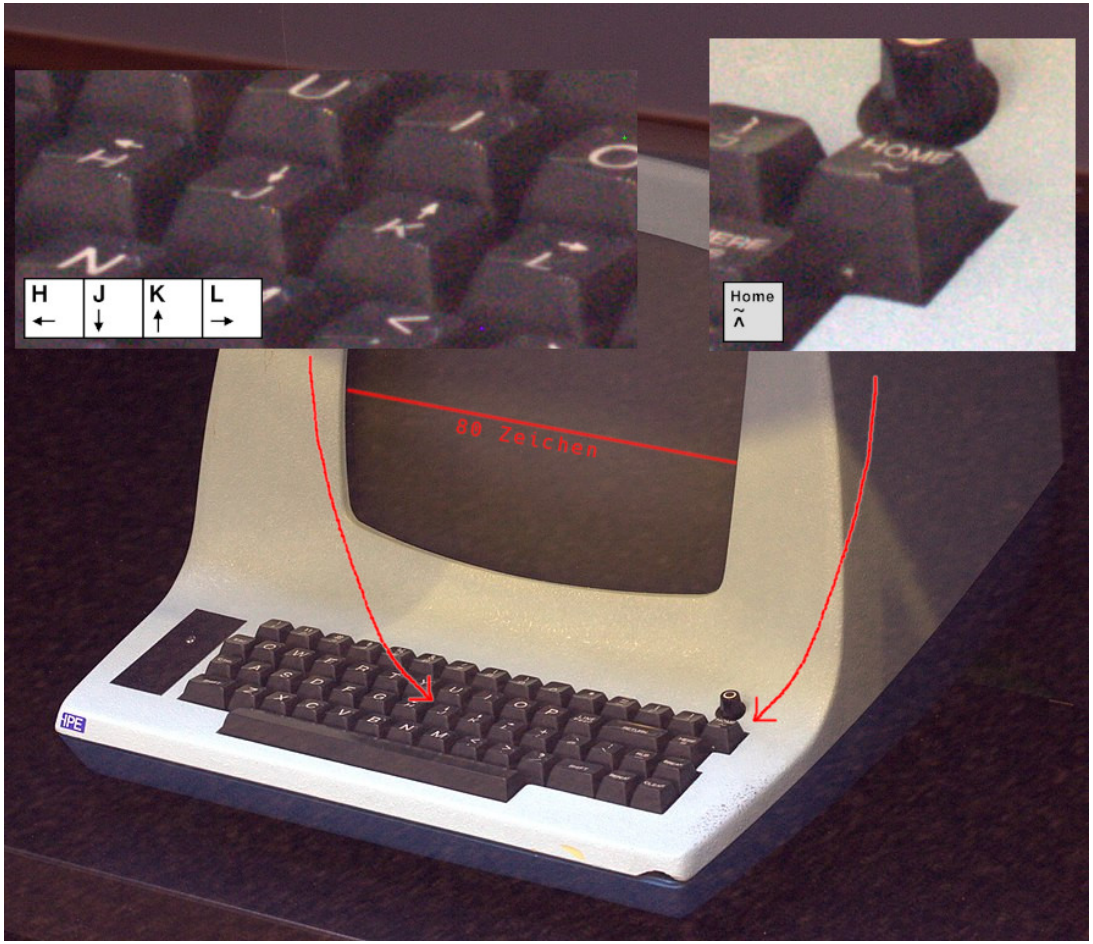
Historie

Teletype ASR-33 (1963) mit Papierrolle als Ausgabe.

Historie

Terminal ADM-3A (1976)

Historie



Terminal ADM-3A (1976): „~“ (Tilde) auf „Home“-Taste, Pfeile auf H J K L Tasten & 80 Zeichen pro Zeile.

Begriffe

- Kommandozeile / Konsole / CLI
- Shell
  - `sh`
  - `bash`
  - `zsh`
  - `ksh`
  - `csh`
  - `dash`
  - `fish`
- Terminal
- Terminal-Emulator

Befehle

- Shell-Built-Ins:
  - `pwd`
  - `cd`
  - `echo`
  - `true`
  - `false`
  - `type`
- Programme:
  - `firefox`
  - `touch`
  - `ls`
  - `cat`
  - `less`
  - `man`

Argumente u.Ä.

Es gibt Flags...

- `firefox --help`
- `firefox -h`

... Parameter, ...

- `type echo`
- `man echo`

... und Argumente.

- `ls --color=always`
- `dd if=/dev/urandom count=1`

Ein- und Ausgaben

- `stdout`
- `stderr`
- `stdin`
- Exit Codes
- Ausgaben in Dateien umleiten:
  `echo foobar > foobar.txt`
- Ausgaben als Eingaben für andere Programme verwenden:
  `echo foobar | rev`
- Ausgaben in Variablen speichern:
  `meinname=$(whoami)`
  `echo $meinname`

History und Autovervollständigung

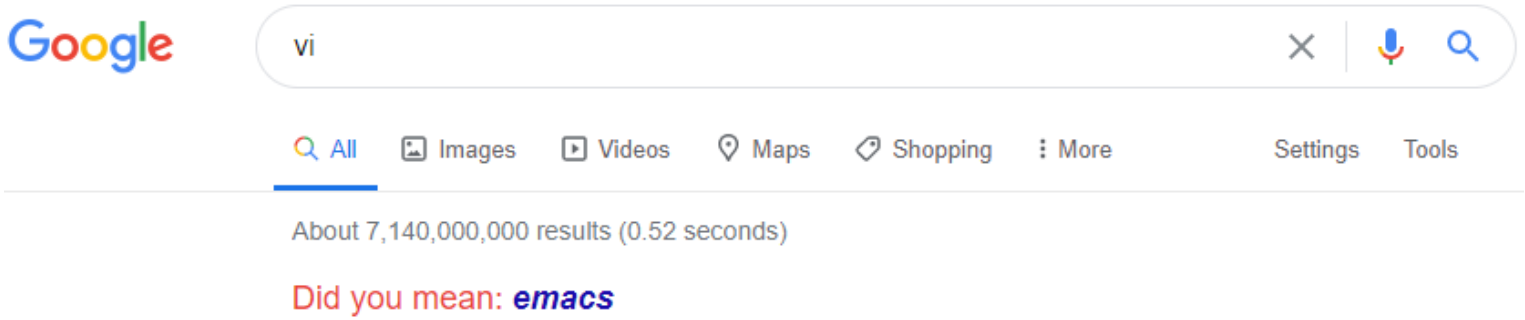
- History:
  - Pfeiltasten
  - `history`
  - `cat ~/.bash_history`
  - Strg + R
- Autovervollständigung: Tab

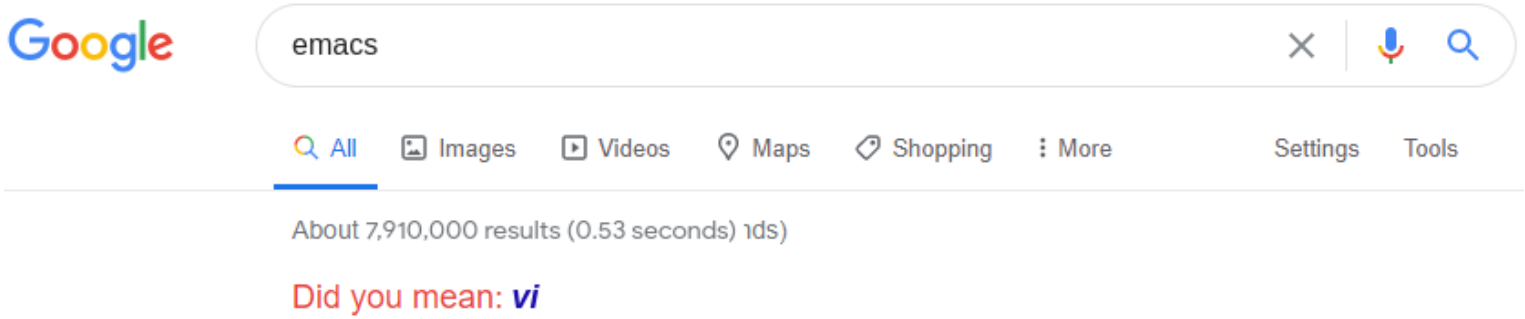
Dateien

- Erstellen: `touch foo.txt`
- Lesen: `cat foo.txt` oder `less foo.txt`
- Schreiben:
  - > überschreibt die Datei
  - >> hängt hinten an die Datei an.
- Filtern: `grep suchtext foo.text`
- Kopieren: `cp quelle.txt ziel.txt`
- Umbenennen / Verschieben: `mv vorher.txt nachher.txt`
- Löschen: `rm foo.txt`

Texteditoren

- `nano`
- `micro`
- `vi / vim / neovim`
- `emacs`





Ordner

- Erstellen: `mkdir foo`
- Inhalt auflisten: `ls`
- Kopieren: `cp -r quelle ziel`
- Umbenennen / Verschieben: `mv vorher nachher`
- Löschen: `rm -r foo`

Dateien und Ordner suchen

- Dateien suchen:

```
find -name foo.txt
find -name "*.txt*
find -type d -name bar
```

- Nach bestimmten Texten in Dateien suchen:

```
grep -Hirn 'suchtext'
```

`ls` im Detail

```
$ ls -lahd .bashrc .local
-rw-r--r-- 1 ruben ruben 3.5K Apr  4  2022 .bashrc
drwx------ 5 ruben ruben 4.0K Oct 19 13:39 .local
```

a b c d e f g

a) Berechtigungen
b) Anzahl Sub-Einträge
c) Besitzer
d) Besitzer-Gruppe
e) Größe
f) Zeit der letzten Änderung
g) Name

Berechtigungen

```
-rw-r--r--  1 ruben ruben 3
drwx------  5 ruben ruben 4
```

a b c d

a) Ordner?
b) Besitzer (u)
c) Besitzer-Gruppe (g)
d) Alle anderen (o)

d:	Directory
r:	Read
w:	Write
x:	Execute

Berechtigungen und Besitzer ändern

- Datei für alle lesbar machen:
  `chmod a+r foo.txt`
- Datei ausführbar machen:
  `chmod +x foo.txt`
- Besitzer auf „hauke“ ändern:
  `chown hauke:haukesgruppe foo.txt`

SSH

- Mit SSH können wir uns auf entfernten Rechnern einloggen:
- `ssh 4felgenh@rzssh1.informatik.uni-hamburg.de`
- `https://mafiasi.de/SSH`

Dateien herunterladen

- `curl example.org`
- `wget example.org`

Shell-Skripte

`bash` ist eine vollwertige Programmiersprache mit

- Variablen
- Parametern
- Verzweigung
- Schleifen
- Funktionen
- Datenstrukturen
  - Arrays
  - Maps

Mein erstes Shell-Skript

```
#!/usr/bin/env bash

echo Hallo Welt
```

- Skript ausführbar machen:
  `chmod +x meinskript.sh`
- Skript ausführen:
  `./meinskript.sh`

Variablen

```
#!/usr/bin/env bash

wer=Welt

echo Hallo $wer

wer=mafia

echo Hallo $wer
```

Parameter

```
#!/usr/bin/env bash

wer=$1

echo Hallo $wer
```

Verzweigung

```
#!/usr/bin/env bash

wer=$1

echo Hallo $wer

if [ "$wer" == "mafia" ] ; then
    food=Pizza
else
    food=Pasta
fi

echo Hier ist dein Teller $food
```

Schleifen

```
#!/usr/bin/env bash

echo Ich aß heute 5 Kekse. Erst den 1. Keks...

for i in {2..5} ; do
    echo Und dann den $i. Keks...
done
```

Funktionen

```
#!/usr/bin/env bash

function hallo (
    wer=$1
    echo Hallo $wer
)

bye() (
    wer=$1
    echo Bye $wer
)

hallo $1
bye $1
```

Anführungszeichen und Klammern

- Strings:
  ```
  name='Meine Textdatei.txt'
  ```
- Erhalten von Whitespace:
  ```
  cat "$name"
  ```
- Command Substitution:
  ```
  content="$(cat "$name")"
  content="`cat "$name"`"
  ```
- Compound Statements:
  ```
  {echo foo ; echo bar}
  (echo foo ; echo bar)
  ```
- Test Brackets:
  ```
  if [[ "$variable" == "foo" ]] ; then ...
  if [ "$variable" == "foo" ] ; then ...
  ```
- Arithmetik-Klammern:
  ```
  echo $(( 5 + 5 ))
  ```
- Array Builder:
  ```
  echo {01..05}
  ```

Tipps und Tricks

- https://www.shellcheck.net/
- Unofficial Bash Strict Mode: http://redsymbol.net/articles/unofficial-bash-strict-mode/
- https://raw.githubusercontent.com/felsenhower/file-templates/master/source/bash_script.sh

```
#!/usr/bin/env bash

set -euo pipefail

echo 'Hello World!'
```

Nützliche Programme

`awk`	Programmiersprache zur Textmanipulation
`sed`	Muster in Texten ersetzen
`grep`	Muster in Texten finden
`head`	Nur die ersten `n` Zeilen ausgeben
`tail`	Nur die letzten `n` Zeilen ausgeben
`wc`	Zeichen, Wörter oder Zeilen zählen
`sort`	Sortiere die Ausgabe
`read`	Lies Eingaben von `stdin` in Variablen
`htop`	Grafischer Taskmanager
`kill`	Prozesse beenden
`sleep`	Schlafe `n` Sekunden
`date`	Gib Datum und Zeit aus
`file`	Gibt den Dateitypen einer Datei aus
`lpr`	Zum Drucken, sehr nützlich auf `rzssh1`

Fingerübungen

- [Easy] Schreibe ein Shell-Skript, das alle Textdateien in einem Ordner auflistet. Der Zielordner soll der erste Parameter des Skriptes sein.
- [Medium] Gib zusätzlich die Gesamtgröße dieser Textdateien in Byte aus.
- [Advanced] Betrachte nur Textdateien, die vor weniger als 1 Tag (86.400 Sekunden) geändert wurden.
  - Tipp: UNIX Timestamps