HowTo: Linux

Liviana Franke

5. November 2023

Terminal Historie

- Im 19. Jahrhundert für Telegramme
  - Teleprinter
  - Teletype(writer)
  - Fernschreiber
- TTY – **T**ele**ty**pe als Ein-/Ausgabe Gerät für Computer
- Bis 1970er: Dumb Terminals
- Ab Mitte 1960er: Smart Terminals
- Ab Mitte 1990er: Virtuelle/Pseudo Terminals
  - **P**seudo-Tele**ty**pe – PTY
  - Terminal-Emulatoren (Software, z.B. konsole, xterm, . . .) verbinden sich mit PTY

Historie

Abbildung 1: Teletype Corporation ASR-33 im Computer History Museum Californien aus dem Jahr 1963 mit Papierrolle als Ausgabe (unter CC BY-SA 3.0).

Historie

Abbildung 1: Terminal ADM-3A mit externem Keypad aus dem Jahr 1976 (unter CC BY-SA 3.0).

Historie

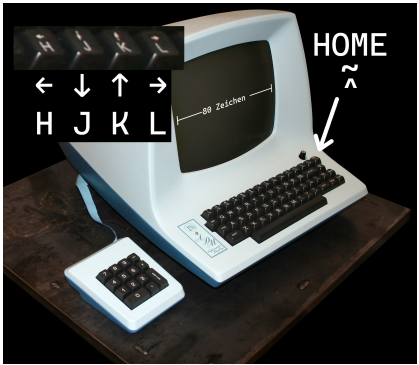

Abbildung 1: Terminal ADM-3A: Pfeiltasten auf H, J, K & L; Tilde auf Home; 80 Zeichen pro Zeile (unter CC BY-SA 3.0).

Begriffe

- Befehlszeile = Kommandozeile = Konsole = Commandline = CLI
- Shell
  - `sh` – früher *Bourne shell*, heute meist Symlink auf die System Shell
  - `bash` – *Bourne-again shell*
  - `zsh` – *Z shell*
  - `ksh` – *KornShell*
  - `csh` – *C shell* (seashell)
  - `dash` – *Debian Almquist shell*
  - `fish` – *friendly interactive shell*
- Terminal
- Terminal-Emulator, oft kurz Terminal

Befehle I

- Shell-Built-Ins
  `pwd`, `cd`, `echo`, `true`, `false`, `type`, ...
- Programme
  `firefox`, `touch`, `ls`, `cat`, `less`, `man`, ...
- Shell-Functions
  Abhängig von der Shell, i.d.R. in Form von Shell-Scripts

Befehle II

- Flags
  `firefox -help`
  `firefox -h`
- Parameter
  `type echo`
  `man echo`
- Argumente
  `ls -color=always`
  `tar -t -f out.tar.gz`

Ein- und Ausgabe

- Standardkanäle:
  - Standardeingabe: `stdin`
  - Standardausgabe: `stdout`
  - Standardfehlerausgabe: `stderr`
- Exit Codes (Exit Status)
- Um- und Weiterleiten von Ausgaben:
  - Redirect: `echo foobar > foobar.txt`
  - Redirect or Append: `echo foobar >> foobar.txt`
  - Pipe: `echo foobar | baz`
  - Variablen (Shell abhängig):

```
meinname=$(whoami)
echo "$meinname"
```

Chronik/History

- Pfeiltasten hoch/runter – direkte Navigation
- `history` – Shell-Built-In
- `less ~/.bash_history` – Bash spezifisch
- `Ctrl+R` – interaktive Suche (Shell abhängig)
- `tab` – Autovervollständigung (Shell abhängig)

Verzeichnisse

- Erstellen: `mkdir foo`
- Löschen (leer): `rmdir foo`
- Löschen (voll): `rm -r foo` (Vorsichtig!)
- Kopieren: `cp -r Quelle Ziel`
- Umbenennen/Verschieben: `mv Quelle Ziel`
- Inhalt auflisten: `ls foo`

Dateien I

- Erstellen: `touch foo.txt`
- Lesen: `cat foo.txt`, besser `less foo.txt`
- Schreiben:
  - `bar > foo.txt` erstellt oder überschreibt die Datei
  - `bar >> foo.txt` erstellt oder hängt an die Datei
- Löschen: `rm foo.txt`
- Kopieren: `cp Quelle.txt Ziel.txt`
- Umbenennen/Verschieben: `mv Quelle.txt Ziel.txt`

Dateien II

- Filtern:
  - Statisch: `grep RegExp foo.txt`
  - Interaktiv:
    1. `less foo.txt`
    2. Ampersand (&) gefolgt von RegExp tippen
    3. Mit Enter Filter anwenden
    4. Mit Ampersand (&) und Enter um Filter aufzuheben

Texteditoren I

- `nano`, `micro`
- `kilo`, `kibi`
- `emacs`
- `vi`, `vim`, `nvim`
- `kakoune`, `helix`

Texteditoren II

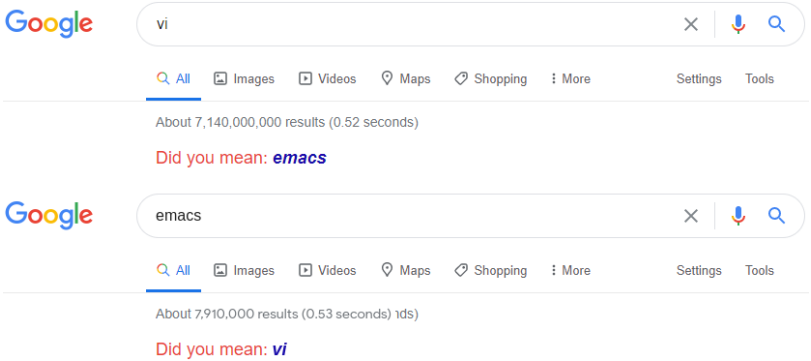

Abbildung 2: Are you okay, Google? – Ein ehemaliger Suchzyklus in der Google Suche.

Suchen

- Dateien suchen mit GNU `find`:
  `find -name foo.txt find /home/user/.config/ -type f -name '*bar'`
- Dateien suchen mit `fd`:
  `fd "string|regex"` –type d `fd --extension ".txt" /home/user/.config`
- Nach Datei mit bestimmten Inhalt suchen mit `grep`:
  `grep -Hirn 'suchtext' | less` (langsam bei vielen Dateien)
- Nach Dateien mit bestimmtem Inhalt suchen mit rip-grep:
  `rg 'suchtext' | less` (verdammt schnell, da schlaue Filter)

ls im Detail

```
$ ls -alhd .bash_history .local /bin
-rw------- 1 liv  liv  1.4K Sep 13 22:47 .bash_history
drwxr-xr-x 1 liv  liv    20 Jan  8  2023 .local
lrwxrwxrwx 1 root root    7 Sep 18 15:18 /bin -> usr/bin
========== = ==== ==== ==== ============ ===============
    a      b  c    d    e        f              g
```

a) Filetypeflag und Berechtigungen
b) Anzahl der Subeinträge
c) BesitzerIn
d) Gruppe von BesitzerIn
e) Größe (an Blöcken auf dem Dateisystem $\neq$ rekursive Größe)
f) Zeit der letzten Änderung (nicht rekursiv)
g) Dateiname

Berechtigungen I

```
- rw- --- ---
d rwx r-x r-x
l rwx rwx rwx
= === === ===
a  b   c   d
```

a) Dateityp-Flag:
   `-`: Reguläre Datei
   `d`: Verzeichnis
   `l`: Symlink
   `[bcCDMnpPs?]`: Andere Spezialtypen
b) Berechtigungen von BesizterIn
   `r`: Leserechte (Oktett: 4)
   `w`: Schreibrechte (Oktett: 2)
   `x`: Ausführrechte (Oktett: 1)
c) Berechtigungen der Gruppe
d) Berechtigungen aller anderen

Berechtigungen II

- Datei lesbar machen für...
  ...BesitzerIn: `chmod u+r foo.txt`
  ...Gruppe: `chmod g+r foo.txt`
  ...andere: `chmod o+r foo.txt`
  ...alle: `chmod a+r foo.txt`
  Analog für schreibbar (`+w`) bzw. ausführbar (`+x`)
- Datei für Gruppe nicht mehr lesbar machen:
  `chmod g-r foo.txt`
- BesitzerIn alle Rechte, Gruppe nur Leserechte, Anderen keine Rechte geben:
  `chmod u+rwx,g+r-wx,o-rwx foo.txt`
  `chmod 0740 foo.txt`

Berechtigungen III

- BesitzerIn auf „Foo“ ändern:
  `chown Foo foo.txt`
- Gruppe auf „Bar“ ändern:
  `chown :Bar foo.txt`
  `chgrp Bar foo.txt`
- BesitzerIn auf „Foo“ und Gruppe auf „Bar“ ändern:
  `chown Foo:Bar foo.txt`

SSH

- Secure Shell
- Via Netzwerk auf entfernte Rechner einloggen
- Öffnet Shell des anderen Rechners
- `ssh "<informatik-kennung>@rzssh1.informatik.uni-hamburg.de"`
- s. Wiki Eintrag: https://mafiasi.de/SSH

Networking Befehle

- Datei via HTTP/HTTPS runterladen:
  `curl "http://mafiasi.de"` `curl "https://mafiasi.de"`
  `wget "http://mafiasi.de"`
- Wörterbuch-Einträge:
  `curl "dict://dict.org/define:mafia"`
- Manuelle TCP Verbindung aufbauen:
  `nc "134.100.14.61" 80`

Shell-Skripte

- Shells unterscheiden sich i.d.R. in ihrer Syntax
- Im Folgenden Beispiele anhand `bash` und `fish`
- Beide sind vollwertige Programmiersprachen mit
  - Variablen
  - Parametern
  - Verzweigungen
  - Schleifen
  - Funktionen
  - Datenstrukturen
    - Arrays
    - Maps (Bash: ab v4.0, fish: in Arbeit)

Ein erstes Shell-Skript

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

echo "Hallo Welt!"
```

- Ausführbar machen:
  `chmod u+x skript.sh`
- Ausführen:
  `./skript.sh`

```
#!/usr/bin/env fish

echo "Hallo Welt!"
```

- Ausführbar machen:
  `chmod u+x skript.fish`
- Ausführen:
  `./skript.fish`

Variablen

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

wer="Welt"
echo "Hallo $wer!"
wer="Mafia"
echo "Hallo $wer!"

# Output von ./skript.sh
# > Hallo Welt!
# > Hallo Mafia!
```

```fish
#!/usr/bin/env fish

set wer "Welt"
echo "Hallo $wer!"
set wer "Mafia"
echo "Hallo $wer!"

# Output von ./skript.fish
# > Hallo Welt!
# > Hallo Mafia!
```

Parameter

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

wer1="$1"
wer2="$2"
echo "Hallo $wer1 und $wer2!"

# Output von ./skript.sh Mafia Welt
# > Hallo Mafia und Welt!
```

```fish
#!/usr/bin/env fish

set wer1 "$argv[1]"
set wer2 "$argv[2]"
echo "Hallo $wer1 und $wer2!"

# Output von ./skript.fish Mafia Welt
# > Hallo Mafia und Welt!
```

Verzweigung

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

wer="$1"
echo "Hallo $wer,"

if [ "$wer" == "Mafia" ]; then
  essen=Pizza
else
  essen=Pasta
fi

echo "Hier ist dein Teller $essen."
```

```fish
#!/usr/bin/env fish

set wer "$argv[1]"
echo "Hallo $wer,"

if [ "$wer" = "Mafia" ]
  set essen "Pizza"
else
  set essen "Pasta"
end

echo "Hier ist dein Teller $essen."
```

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
```

```
#!/usr/bin/env fish

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

for i in (seq 2 5)
  echo "Und dann den $i. Keks..."
end
```

Funktionen

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

function hallo (
  wer="$1"
  echo "Hallo $wer"
)
bye() (
  wer="$1"
  echo "Bye $wer"
)

hallo "$1"
bye "$1"
```

```fish
#!/usr/bin/env fish

function hallo
  set wer "$argv[1]"
  echo "Hallo $wer"
end
function bye
  set wer "$argv[1]"
  echo "Bye $wer"
end

hallo "$1"
bye "$1"
```

Anführungszeichen und Klammern

- Strings ohne Escape-Sequenzen: `'single quotes'`
- Strings mit Escape-Sequenzen: `"double quotes"`
- Kommentare: Zeilen beginnend mit #
- Befehle in einem Block abtrennen mittels Semikolon oder Newline

- Command Substitution:
  ```
  content="$(cat "$name")"
  content="`cat "$name"`
  ```
- Arithmetik:
  ```
  echo $((5 + 5))
  ```
- Array Builder:
  ```
  echo {01..05}
  ```

- Command Substitution:
  ```
  set content "$(cat "$name")"
  set content (cat "$name")
  ```
  (Newlines werden zu Spaces!)
- Arithmetik:
  ```
  math 5 + 5
  ```
- Array Builder:
  ```
  seq -w 01 05
  ```

Tipps und Tricks

- Shell Checker für sh, bash, dash und ksh:
  https://www.shellcheck.net (für sh/bash/dash/ksh)
- Inoffizieller Bash Strict Mode:
  http://redsymbol.net/articles/unofficial-bash-strict-mode/
  - `set -e`
    Beendet Skript sofort nach einem Fehler
  - `set -u`
    Zugriffe auf undefinierte Variablen erzeugen Fehler
  - `set -o pipefail`
    Verhindert verdecken von Error Codes

Nützliche Programme I

`grep`	Muster in Texten finden
`sed`	Muster in Texten ersetzen
`awk`	Programmiersprache zur Textmanipulation
`head`	Die ersten *n* Zeilen oder Bytes einer Datei ausgeben
`tail`	Die letzten *n* Zeilen oder Bytes einer Datei ausgeben
`wc`	Zeichen, Wörter oder Zeilen zählen
`sort`	Ausgabe sortien
`read`	Eingaben von `stdin` in Variablen lesen
`htop`	Textgraphischer Taskmanager
`bpytop`	Textgraphgischer Taskmanager und mehr
`man`	Manual Pages für C-Funktionen, Shell-Built-Ins, ...
`tldr`	TL;DR für Man Pages mit Beispielen

Nützliche Programme II

`kill`	Signale (auch Kill-Signale) an Prozesse senden
`killall`	Signale (auch Kill-Signale) an alle Prozesse eines Programmes senden
`sleep`	Prozess/Shell für *n* Sekunden schlafen legen
`watch`	Befehl alle *n* Sekunden wiederholen
`time`	Befehl timen
`date`	Datum und Uhrzeit formatiert ausgeben
`file`	Datentyp einer Datei ausgeben
`lpr`	Dateien drucken (nützlich auf `rzssh1`)
`pdfunite`	Mehrere PDFs zusammenfügen
`ln -s`	Symlinks erstellen
`qalc`	Qalculate Terminal Taschenrechner mit Einheiten und mehr
`dos2unix`	Dateien mit CRLF zu LF konvertieren
`eza`	`ls` mit Extras

Nützliche Programme III

`bat`	`cat` und `less` mit Extras
`git`	Versionskontrolle und -verwaltung
`lsblk`	Blockdevices (Speichergeräte) auflisten
`lsusb`	USB Geräte auflisten
`tmux`	Terminal Multiplexer mit vielen Features
`xdotool`	Tastendrücke und Mausevents emulieren
...	...