Die Kommandozeile

Wie funktioniert das und was kann man damit alles machen?

Achtung!

Die Kommandozeile ist ein mächtiges Werkzeug. Es ist sehr leicht, damit sein System irreparabel zu beschädigen oder einen Verlust von Daten zu erleiden; insbesondere (aber nicht nur!) wenn man als Superuser ("Root") auf der Kommandozeile arbeitet.

Diese Folien dienen zur reinen Information und stellen keinerlei Aufforderung zu irgendwelchen Aktionen dar.

Der Gebrauch der hier angebotenen Informationen, Beispiele und Konzepte geschieht ausschließlich auf eigene Gefahr.

- Ganz am Anfang: Telegraphie als das "Viktorianische Internet"
- Morsecode als erstes "Online-Protokoll", Morsestationen als die ersten "Proto"-Terminals.
- Weiterentwicklung der Morse-Telegraphie: **Fernschreiber**
  (Engl.: *TeleTYpeWriter* -> `tty`)
- Fernschreiber: Einsatz als frühe Computer-Terminals, aber nur suboptimal für Interaktion mit IT:
  - Papierverbrauch, Ausdruck meistens unnötig
  - Zu langsam
  - Keine komplexere Interaktion (editieren etc.)

- Ab den 1970ern: Aufkommen der Computerterminals
- 1980er bis heute: GUIs incl. neuer Interaktionsformen (Maus, Touchscreen, Spracherkennung, ...)

Warum soll ich mir die Mühe mit der Kommandozeile machen?
Ist das nicht ein Relikt aus der Computer-Steinzeit?
Ich habe doch meinen schicken Desktop ... ?

- Erheblich "näher" am System (GUIs verbergen Interna als 'Black Box')
- Bessere Kontrolle über den Rechner (einer **der** Gründe für Linux)
- Schnellere Bedienung (mit entsprechender Übung)

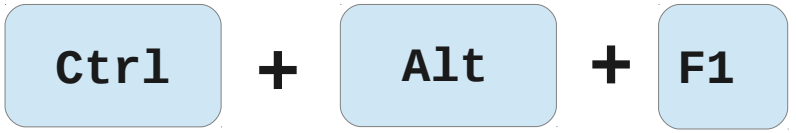
  -> Kurze & knackige Kommandos auch für komplexe Aktionen

  -> Kein Wechsel Maus - Tastatur
- Netzzugriff (remote login) auf entfernten Rechner
- Skripte zur Verarbeitung grosser Datenmengen
- Im Notfall (Systemreparatur) ist evtl. keine GUI mehr vorhanden
- Erheblich geringerer Ressourcenverbrauch als GUIs
  (wichtig z.B. für Raspberry Pi u.ä.)

Zuerst brauchen wir ein Terminal. Wie kriegen wir eines?

a) Einen **Terminal Emulator** ausführen (ist bei jedem Linux Desktop mit dabei)

b) oder: Die GUI verlassen und die **Linux Console** aufrufen mit:



c) oder: Sich auf einem (entfernten) Rechner einloggen mit Programmen wie **PuTTY**

xterm

Gnome Terminal

KDE-Terminal

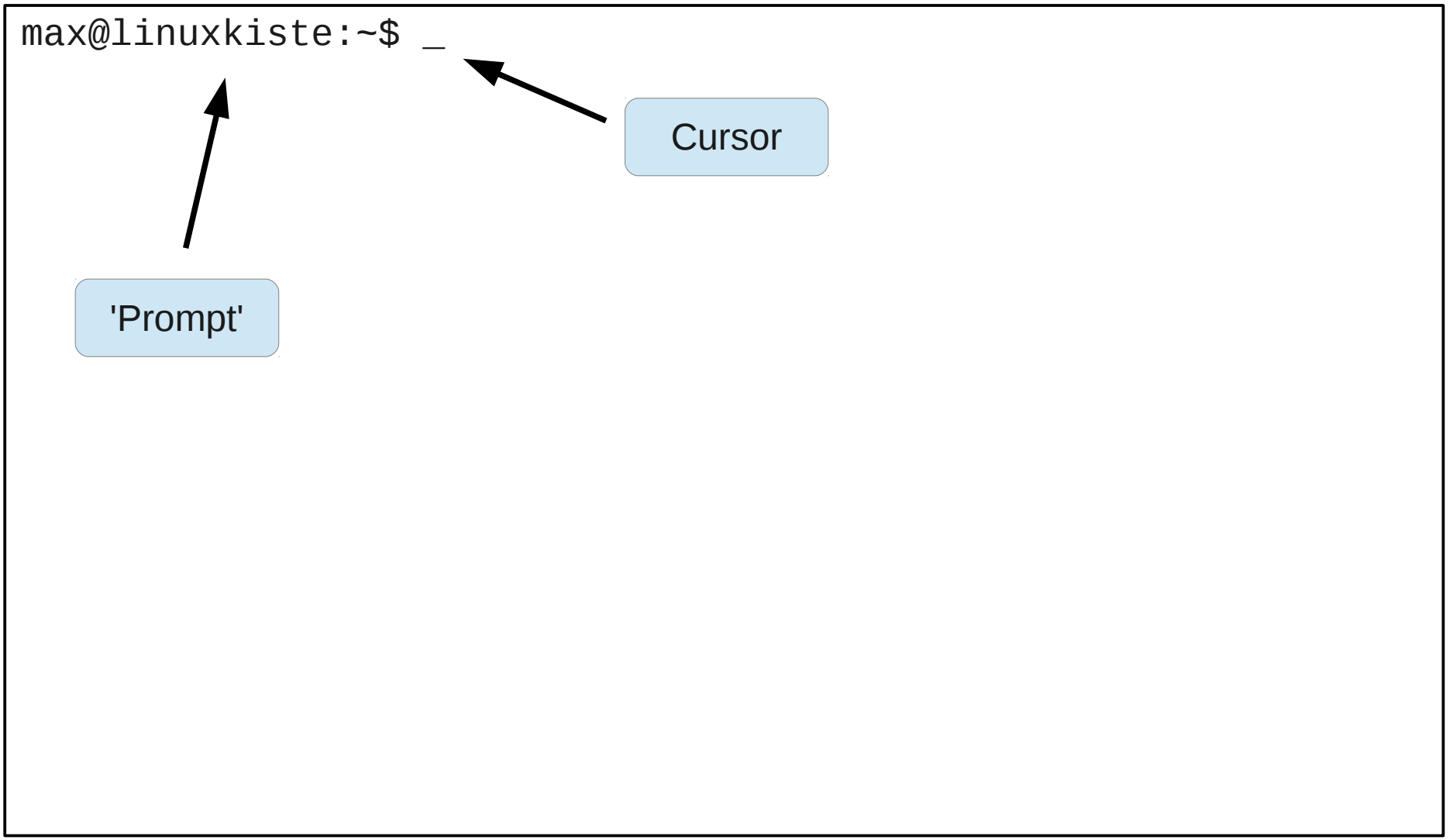
bash

csh

ash

ksh

zsh


max@linuxkiste:~$ _
Cursor
'Prompt'

```
max@linuxkiste:~$ gimp
```

```
max@linuxkiste:~$ gimp urlaubsphoto_sonnenuntergang.jpg
(GIMP startet und öffnet gleich das Urlaubsphoto)

max@linuxkiste:~$ gimp -v
GNU Image Manipulation Program Version 2.8.2
git-describe: GIMP_2_8_0-194-ga42a02e

verwendet GEGL Version 0.2.0 (gebaut gegen Version 0.2.0)
verwendet GLib Version 2.32.4 (gebaut gegen Version 2.32.4)
verwendet GdkPixbuf Version 2.26.1 (gebaut gegen Version
2.26.1)
verwendet GTK+ Version 2.24.10 (gebaut gegen Version 2.24.10)
verwendet Pango Version 1.30.0 (gebaut gegen Version 1.30.0)
verwendet Fontconfig Version 2.9.0 (gebaut gegen Version 2.9.0)
verwendet Cairo Version 1.12.2 (gebaut gegen Version 1.12.2)

max@linuxkiste:~$ _
```

Grundlegende Struktur eines Kommandos:

```
<Kommando> <Parameter1> <Parameter2> <...>
```

Übliche Parameterformen:

- **Argumente**: Dateinamen, Suchmuster etc.

  ```
  ~$ gimp sonnenuntergang.jpg
  ```

- **Single Options**: Ein Minus '-' und ein Zeichen

  ```
  ~$ ls -l -r -t -h
  ```

- **Combined single options**: Ein Minus '-' und mehrere Zeichen

  ```
  ~$ ls -lrth
  ```

- **Long options**: Zwei Minus '-' und eine Beschreibung

  ```
  ~$ gimp --license
  ```

Kurzüberblick über Ausruf und Optionen eines Kommandos häufig mit **-h**
(häufig, aber leider nicht immer)

```
max@linuxkiste:~$ gimp -h
Aufruf:
  gimp [OPTION …] [DATEI|URI...]

GNU Image Manipulation Program

Hilfeoptionen:
  -h, --help            Hilfeoptionen anzeigen
  --help-all            Alle Hilfeoptionen anzeigen
  --help-gegl           Show GEGL Options
  --help-gtk            GTK+-Optionen anzeigen

Anwendungsoptionen:
  -v, --version         Versionsinformationen anzeigen
  --license             Lizenzinformationen anzeigen
  --verbose             Ausführlicher verhalten
 ...
```

Ausführlicher: Optionen und Parameter über die **Man-Page** ("Manual Page")

```
max@linuxkiste:~$ man gimp
```

```
GIMP(1)                  GIMP Manual Pages                  GIMP(1)

NAME
    gimp - an image manipulation and paint program.

SYNOPSIS
    gimp   [-h] [--help] [--help-all] [--help-gtk] [-v]
    [-version] [--license]  [--verbose]  [-n]  [--new-instance]
    [-a] [--as-new] [-i] [--no-interface] [-d] [--no-data] [-f]
    [--no-fonts] [-s] [--no-splash] [--no-shm] [--no-cpu-accel]
    [--display display] [--session <name>] [-g]
    [--gimprc  <gimprc>] [--system-gimprc <gimprc>]
    [--dump-gimprc] [--console-messages]    [--debug-handlers]
    [--stack-trace-mode <mode>] [--pdb-compat-mode <mode>]
    [--batch-interpreter  <procedure>]  [-b]
    [--batch <command>] [filename] ...

DESCRIPTION
    GIMP is the GNU Image Manipulation Program. It is used to
    edit and manipulate images. It can load and save a variety
    of image formats and can be used to convert between
    formats. GIMP can also be used as a paint program.
...
```

Die Shell hat einige fest eingebaute (interne) Kommandos, für die keine Man-Page vorhanden ist. Hier kann **`help`** weiterhelfen:

```
max@linuxkiste:~$ man cd
Kein Handbucheintrag für cd vorhanden
max@linuxkiste:~$ help cd
cd: cd [-L|[-P [-e]]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is
    the value of the HOME shell variable.
...
```

Aufgepasst: Ein Kommando blockiert die Konsole solange bis es beendet ist.

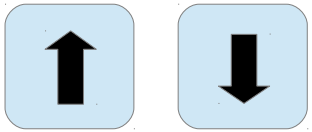
```
max@linuxkiste:~$ gimp
(Konsole nimmt keine Kommandos an)
```

Lösung: Programm im Hintergrund starten (mit '**&**'-Zeichen am Ende)

```
max@linuxkiste:~$ gimp &
[1] 8169
max@linuxkiste:~$ _
```

Erzwungenes Beenden von Programmen im Vordergrund der Konsole mittels:

(funktioniert häufig, aber leider nicht immer ...)



Durch vorherige Kommandos scrollen

Textausgabe durchscrollen

Textausgabe durchscrollen (Linux Konsole)

Autocompletion

Prozess im Vordergrund abbrechen

Das sog. *aktuelle Verzeichnis* wird (meist) im Prompt angezeigt:

```
max@linuxkiste:/usr/local/share$ _
```

Die Tilde **'~'** bedeutet dabei das Home-Verzeichnis des Benutzers

```
max@linuxkiste:~$ _
```

```
max@linuxkiste:~/Dokumente/Post$ _
```

pwd zeigt das aktuelle Verzeichnis an ('**P**rint **W**orking **D**irectory'):

```
max@linuxkiste:~/Dokumente/Post$ pwd
/home/max/Dokumente/Post
```

cd wechselt das aktuelle Verzeichnis (Tipp: Autocompletion per <tab> sehr nützlich)

Relative Pfade:

```
max@linuxkiste:~/Dokumente$ cd Post
max@linuxkiste:~/Dokumente/Post$ _
```

Ein Verzeichnis zurück:

```
max@linuxkiste:~/Dokumente/Post$ cd ..
max@linuxkiste:~/Dokumente/$ _
```

Absolute Pfade:

```
max@linuxkiste:~/Dokumente/Post$ cd /usr/share
max@linuxkiste:/usr/share$ _
```

Home-Verzeichnis:

```
max@linuxkiste:/usr/share$ cd ~/Dokumente
max@linuxkiste:~/Dokumente$ _
```

ls zeigt Inhalt eines Verzeichnisses an:

```
max@linuxkiste:~/Dokumente$ ls
Bücher  Job  Post  Rezepte  howto.txt  kuendigung.pdf
sonnenuntergang.jpg
max@linuxkiste:~/Dokumente$ ls Rezepte/
cocktails.txt  curry_huehnchen.txt  spargel.txt
```

ls hat enorm viele Optionen. Man sollte kennen:

```
-l        Ausgabe als Liste
-a        Ausgabe versteckter Dateien (Name beginnt mit '.')
-h        (mit -l) Größe als 'human-readable' ausgeben
-t        Sortieren nach Änderungszeit
-X        Sortieren nach Dateiendung
--color   Farbige Ausgabe
```

ls zeigt Inhalt des aktuellen Verzeichnisses an (Listenformat):

```
max@linuxkiste:~/Dokumente$ ls -laht --color
drwxr-xr-x 13 max users 4,0K Mai 26 15:10 .
drwxr-xr-x 63 max users 4,0K Mai 26 11:44 ..
-rw-r--r--  1 max users 1,3M Apr 26 11:05 kuendigung.pdf
-rw-r--r--  1 max users  91K Mär 10 17:43 howto.txt
-rw-r--r--  1 max users 984K Mär 10 15:52 sonnenuntergang.jpg
-rw-------  1 max users  342 Feb 05 16:00 .hidden
drwxr-xr-x 12 max users 4,0K Feb 05 15:00 Bücher
drwxr-xr-x  8 max users 4,0K Feb 05 15:00 Job
drwxr-xr-x  4 max users 4,0K Feb 05 15:00 Post
drwxr-xr-x  5 max users 4,0K Feb 05 15:00 Rezepte
```

cp kopiert Dateien und Verzeichnisse (Autocompletion auch hier sehr hilfreich)
mv verschiebt Dateien und Verzeichnisse.

```
max@linuxkiste:~/Dokumente$ ls Post/
bestellung.pdf  storno.pdf
max@linuxkiste:~/Dokumente$ cp kuendigung.pdf Post/
max@linuxkiste:~/Dokumente$ ls Post/
bestellung.pdf  kuendigung.pdf  storno.pdf
max@linuxkiste:~/Dokumente$ mv Post/storno.pdf .
max@linuxkiste:~/Dokumente$ ls Post/
bestellung.pdf  kuendigung.pdf
```

Die wichtigsten Optionen:

```
-i        Vor Überschreiben nachfragen
-n        Existierende Dateien nicht überschreiben
-u        Nur kopieren wenn Datei nicht existiert oder neuer
-r        Verzeichnisse rekursiv kopieren
```

Bedeutung von `'.'` und `'..'`

`'.'` und `'..'` sind automatisch in jedem Verzeichnis vorhanden

`.` bezeichnet das Verzeichnis selbst
`..` bezeichnet das Verzeichnis eine Ebene höher

Kopieren einer Datei ins aktuelle Verzeichnis:

```
max@linuxkiste:~/Dokumente$ cp Post/kuendigung.pdf .
```

Wechseln ins Verzeichnis höher:

```
max@linuxkiste:~/Dokumente$ cd ..
max@linuxkiste:~$ _
```

rm löscht Dateien und Verzeichnisse

```
max@linuxkiste:~/Dokumente$ ls Post/
bestellung.pdf  kuendigung.pdf
max@linuxkiste:~/Dokumente$ rm Post/kuendigung.pdf
max@linuxkiste:~/Dokumente$ ls Post/
bestellung.pdf
```

Die wichtigsten Optionen:

```
-f        Löschen ohne Nachfragen (Vorsicht!)
-i        Vor Löschen nachfragen
-r        Verzeichnisse mit Inhalt rekursiv löschen
```

cp, mv, rm können mehrere Dateien auf einmal verarbeiten:

```
max@linuxkiste:~/Dokumente/Post$ cp bestellung.pdf
kuendigung.pdf storno.pdf ~/Backup/
```

Dateien mit Ähnlichkeiten im Dateinamen kann man per *Wildcard* zusammenfassen:

```
max@linuxkiste:~/Dokumente/Post$ cp *.pdf ~/Backup/
```

```
*           Beliebig viele Zeichen (auch keines)

?           Genau ein beliebiges Zeichen

[abc]       Ein Zeichen aus der Liste

[a-m]       Ein Zeichen aus dem Bereich

{pdf,txt}   Eine der angegebenen Zeichenketten
```

Zeige alle PDF-Dateien im Verzeichnis an:

```
max@linuxkiste:~$ ls *.pdf
```

Zeige alle Bilder mit Nummer 100-199 an:

```
max@linuxkiste:~$ ls Bild_1??.jpg
```

Zeige alle Dateien an, die mit a, b, c oder d anfangen:

```
max@linuxkiste:~$ ls [a-d]*
```

Zeige alle PDF- und TXT-Dateien im Verzeichnis an:

```
max@linuxkiste:~$ ls *.{pdf,txt}
```

Diverse Kommandos, die man auch kennen sollte:

```
mkdir     Neuen Ordner anlegen
touch     Neue Datei anlegen (eigentlich: Zeitstempel ändern)
df        Dateisystembelegung anzeigen
du        Speicherplatz von Dateien anzeigen
cat       Inhalt einer Datei anzeigen (alles auf einmal)
less      Inhalt einer Datei anzeigen (mit Scrollen)
find      Dateien finden
grep      Muster in Dateien finden
vi        Der klassische UNIX Texteditor
nano      Noch ein Texteditor
```

Aufgepasst: Die Dateikommandos löschen / überschreiben Dateien idR ohne nachzufragen!

Es gibt keinen "Papierkorb", aus dem man versehentlich gelöschte Dateien wiederherstellen kann!

Tippfehler können schwerwiegende Auswirkungen haben.

Beispiel: Geplant mit folgendem Kommando ist eigentlich, alle .o-Dateien zu löschen. Aus Versehen ist ein Leerzeichen reingekommen. Was passiert?

```
$ rm * .o
```

Im Zweifelsfall lieber die `-i` Option verwenden, mit der die Shell immer erst nachfragt, bevor Dateien gelöscht / überschrieben werden.

```
$ rm -i *.o
```

Für Optionen, die man häufig verwendet kann man mit **`alias`** eine Kurzform definieren:

Beispiel: ls als Liste und mit Farbhervorhebung

```
$ alias ll='ls -l --color=auto'
```

Sicherheitsmaßnahme, damit bei cp, mv, rm immer nachgefragt wird:

```
$ alias cp='cp -i'
$ alias mv='mv -i'
$ alias rm='rm -i'
```

Und damit man das dauerhaft hat kann man es in die `.bashrc` eintragen ... das ist aber Material für einen nächsten Vortrag.

uname	Systeminformationen anzeigen (Kernelversion etc.)
dmesg	Kernel Ring Buffer anzeigen
lspci	PCI-Bus Belegung anzeigen
lsusb	USB-Bus Belegung anzeigen
lsmod	Geladene Kernelmodule anzeigen
lsof	Geöffnete Dateien anzeigen
uptime	Anzeigen wie lange das System schon läuft
top	Monitor der laufenden Prozesse (fortlaufend)
ps	Laufende Prozesse anzeigen (als Liste)
kill	Prozess stoppen (eigentlich: Signal an Prozess senden)
fdisk	Partitionierung der Festplatte anzeigen / ändern
mount	Gerät in das Dateisystem einhängen

5. Was nicht behandelt wurde

- Shell Konfigurationsdateien
- Redirects und Pipes
- Den Prompt individuell konfigurieren
- Umgebungsvariablen
- Prozessverwaltung
- Command expansion
- Shell Scripting
- Programmpakete direkt aus Sourcen bauen
- Terminal capabilities
- TUIs ('Text User Interfaces' - 'Halbgraphik')