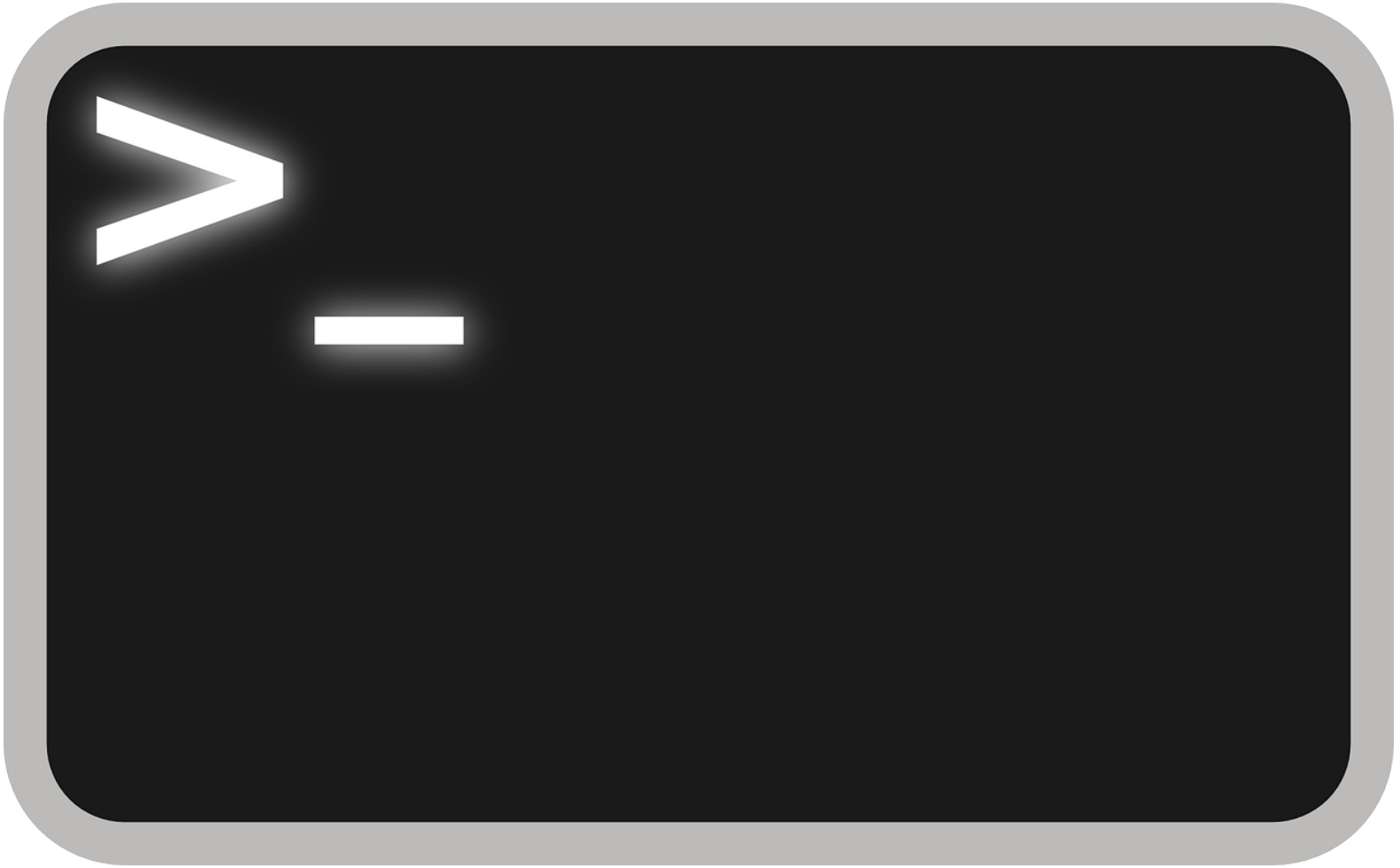
spline_

Freie Universität Berlin

Terminal & Scripte



Terminal

Was ist ein Terminal?:

- Eine Ein- & Ausgabeschnittstelle zwischen dem Benutzer und dem Rechner
- Das heißt: wir können den Rechner mit einer „nicht grafischen Oberfläche“ bedienen

spline

Freie Universität Berlin

Terminal

Bevor wir Anfangen:

- Das Standardterminal in Linux heißt ***tty*** (teletypewriter)
- Man kann es mit Ctrl+Alt+F*x* – Tastenkombination öffnen, wobei *x* eine Zahl ist (zum Beispiel: Ctrl+Alt+F3)
- Man sieht, dass es mehrere ttys gibt
- In der grafischen Sitzung gibt es Terminal-Emulatoren, zum Beispiel: Konsole, Xterm, Gnome-Terminal, Terminology usw.
- Im Terminal läuft eine ***Shell*** (Software, die den Benutzer mit dem Computer vebindet)
- Es gibt mehrere Shells, zum Beispiel: dash, bash, zsh, fish usw.
- Standartshell ist bash (die werden wir auch verwenden)

spline

Freie Universität Berlin

Verzeichnis-Hierarchie

Struktur:

/	Wurzelverzeichnis
/bin	grundlegende Befehle und Programme (binary)
/sbin	grundlegende Systembefehle (benötigen root-Rechte)
/lib	Kernel-Module und dyn. ladbare Bibliotheken
/usr	Bibliotheken, Systemtools, installierte Programme
/tmp	temporäre Daten (werden nach dem Neustart gelöscht)
/dev	am Computer angeschlossene Geräte
/boot	Bootloader & Kernelimage
/etc	Systemkonfigurationsdateien
/var	enthält nur Verzeichnisse ← werden durch Programme verändert
/home	Benutzerverzeichnis

spline_

Freie Universität Berlin

Terminalbefehle

Wo sind wir?:

- Wenn wir die Shell öffnen, dann sind wir standardmäßig in unserem Home-Verzeichnis: ***/home/<user>***
- Um den Pfad anzuzeigen, können wir ***pwd*** in die Shell eingeben und Enter drücken. Dieser Befehl liefert uns den absoluten Pfad des aktuellen Verzeichnisses

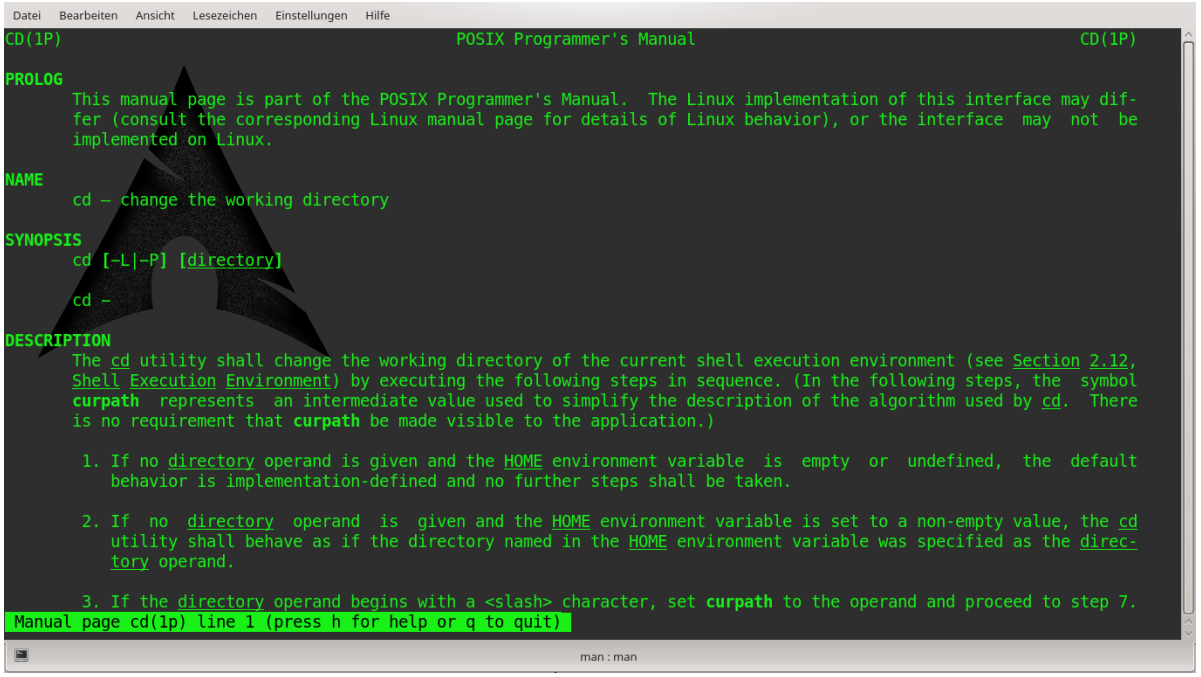
```
[gordon@arch ~]$ pwd
/home/gordon
```

Terminalbefehle

Manpage?:

- Mit der Manpage erhält man eine Anleitung zu den jeweils gesuchten Befehlen und Programmen
- Syntax: ***man <Option> <Befehl>***

```
[gordon@arch ~]$ man cd
```



spline

Freie Universität Berlin

Terminalbefehle

ls:

- Um zu sehen, welche Dateien und Ordner sich in einem Verzeichnis befinden, können wir den Befehl ***ls*** benutzen
- Syntax: ***ls <Option> <Pfad>***
- ***ls -a:*** Zeigt den gesamten Ordnerinhalt (versteckte Dateien auch)
- ***ls -l:*** Zeigt Details für Ordner und Dateien (Eigentümer, Rechte, Datum usw.)
- ***ls -s:*** Zeigt die Größe von Dateien und Ordnern
- ***ls -sh:*** Zeigt die Größe im menschenlesbaren Format
- Man kann auch alle Optionen gleichzeitig benutzen: ***ls -shla***

```
[gordon@arch ~]$ ls
 Bilder      Downloads   Schreibtisch  'VirtualBox VMs'   Öffentlich
 Dokumente   Musik       Videos         Vorlagen
```

spline

Freie Universität Berlin

Terminalbefehle

cd:

- Um in ein Verzeichnis zu wechseln, kann man den Befehl ***cd*** benutzen
- Dazu gibt man den Pfad oder einen Ordner im aktuellen Verzeichnis an
- ***cd*** ohne Parameter bringt euch zurürck zu eurem Benutzerverzeichnis
- Syntax: ***cd <Pfad/Ordner>***
  cd <Ordner>

```
[gordon@arch ~]$ cd Dokumente
[gordon@arch Dokumente]$
```

```
[gordon@arch Linuxkurs]$ cd /home/gordon/Dokumente/Linuxkurs
[gordon@arch Linuxkurs]$
```

spline_

Freie Universität Berlin

Terminalbefehle

Pfadaliase:

.	Aktuelles Arbeitsverzeichnis
..	Darüberliegendes Verzeichnis
~	Benutzerverzeichnis
-	Vorheriges Verzeichnis (nur mit ***cd*** Befehl)

spline

Freie Universität Berlin

Terminalbefehle

touch:

- Um eine Datei anzulegen, benutzt man den Befehl ***touch***
- Syntax: ***touch <Pfad/Dateiname>***
  touch <Dateiname>

```
[gordon@arch Test]$ touch test
[gordon@arch Test]$ ls
test
```

spline_

Freie Universität Berlin

Terminalbefehle

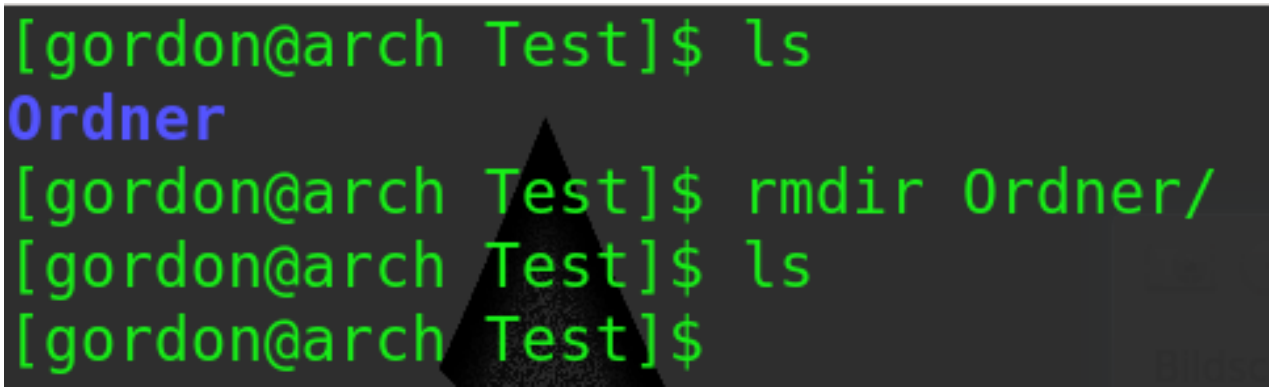
mkdir:

- Um einen Ordner anzulegen, benutzt man den Befehl ***mkdir***
- Syntax: ***mkdir <Pfad/Ordnername>***
  mkdir <Ordnername>

Terminalbefehle

rmdir:

- Um einen **leeren** Ordner zu löschen, benutzt man den Befehl ***rmdir***
- Syntax: ***rmdir <Pfad/Ordnername>***
  rmdir <Ordnername>



spline

Freie Universität Berlin

Terminalbefehle

rm, rm -r, rm -rf:

- Um eine Datei zu löschen, benutzt man den Befehl ***rm***
- Um einen Ordner zu löschen, muss man ***rm*** mit dem Parameter ***-r*** (rekursiv) benutzen. Um die Abfrage zu vermeiden, ob der Ordner gelöscht werden soll, kann man noch ***-f*** (force) verwenden
- Syntax:

  rm <Pfad>/Dateiname

  rm Dateiname

  rm -r <Pfad>/Ordnername

  rm -r Ordnername

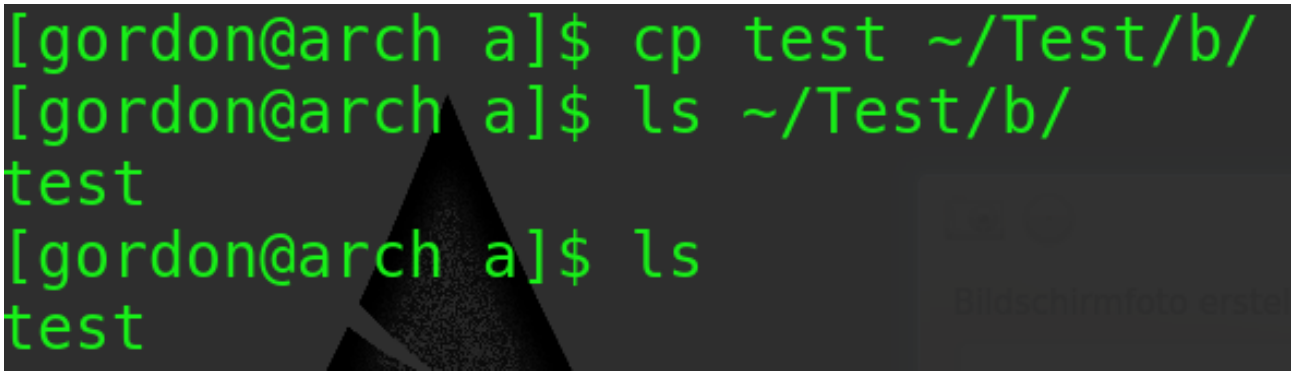
```
[gordon@arch Test]$ ls
test
[gordon@arch Test]$ rm test
[gordon@arch Test]$ ls
[gordon@arch Test]$
```

```
[gordon@arch Test]$ ls
Ordner
[gordon@arch Test]$ rm -rf Ordner
[gordon@arch Test]$ ls
[gordon@arch Test]$
```

Terminalbefehle

cp, cp -r:

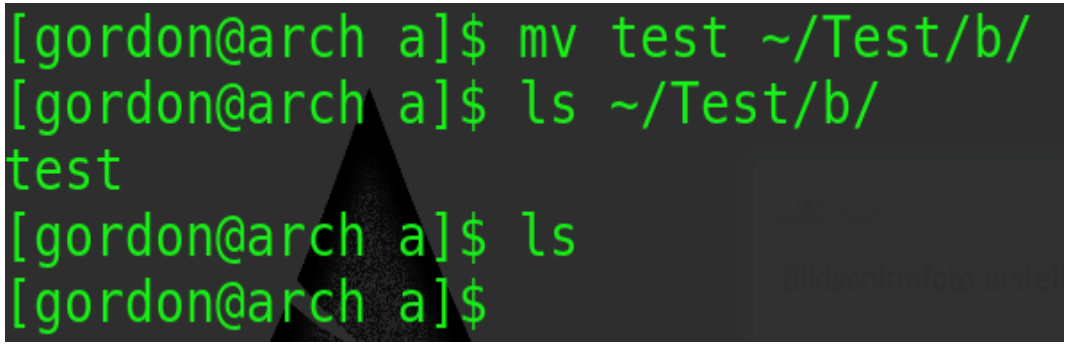
- Um eine Datei zu kopieren, benutzt man den Befehl ***cp***
- Syntax: ***cp <Pfad/Datei> <Zielpfad>***
- Um einen Ordner zu kopieren, muss man zusätzlich den Parameter ***-r*** (rekursiv) benutzen
- Syntax: ***cp -r <Pfad/Ordner> <Zielpfad>***

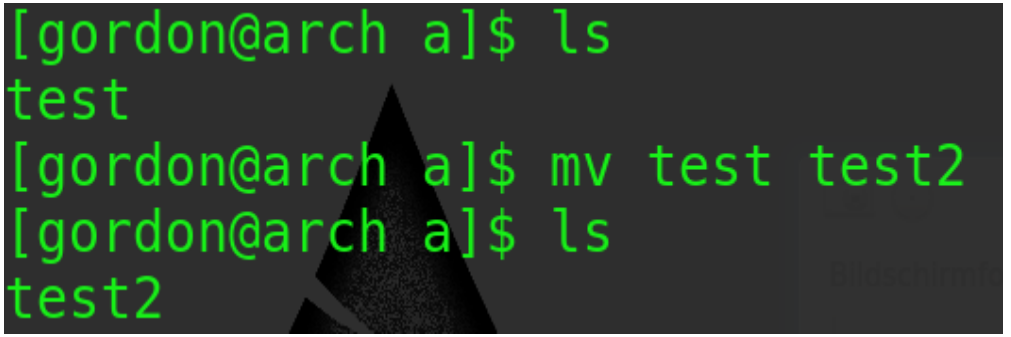


Terminalbefehle

mv:

- Um eine Datei oder Ordner zu verschieben und/oder umzubenennen, benutzt man den Befehl ***mv***
- Syntax: ***mv <Pfad/Datei> <Zielpfad/Datei>***
  mv <Pfad/Ordner> <Zielpfad/Ordner>

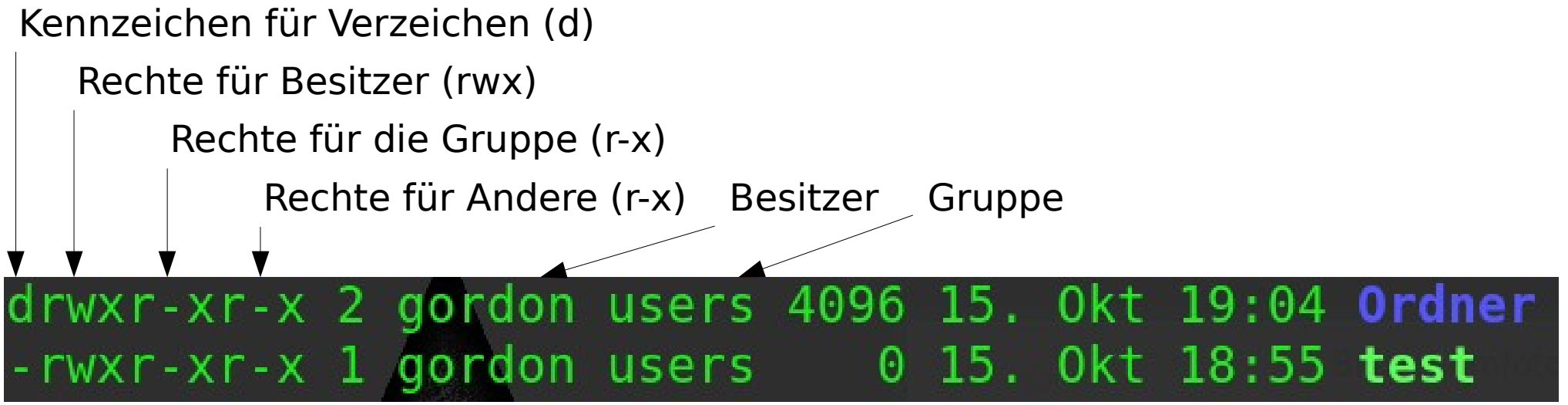




Terminalbefehle

Rechte und Besitzer:

- Bevor wir mit weiteren Befehlen fortfahren, werden wir uns die Details von der Datei und dem Ordner anschauen: ***ls -l***



spline_

Freie Universität Berlin

Terminalbefehle

chmod:

- Mit dem Befehl chmod können wir die Rechte für eine Datei oder einen Ordner vergeben oder entziehen
- Da gibt es 3 Arten (***r*** – lesen, ***w*** – schreiben, ***x*** - ausführbar)
- Außerdem müssen wir entscheiden, für welche Benutzer und Gruppe wir die Rechte vergeben oder entziehen
- ***u*** – Besitzer von der Datei/Ordner
- ***g*** - Gruppe der Datei
- ***o*** – Andere Benutzer
- ***a*** – Besitzer, Gruppe und Andere

spline_

Freie Universität Berlin

Terminalbefehle

chmod:

- Um die Rechte zu vergeben, benutzt man ***chmod +<Rechte>***
- Um die Rechte zu entziehen, benutzt man ***chmod -<Rechte>***
- Um für bestimmte Benutzer oder Gruppen die Rechte zu vergeben oder entziehen, kann man vor dem **+** oder **–** die entsprechenden Parameter angeben (siehe vorherige Folie)

```
[gordon@arch Test]$ ls -l
insgesamt 0
-rwxr-xr-x 1 gordon users 0 15. Okt 18:55 test
[gordon@arch Test]$ chmod a+rwx test
[gordon@arch Test]$ ls -l
insgesamt 0
-rwxrwxrwx 1 gordon users 0 15. Okt 18:55 test
```

spline_

Freie Universität Berlin

Terminalbefehle

chown:

- Um den Besitzer und die Gruppe der Datei oder Ordner zu verändern, kann man den Befehl ***chown*** benutzen
- Syntax:

  chown<Benutzer>:<Gruppe><Pfad/Dateiname>
  chown <Benutzer>:<Gruppe> <Dateiname>

```
[root@arch Test]# ls -l
insgesamt 0
-rwxrwxrwx 1 root root 0 15. Okt 18:55 test
[root@arch Test]# chown gordon:users test
[root@arch Test]# ls -l
insgesamt 0
-rwxrwxrwx 1 gordon users 0 15. Okt 18:55 test
```

spline_

Freie Universität Berlin

Terminalbefehle

echo:

- Um einen Text auf dem Bildschirm auszugeben, kann man den Befehl ***echo*** verwenden
- Syntax: ***echo <“Text“>***
- Man kann auch mit dem Befehl etwas in die Datei reinschreiben: ***echo [“Test“] > [Dateiname]*** (überschreibt test.txt)
- Wenn man etwas am Ende der Datei hinzufügen möchte (nächste Zeile), schreibt man: ***echo [“Test“] >> [Datei]***

```
[gordon@arch Test]$ echo "Hallo Welt"
Hallo Welt
```

spline

Freie Universität Berlin

Terminalbefehle

cat:

- Um den Inhalt einer Datei auszugeben, benutzt man den Befehl ***cat***
- Syntax: ***cat <Dateiname>***
- Man kann auch den Inhalt von einer Datei in die andere schreiben. Funktioniert wie bei ***echo:***

  cat [Datei1] > [Datei2]]

  cat [Datei1] >> [Datei2]]

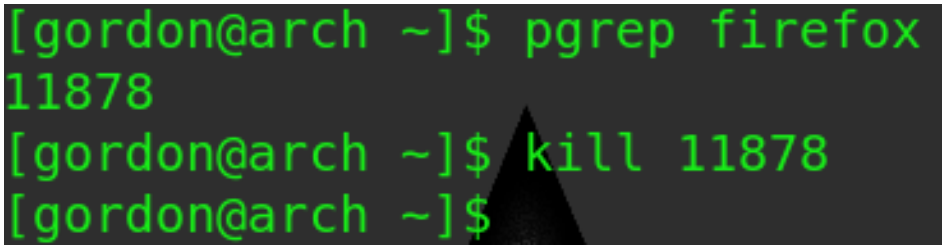
spline_

Freie Universität Berlin

Terminalbefehle

Prozesse:

- Um die ID von einem Prozess rauszufinden, kann man **pgrep** benutzen
- Man kann ein Programm mit dem Befehl **kill** beenden
- Das Programm kann als mehrere Prozesse laufen, um alle zu beenden kann man **pkill** verwenden. Da muss man **pgrep** nicht verwenden
- Syntax: **pgrep *<Pattern>***
  kill <ProzessID>
  pkill <Pattern>



spline_

Freie Universität Berlin

Terminalbefehle

sudo, su:

- Einige Befehle/Operationen darf man nur mit Administrator (root) – Rechten verwenden/durchführen
- Falls der Benutzer sudo-Rechte hat, dann kann man den Befehl mit ***sudo <Befehl>*** ausführen
- Falls nicht, muss man entweder dem Benutzer diese Rechte geben oder als root einlogen: ***su*** und den Befehl ausführen

```
[gordon@arch Test]$ sudo apt update
[sudo] Passwort für gordon:
```

spline

Freie Universität Berlin

Terminalbefehle

apt:

- ***apt*** ist ein Paketmanager (Frontend) von der Paketverwaltung ***dpkg*** in Debian-basierten Systemen
- Dazu sind 4 folgende Befehle wichtig:

  sudo apt update – Aktualisiert die Paketliste

  sudo apt upgrade – Aktualisiert die Programme

  sudo apt dist-upgrade – Aktualisiert das System

  sudo apt install <Paket> - Installiert ein Programm

  sudo apt remove <Paket> – Löscht ein Programm

spline_

Freie Universität Berlin

Shell-Scripte

Was ist ein Shell-Script?:

- Ein Shell-Script ist ein in Shell geschriebenes Programm
- Im Grunde eine Reihe von Befehlen
- Man schreibt mehrere Befehle hintereinander in eine Datei
- Die Datei hat meistens die Endung ***.sh***, aber nicht notwendigerweise
- Das Programm wird standardmäßig von ***bash*** interpretiert, aber man kann auch andere Shells benutzen

spline_

Freie Universität Berlin

Shell-Scripte

Wie schreibt man einen Shell-Script?:

- Am Anfang soll (nach einem Shebang ***#!***) definiert werden, welchen Interpreter wir benutzen (ein Programm, das den darunter stehenden Text interpretiert)
- Mit der Zeile ***#!/bin/bash*** sagen wir, dass das Programm mit ***bash*** interpretiert werden soll
- Anschliessend müssen wir unsere Datei Ausführbar machen: ***chmod +x <Datei>***
- Jetzt müssen wir das nur noch starten: ***./<Datei>*** oder ***<Pfad zu der Datei>***

spline

Freie Universität Berlin

Shell-Scripte

Wie sieht ein Shell-Script aus?:

- Wir schreiben jetzt ein Script „Hello World“

  Anmerkung: *Kommentare kann man mit # schreiben*

```
[gordon@arch ~]$ touch hello
[gordon@arch ~]$ vim hello
```

```
#!/bin/bash
echo "Hello World" #Gibt "Hello World" aus
```

```
[gordon@arch ~]$ chmod +x hello
[gordon@arch ~]$ ./hello
Hello World
```