

1. Variablen:

- PATH=/bin
 - → ohne Leerzeichen!!! → sonst interpretiert das System /bin als Kommando
- vordefinierte Variablen: haben von der Shell einen bestimmten Wert vorgegeben

- \$...
 - \$# → Anzahl der Kommandoargumente
 - `ls -l /etc` → Wert: 2
 - `rechne 5 2 3` → Wert: 3
 - \$0 → 1. Kommandoargument
 - `ls -l /etc` → Wert: `ls`
 - \$1 → 2. Kommandoargument
 - `ls -l /etc` → Wert: `-l`
 - `rechne 5 2 3` → Wert: `5`
 - ...
 - \$9 → ...
 - \$* → alle Kommandoargumente
 - `rechne 5 2 3` → Wert: `5 2 3`
 - @\$ → = \$*
 - \$? → Rückgabewert des letzten Kommandos
 - `0` → fehlerfreier Ablauf
 - `!0` → Fehler ist aufgetreten
 - \$\$ → enthält Prozessnummer der aktiven Shell
 - \$! → enthält Prozessnummer des letzten Hintergrundprozesses

2. bash (Funktionen)

- Befehlszeilenergänzung:
 - Kommandos und Verzeichnisse mit der Tab-Taste ergänzen
- History der Shell:
 - Taste ↑ → zeigt die letzten Kommandos an
 - `history` → zeigt letzten Kommandos in Tabelle (Nr Kom) an
 - `!Nummer` → altes Kom. ausführen

3. Reguläre Ausdrücke:

- Zeichen werden durch Jokerzeichen ersetzt
- Jokerzeichen
 - * → 0 – n Zeichen
 - ? → ein Zeichen
 - Klammererweiterungen:
 - [...] → stehen für einen der angegebenen Stellen
 - `[abc 1-50]` → `a,b,c,1,2,3,4,5,0`
 - {...} → steht für einen Wert aus den angegebenen Werten
 - `touch dat{1,2,3}` → `dat1, dat2, dat3`

4. Sonderzeichen der Shell

- _ (Leer) → trennt Kommandoargumente
 - `touch guten tag` → 2 neue Dateien `guten` und `tag`
- \$ → greift auf einen Variablenwert zu
- ! → (ver)sucht ein Kommando aus der history auszuführen
- / → Wechsel in das root-Verzeichnis
- ~ → Home-Verzeichnis des akt. Users
- ...

5. Kommandos: `#fdisk`, `(#)mount`, `(#)umount`:

- `#fdisk` → Anzeigen & Verändern(!!!) von Partitionen
 - `#fdisk /dev/hda` → Partitionen der 1. DIE-Platte zeigen
 - `#m` → Kommandos zeigen
 - `#p` → Partition zeigen
 - `#q` → Fdisk verlassen

`(#) mount`, `(#) unmount`

- `# mount -t ntfs -o ro /dev/hda1 /mnt/windoof`
 - `-t ntfs` → Das Dateisystem, das gemountet werden soll ist: NTFS
 - `-o ro` → Mountoptionen: ro = read-only
- `# mount -t ntfs -o ro /dev/hda1`
 - falls das Ziel fehlt, dann mounten wie es in der fstab steht
- `# unmount /dev/hda1`
 - muss gemacht werden BEVOR CDs entnommen werden können
 - vorher das Verz., das geUNmountet werden soll verlassen
- `/proc/mounts` zeigt an welche Verzeichnisse im Moment gemountet sind

6. Übung (Reguläre Ausdrücke)

- Dateien `brief`, `brief1`, `brief2`, `brief3`, `brief11` und `brief14` möglichst einfach anlegen.
 - Lösungen:
 - `> touch brief{1,2,3,11,14}` → (besser)
 - `> touch brief{ ,1,2,3,11,14}` → `brief_` (m. Leer)
 - Mögliche Fehler:
 - `> touch brief brief[1-3]` → `brief`, `brief[1-3]`
- alle `brief`-Dateien anzeigen:
 - Lösungen:
 - `> ls brief*`
- Was zeigt `ls brief?` an?
 - Lösung:
 - Alles, was noch eine Stelle hat und mit `brief` beginnt
 - `brief1`, `brief2`, `brief3`
- Was zeigt `brief[1-3]` an?
 - Lösung:
 - `brief1`, `brief2`, `brief3`
- Wie kann man `brief1`, `brief2`, `brief3`, `brief11` und `brief14` anzeigen?
 - Lösung:
 - `> ls brief[1-3] brief1?`
 - `brief[1-3]` → `brief1`, `brief2`, `brief3`
 - `brief1?` → `brief11`, `brief14`
 - `> ls brief? brief1?`
 - `> ls brief*? / > ls brief?*`
 - `> ls brief[1-3]*`
 - `> ls brief*[1-4]`
 - mögliche Fehler:
 - `> ls brief[1-14]` → `brief1`
 - Stellen! in der Klammer
 - `> ls brief[1-3][14]` → `brief11`, `brief14`
 - alles, dass nach `brief` eine 1,2 oder 3 hat und dann kommt eine 1 oder 4

7. **Schützen von Sonderzeichen:**

- Man kann (normalerweise) keine Dateien anlegen, die im Namen Sonderzeichen enthalten. Aus diesem Grund muss man sie „schützen“.
- `\` schützt alle Sonderzeichen außer „/“
- `„...“` schützt eingeschlossene Zeichen außer „\$, !, /, ...“
- `,...'` schützt eingeschlossene Zeichen außer „/“ (auf der #-Taste)
- Beispiele:
 - `touch „$USER“ → tux || touch ‚$USER‘ → $USER || touch $USER → tux`

8. **Kommandosubstitution:**

- Ersetzen von Kommandos durch ihre Ausgabe.
- ``...`` (neben `←`)
- Beispiele:
 - `> touch backup `date` → 5 vers. Dateien aus einzeln. Datumsteilen`
 - `> touch backup „`date`“ → backup Mit Nov 23 12:32:45 CET 2004`
- `date` Kommando
 - `> date +%H%M → 12Uhr40`
 - `> touch `date +%H%M` → Datei: 12Uhr40`

9. **Ein- und Ausgabeumleitung**

