

1. Bedingte Kommandoverknüpfung:

- `com1; com2` → Hintereinanderausführung
- `com1 && com2` → com2 NUR ausführen, wenn com1 fehlerfrei lief
- `com1 || com2` → com2 NUR ausführen, wenn com1 fehlerhaft

2. Jobkontrolle:

- `Strg-z` → im Vordergrund laufende Programme anhalten
- `com &` → com im Hintergrund ausführen
- `jobs -l` → laufende Jobs anzeigen (in der aktuellen Shell)
- `bs [jobID]` → Prozess in den Hintergrund
- `fs [jobID]` → Prozess in den Vordergrund

3. Prozesskontrolle:

- `ps -axl` → zeigt alle Prozesse an
- `kill -s SIGNAL PID`
 - Signale: 15 SIGTERM friedliches Ende (default)
 - 9 SIGKILL töten
 - 1 SIGUP Konfiguration neu einlesen (b. ändern)
- `top` → ähnlich wie `ps -axl`, aber ständig aktuell

4. Die Shell-Skripte

- muss ausführbar sein (x-Bit setzen): `> chmod skriptName 744`
- 1.Zeile: `#!/bin/bash`
- letzte Zeile: sollte leer sein
- BeispielSkript (halloSkript): Begrüßung des aktuellen Users & akt. Urzeit zeigen
 - 1 `#!/bin/bash`
 - 2 `echo $USER`
 - 3
 - 4 `date +%H\ Uhr\ %M # "\"schützen die Leerzeichen`
 - 5
- Skripte ausführen (, die im aktuellen Verzeichnis sind, welches aber nicht im PATH ist):
 1. `> ./halloSkript`
 2. `> sh halloSkript`
 3. `> /home/tux/halloSkript`
 4. aktuelles Verzeichnis in PATH aufnehmen
 5. Skript ins PATH-Verzeichnis kopieren
- BeispielSkript 2: Backup-Skript (Testat) siehe Datei

5. Wiederholungsanweisungen:

- for
 - Syntax
 - for n in Liste
 - do befehle
 - done
 - Beispiel:
 - for n in 123 # ohne Blank
 - do
 - echo \$n
 - done
 - # Ausgabe: 1 „\n“ 2 „\n“ 3
 - etc/hosts mit einem Skript editieren
 - Ergebnis (SOLL):
 - 143.93.53.1 linux 1 ...
 - 143.93.53.100 linux 100
 - → echo 143.93.53.1 >> /etc/hosts #anhängen (nicht löschen (>))
 - Lösung:
 - #!/bin/bash
 - for n in 1 2 3 4 5 6 ...
 - do
 - echo 143.93.53.\$n linux\$n >> /etc/hosts
 - done
 - Besser: seq → seq 5 (→ Ausg.: 1 „\n“ 2 „\n“ 3 „\n“ 4 „\n“ 5)
 - Format: echo -e „143.93.53.\$n \t linux\$n“ >...
- while
 - Syntax
 - while bedingung # ausführen solange Bed. wahr ist
 - do befehle
 - done # while_test
- until
 - Syntax
 - until bedingung #ausführen bis die Bed. wahr ist
 - do befehle
 - done # until_test
- Exkurs: Zählschleife
 - i = 1
 - while test \$i -le 100 # Zählschl. bis 100 (while lower than 100)
 - do
 - echo \$I
 - i = `expr \$i +2`
 - done

6. Fallunterscheidungen

- **if**
 - Syntax
 - `if bedingung`
 - `then befehle`
 - `[elif bedingung2`
 - `then befehle2`
 - `...`
 - `else befehle]`
 - Beispiel
 - `if test! -e backup` *#prüft, ob das Verz. NICHT existiert*
- **case**
 - Syntax
 - `case var in`
 - `muster1) befehl1;;`
 - `muster2) befehl2;;`
 - `...`
 - `*) befehl;;`
 - `esac`