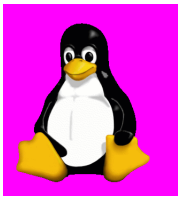
Einige Linux - Befehle

Kursunterlagen



Verfasser: F. Hodel
Ausgabe 2012

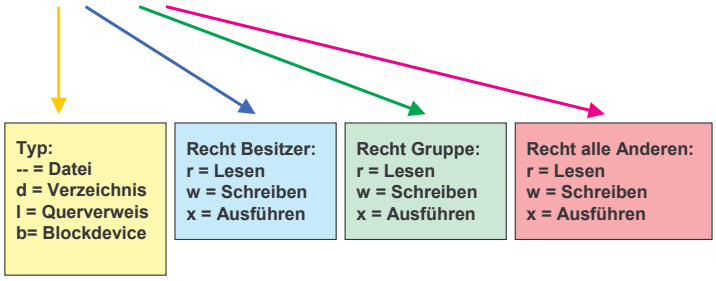
www.anetgmbh.ch

1

Ziel des Kurses ist das Kennenlernen einiger Linux-Befehle.

Datei-Rechte

- Jede Datei gehört einem Benutzer und einer Gruppe
- Diesem Benutzer, der Gruppe und allen anderen können Lese-, Schreib- und Ausführungsrechte zugewiesen werden.
- Anzeige der aktuellen Rechte mit: **ls -l** oder mit: **dir**
- -rwxr-xr-x 1 fho users 15345 Jun 20 15:05 testfile



2

Jede Datei gehört einem Benutzer und einer Gruppe. Die Berechtigung unterscheidet zwischen Besitzer, die Gruppe und allen anderen "(the World").

Für jedes dieser Elemente gibt es drei Rechte:

- r für Lesen
- w für Schreiben
- x für Ausführen (Programme müssen also nicht xxx.exe heissen, sondern das x-gesetzt haben).

An erster Stelle steht noch, ob es sich um ein Directory (d) oder einen Link(l =Verknüpfung) handelt. Ausserdem kann es noch ein Character- oder Block Device sein (Gerät das einzelne Buchstaben oder ganze Blöcke empfangen kann).

Dateirechte ändern

- **chmod** ändert die Rechte
  chmod 744 testfile
  (ändert die Berechtigung auf -rwxr- -r- -)
  (r+w+x = 4+2+1 = 7 für den Besitzer, r = 4 für die Gruppe und alle anderen)
  chmod go-w testfile
  (entzieht der Gruppe (g) und allen anderen (o) das Schreibrecht (-w)) Hinweis: **go-w** ohne Leerschlag!
- **chown** ändert den Besitzer und/oder die Gruppe
  chown fho testfile
  (fho ist der neue Besitzer der Datei testfile)
  chown fho:users testfile
  (fho ist der neue Besitzer und users ist die neue Gruppe der Datei testfile, fho muss Mitglied von users sein)

3

Jede Datei hat einen Besitzer und eine Gruppe. Beide können geändert werden, wobei der neue Besitzer Mitglied der Gruppe sein muss.

Die Ziffernform der Berechtigungen wird bei jeder Datei bei den Befehlen angezeigt:

- dir
- ls -l

Die Berechtigungen können mit dem Befehl **chmod** angepasst werden. Dies kann per Ziffer oder Buchstaben angegeben werden:

- 4 für Read
- 2 für Write
- 1 für Execute

Also:

7 = 4 + 2 + 1 = rwx
6 = 4 + 2 = rw
5 = 4 + 1 = rx

Berechtigungen für ein Verzeichnis mit allen Unterverzeichnissen ändern (rekursiv):

chmod 775 /test/ -R

Absolute und relative Pfade

- Ein absoluter Pfad beginnt immer mit einem **/** und zeigt den gesamten Pfad von root bis zum Ziel. Das aktuelle Verzeichnis ist spielt *keine* Rolle.
- Ein relativer Pfad beginnt **nicht** mit einem **/** und zeigt nur den Weg vom aktuellen Verzeichnis bis zum Ziel

Absoluter Pfad:

cd /etc/samba

Relativer Pfad:

cd samba/data

4

Absolute Pfade geben immer den vollständigen Pfad bis zu Date an. Das ist zwar länger, aber die Datei wird immer korrekt gefunden. Relative Pfade beginnen beim aktuellen Verzeichnis (pwd print working directory). Sie sind kürzer oder können gar ganz fehlen, wenn die Datei im aktuellen Verzeichnis ist. Allerdings wird die Datei nicht mehr gefunden, wenn ich das aktuelle Verzeichnis ändere.

Die Logik ist unter Windows genau gleich, es müssen lediglich die / durch einen \ ersetzt werden.

Kopieren und umbenennen

- **cp** „alte Datei“ „neue Datei“
  Datei kopieren (allenfalls mit Pfaden)
- **mv** „alte Datei“ „neue Datei“
  alte Date in neue Datei umbenennen,
  (alte Datei existiert dann nicht mehr)
- **rm** löscht eine Datei, Recursiv (-R) auch ganze Verzeichnisstrukturen (Vorsicht!)

```
cp /etc/samba/smb.conf  /etc/samba/smb.conf_sav
mv  test1.txt  test2.txt
rm   test2.txt
rm -R /test/      (Löscht Verzeichnis /test mit allen Unterverzeichnissen)
rmdir /test       (Löscht das einzelne Verzeichnis /test)
```

5

Der Copy-Befehl cp kann Dateien kopieren auf einen neuen Namen, auf Wunsch auch gleich in ein neues Verzeichnis.

Move (mv) kann Dateien in ein anderes Verzeichnis verschieben oder einfach den Namen ändern.

Remove (rm) kann Dateien und ganze Verzeichnisse löschen, auch Rekurisv. Vorsicht: tun Sie dies nicht als root im /-Verzeichnis, so löschen Sie das ganze System!

Ein einzelnes Verzeichnis kann mit rmdir gelöscht werden (darf dann keine Unterverzeichnisse haben).

Dateien suchen

- **find** [Pfad] [Ausdruck]
  - Fehlt der Pfad wird im aktuellen Verzeichnis gesucht
  - Unterverzeichnisse werden auch durchsucht
  - maxdepth bestimmt die Suchtiefe
  - Name gibt den zu suchenden Namen an (oder den ersten Teil davon)
- Beispiele

  find /etc/ -name named*
  (sucht im Verzeichnis /etc alle Dateien, die mit named beginnen, auch in den Unterverzeichnissen)

  find /etc/ -maxdepth 2 -name named.c*
  (sucht im Verzeichnis /etc und *zwei* Stufen der Unterverzeichnisse alle Dateien, die mit named.c beginnen)

6

find dient zum auffinden von Dateien. Damit kann im aktuellen Verzeichnis oder einem anderen Verzeichnis gesucht werden. Man kann auch in allen Unterverzeichnissen suchen lassen (was einige Zeit dauern kann) oder die Suchtiefe begrenzen, so dass z.B. nur in diesem Verzeichnis und der ersten Stufe Unterverzeichnisse gesucht wird.

in Dateien suchen

- **grep** [ausdruck] [Pfad und Datei]
  sucht alle Zeilen in der angegebenen Datei, in denen „Ausdruck“ vorkommt (muss nicht am Zeilenanfang stehen)
  Beispiel:
  grep hans /etc/passwd
  (Sucht alle Zeilen in der Datei /etc/passwd, in denen hans vorkommt)
- **cat** [Pfad und Datei]
  zeigt den Inhalt einer (Text-) Datei an
  Beispiel:
  cat /etc/passwd
  cat /etc/passwd | less (mit Pausen für jede Seite, beenden mit **[Esc] :q**)
- **tail** -f [Pfad und Datei]
  zeigt immer das aktuelle Ende einer Datei an
  Beispiel:
  Tail -f /var/log/messages

7

Auch im Inhalt der Dateien kann gesucht werden.

grep zeigt alle Zeilen einer Datei an, die einen bestimmten Begriff enthalten, egal ob am Anfagn der Zeile oder weitehinten. Dies mach eher nur Sinn bei Text-Dateien.

cat listet die ganze Datei auf (wie type unter Windows). Wenn die Datei grösser ist, sollte mit einer Pipe (|) der Filter more oder less nachgeschaltet werden. So hält die Anzeige bei jeder Seite an. less gestattet auch das Zurückblättern. Zum Beenden von less dient wie beim Editor vi: [Esc] : q [Enter]

tail eignet sich zur Verfolgung der aktuellsten Meldungen in Log-Dateien, es zeigt immer das aktuelle Ende der Datei an. Beendet wird es mit [Ctrl] + c

Datei erstellen

- Oft benötigt man kleine Dateien, etwa zum Testen. Dies erstell man m Besten mit:
- **echo**
  echo schreibt Text auf den Bildschirm. Man kann aber die Ausgabe in eine Datei umleiten, dann wird die Datei erstellt und der Text steht in der Datei drin.
  ```
  Echo  Hallo miteinander > /test/text1.txt
  ```
- Variante: mit >> wird die der Text an die Datei angehängt, falls sie bereits besteht, sonst wird sie erstellt.
  ```
  Echo  und die zweite Zeile > /test/text1.txt
  ```
- **touch**
  touch erstellt eine leere Datei. Falls sie bereits besteht, wird das Änderungsdatum af das aktuelle Datum gesetzt.
  ```
  touch /test/text1.txt
  ```

8

Eine Datei mit Text-Inhalt kann mit echo erstellt werden. Dies eignet sich etwa zum Schreiben von Log-Dateien und Test-Dateien. Die Funktion ist übrigens unter Windows genau gleich vorhanden. Wird als Umleitsymbol > verwendet, wird die Datei in jedem Fall neu erstellt (das heisst, wenn sie bereits besteht, wird sie überschrieben). Mit dem Symbol >> wird der neue Inhalt hinten an die bestehende Datei angefügt.

Touch wird oft verwendet um das Änderungsdatum aller Dateien einer neuen Programmversion mit dem gleich Datum zu versehen.

ifconfig

- Konfiguration der Netzwerkschnittstellen (LAN, WLAN)
- Benötigt root-Rechte (sonst: „Befehl nicht gefunden“)
- Beispiele:

```
Ifconfig                    (zeigt aktuelle Konfiguration an)
ifconfig  eth0  192.168.112.43  netmask 255.255.255.0 (setzt IP)
ifconfig  eth0  down     (stoppt die erste Ethernetkarte)
ifconfig  eth0  up       (aktiviert die erste Ethernetkarte)
ifconfig  eth0  hw ether  40:00:11:22:33:44  (Mac-Adresse ändern)

ifup  eth0               (Kurzform zum Aktivieren
dhcpcd  -n  eth0         (neue DHCP-Adresse für eth1 beziehen)
                         Vorsicht: dhcpcd nicht dhcpd! = Client Daemon
```

9

ifconfig dient zur Konfiguration von LAN und WLAN-Schnittsellen. Beachten Sie, dass die Änderungen nur temporär sind, da normalerweise die Schnittstellen in /etc/init.d/network beim Booten definiert werden.

Hinweis zu VMWare: Wenn das Netzwerk unter VMWare nicht funktioniert, liegt da meist daran, dass das Image kopiert wurde. VMWare generiert dann für die (/virtuelle) Karte eine neune MAC-Adresse (was sinnvoll ist). In der Konfiguration ist dann aber immer noch die alte, geerbte Netzwerkkarte drin und die geht dann nicht. Die Lösung dazu ist:

-YaST starten (mit root-Passwort)

-Netzwerkgeräte → Netzwerkeinstellungen

Nun wird die geerbte Karte angezeigt. Man erkennt sie daran, dass man sie löschen kann, und genau das sollte man tun [Löschen] . Sofort erscheit dann die neue Karte (die man nicht löschen kann) und diese kann man nun via [Bearbeiten] konfigurieren mit DHCP oder einer fixen IP-Adresse.

Job Control

Prozesse
Prozessnummern
Runlevel

10

Der erste Prozess, der vom Kernel gestartet wird ins das Programm init und erhält die Prozess-Nummer 0. Anschliessend werden vom System und schliesslich vom Benutzer weitere Prozesse gestartet, die alle eine eindeutige Prozessnummer PID erhalten.

Anzeige der laufenden Prozesse

- **top**
  zeigt die momentan laufenden Prozesse an, automtisch aktualisiert und sortiert nach CPU und Memory Belastung
- **ps** -ef | more
  zeigt alle laufenden Prozesse an, sortiert nach Prozessnummer (PID)
- **kill** 345
  stoppt den Prozess mit der PID 345.
  Vorsicht: das Programm wird sofort abgeschossen, nicht normal beendet!
- **xkill**
  Grafische Variante von kill: Totenkopf-Kursor auf das zu beendende Programm setzen und linke Maustaste drücken.

11

top eignet sich zur aktuellen Anzeige der momentan aktiven Prozesse. Die Anzeige ändert so, dass immer der belastendste Prozess zuoberst erscheint.

ps zeigt alle Prozesse an. Da diese recht viele sein können ist es sinnvoll mit dem more oder less Filter zu arbeiten.

kill kann einen beliebigen Prozess abschiessen. Dies ist keine normale Beendigung und man sollte sich die Konsequenzen vorher überlegen.

xkill ist die grafische Variante von kill und eignet sich für grafische Programme, nicht für Hintergrundprogramme.

Testen, ob ein Prozess läuft

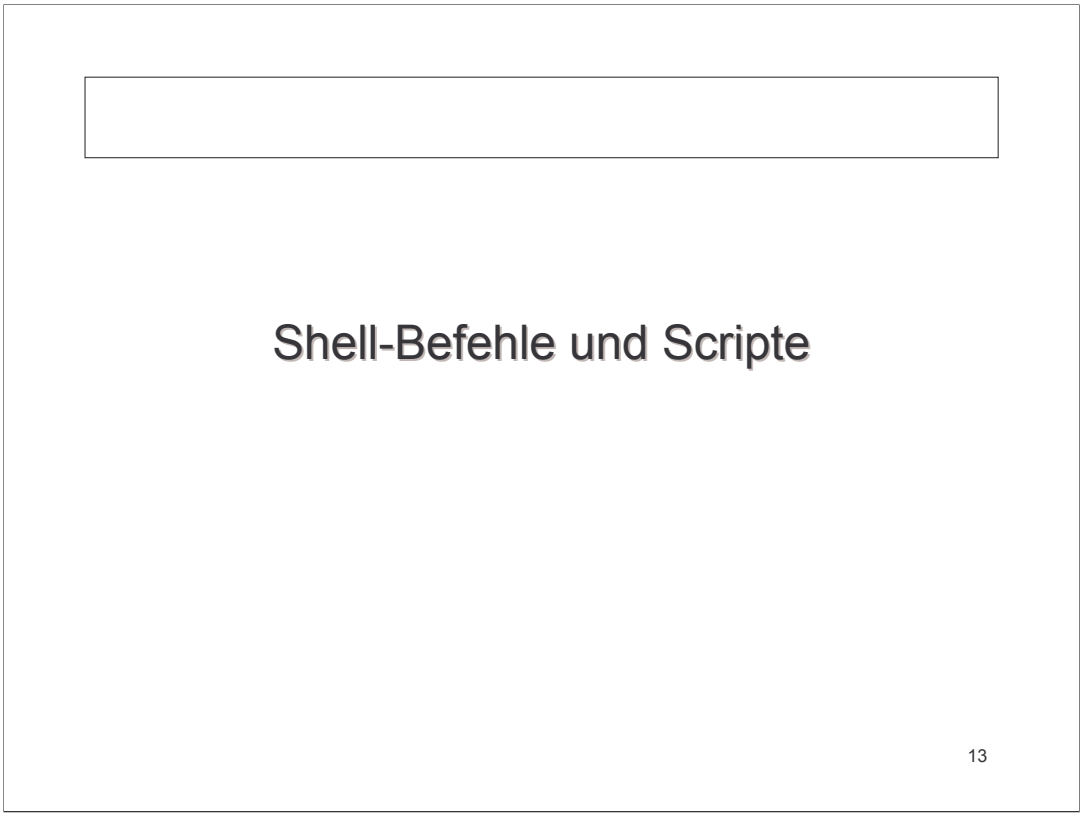
- Funktionen sind im Verzeichnis /etc/init.d
- Meistens mehrere Funktionen verfügbar:
  - start (Dienst starten)
  - stop (Dienst beenden)
  - restart (Dienst neu starten, z.B. nach Neukonfiguration)
  - reload (Konfiguration neu einlesen)
  - status (Abfragen, ob Dienst läuft)
- Beispiele

/etc/init.d/smb status	(läuft der Samba smb Dienst?)
/etc/init.d/nscd stop	(Name Server Cache Dienst stoppen)
rcsmb status	(Variante bei SuSE, immer mit rc....)
rcSuSEfirewall2 stop	(Firewall stoppen)

12

Alle Prozesse werden letztendlich von init gestartet. Die automatisch zu startenden Prozesse werden im Verzeichnis /etc/init.d eingetragen. Dies geschieht mit einem Script. Viele erlauben es, Prozesse zu starten, stoppen und neu-starten. Auch der Status lässt sich normalerweise abfragen.

Bei SuSE gibt es eine Abkürzung zu diesen Funktionen. Dies heissen rcxxxxx, also rc gefolgt vom Programm-Namen.



Shell-Befehle sind mächtig und eigenen sich – im Gegensatz zu einer grafischen Oberfläche – sehr gut zum Automatisieren. Wir lernen einige wichtige Befehl kurz kennen. Die möglichen Parameter kann man sich anzeigen lassen mit man befehl, also z.B.:

man rsync

Beendet wird die Anzeige mit [Esc] : q

Bourne Shell

- Verschiedenen Shell-Varianten sind verfügbar
- Viel Funktionen/Befehle sind gemeinsam
- Die bourne again Shell (bash) ist am meisten verbreitet
- Shell-Script erstellen:
  - Erste Zeile enthält die Information, mit welcher Shell das Script laufen soll:
    #!/bin/bash oder **#!/bin/sh**
  - Der Name benötigt *keine* bestimmte Extension (wie unter Windows .cmd), oft wird .sh genommen
  - Hingegen muss bei den Berechtigungen das Execute-Bit gesetzt sein, sonst wird es nicht gestartet
  - Da unter Linux das aktuelle Verzeichnis *nicht* im Suchpfad ist, muss der Pfad angegeben werden oder (falls das Script im aktuellen Verzeichnis ist) **./** vorangestellt werden (. = aktuelles Verzeichnis)

14

Unter Linux kann man einem Benutzer verschiedene Shell‘s zur Verfügung stellen. Am häufigsten wird die Bourne-Shell bash benutzt. Die Funktionen können je Shell unterschiedlich sein. Deshalb wird in einem Script auf der ersten Zeile angegeben, mit welcher Shell es laufen soll.

Wenn das Script mit einem Editor (z.B. vi) erstellt wird, entsteht eine Text-Datei. Dies kann so *nicht* ausgeführt werden. Deshalb muss dem Script das Execute-Bit gesetzt werden, erst dann lässt es sich aufrufen. Die kann mit dem Befehl **chmod** erreicht werden. Anders als unter Windows spielt die File-Extension keine Rolle. Die Datei kann also heissen, wie sie will. Viele Leute benutzen die Extension .sh.

Variable, Eingabe

- Variable haben einen Namen
- Nicht definierte Variable sind leer
- Beim Abrufen wird dieser Name mit vorangestelltem $ benutzt: $hausnummer
- $1 $2 erster bzw. zweiter eingegebener Parameter
- Echo gibt einen Text am Bildschirm aus (stdout)

```
#!/bin/sh
a=hello     # einer Variablen "a" den Wert "hello" zuweisen
echo $a     # die Variable "a" (am Bildschirm) ausgeben
a=7         # der Variablen "a" einen neuen Wert zuweisen
b=8         # einer Variablen "b" den Wert "8" zuweisen
c=$((a+b))  # einer Variablen "c" die Summe von a+b zuweisen
             # ("$((a+b))" wird substituiert)
echo $c     # die Variable "c" (am Bildschirm) ausgeben
```

15

Programme müssen oft für unterschiedliche Daten benutz werden. Da sind Variable wichtig. Die Variablen werden definiert mit einer Zuweisung (i=xxxxx) und wieder abgerufen mit ihrem jeweiligen Namen mit vorgestelltem $-Zeichen ($i). Mit Variablen kann man auch rechnen (wenn Zahlen drin sind).

Variablen-Namen müssen mit einem Buchstaben beginnen. Die Variablen $1, $2 etc. gibt es zwar auch, sie enthalten aber die dem Programm übergebenen Parameter.

Variable 2

- Variable Zusammensetzen

```
h=hoch r=runter l=    #Weist den drei Variablen Werte zu,
                        #wobei l einen leeren Wert erhält.
echo ${h}sprung       #Gibt hochsprung aus. Die Klammern müssen gesetzt
   werden,
                        #damit h als Variablenname erkannt werden kann.
echo ${h:-$r}         #Gibt hoch aus, da die Variable h belegt ist.
                        #Ansonsten würde der Wert von r ausgegeben.
echo ${tmp:-`date`}   #Gibt das aktuelle Datum aus, wenn die Variable
   tmp
                        #nicht gesetzt ist. (Der Befehl date gibt das
   Datum zurück)
echo ${l:=$r}         #Gibt runter aus, da die Variable l keinen Wert
   enthält.
                        #Gleichzeitig wird l der Wert von r zugewiesen.
echo $l               #Gibt runter aus, da l jetzt den gleichen Inhalt
   hat wie r.
```

16

Variablen kann man auch mit fixem Text oder anderen Variablen zusammensetzten. Standardmässig wird ein Leerschlag dazwischen eingefügt. Will man dies nicht, muss das Escape-Zeichen angegeben werden: \

Muster (Parsing)

- Variabeln können analysiert werden

```
#!/bin/sh
echo Mustererkennung
echo .
echo Alle Dateien, die mit m beginnen
ls m*

echo  Alle Dateien, die mit a bis m beginnen
ls [a-m]*

echo Alle Dateien, die mit a, m oder v beginnen
ls [amv]*

echo Alle Dateiein, die NICHT mit m beginnen
ls [!m]*
```

17

Die eingegebenen Werte müssen meist analysiert werden (Parsing). So können z.B. Muster erkannt werden. Muster werden in [ ] angegeben werden. Ein vorangestelltes ! kehrt den Sinn um (Negation). Ein ? Steht für ein beliebiges Zeichen, ein Stern für mehrere beliebige Zeichen (auch keines).

if Bedingungen

- if Bedingung
- then Befehle, falls wahr
- else Befehle, falls nicht wahr
- fi Ende der bedingten Befehle

```
echo "Bitte Benutzernamen als Parameter eingeben"

if who | grep -q $1    # who: Liste der Benutzer
                       # grep: Suche nach Muster
  then echo $1 " ist angemeldet"
  else echo "Benutzer " $1 " ist NICHT angemeldet"
fi
```

18

Oft müssen Anweisungen nur ausgeführt werden, wenn bestimmte Bedingungen erfüllt sind. Die Bedingung beginnt mit **if** und endet mit der Umkehrung **fi**. Ist die Bedingung wahr, werden die Anweisungen nach then ausgeführt, sonst jene nach else. Eine leere Anweisung besteht aus **:** (NOP=no operation)

Schleifen 1

- for Beginn der Schleife
- i = 254 Zählvariable (beginnt hier bei 254)
- $i aktueller Wert von i
- i=$i-1 i wird anschliessend um 1 erniedrigt
- do es folgen die Befehle
- done Ende der Schleife

```
#!/bin/bash
echo pinge alle Adressen im Subnet:

for ((i=254; $i; i=$i-1)) do ping -c 1 192.168.112.$i; done
```

19

Soll die gleiche Funktion für eine Reihe von Werten ausgeführt werden, eignet sich eine Schlaufe. Hier werden alle IP-Adresse in einem Class C subnet angepingt. Anschliessend könnte im ARP-Cache alle Mac-Adressen der aktiven Stationen abgerufen werden (funktioniert auch bei aktiviertem Firewall).

Schleifen 2

- for Beginn der Schleife
- i = 0 Zählvariable (beginnt hier bei 0)
- i < 5 Abbruchkriterium: solange i unter 5 ist
- i++ i wird anschliessend um 1 erhöht
- ++i Variante: i wird vorgängig um 1 erhöht
- done Ende der Schleife

```
#!/bin/bash
echo fuehre Schlaufe 5 mal aus:

for ((i=0;i<5;i++)); do
  echo \$i ist jetzt:  $i     # do something
  done
```

20

Ein andere Variante führt eine Schlaufe für alle Werte von i aus, bis ein bestimmter Wert erreicht ist. Beachten Sie, dass eine solche Schlaufe endlos läuft, wenn das Abbruchkriterium nicht genau getroffen wird! Deshalb ist ein Test > Endwert sicherer, als = Endwert.

Die Zählvariable i kann mit i++ nach dem Ausführen der Schlaufe oder mit ++i vorher erhöht werden.

Schleifen 3

- while Schleife wird endlos ausgeführt, solange die Bedingung erfüllt ist
- [.....] Bedingung
- do auszuführende Befehle
- done Ende der Schleife

```
#!/bin/bash
echo Schleife mit while: Alle eingegebenen Parameter anzeigen


while [ -n "$1" ]; do
     echo "Parameter gefunden: " $1
     shift     # mit shift werden die Parameter nach
               # Links geshiftet (aus $2 wird $1)
done
```

21

while wird solange ausgeführt, biss ein Abbruch-Kriterium erfüllt ist. Hier ist offensichtlich, dass beim Verpassen dieses Kriteriums eine Endlos-Schleife entsteht.

Eigene Funktion count

- count () Name der eigendefinierten Befehls
- ls Anzeige der Dateien (list)
- | Resultat in nächste Funktion übergeben (Pipe)
- wc Word Count, hier mit –l : Zeilen zählen

```
#!/bin/bash
count () {
      ls | wc -l  # ls: Liste aller Objekte im Verzeichnis
                  # wc: Word-Count; mit Attribut -l werden Zeilen gezählt
                  # in Verbindung mit ls werden also die (nicht
                  # versteckten) Objekte gezählt
}
count        # Aufruf der Funktion
```

22

Hier definieren wir eine eigene Funktion count, welche alle Zeilen der Anzeige des Verzeichnisinhalts zählt, dies mit Hilfe der existierenden Funktion wc.

Returncodes, stdin, stdout,stderr

- Fast alle Programm geben Return-Codes zurück
- 0 Bedeutet: alle ging korrekt
- 1 oder grösser: mindestens ein Teil ging schief

23

Programme geben einen Return-Code zurück. Dieser ist 0 wenn das Programm normal beendet wurde. Werte grösser 0 zeigen an, dass etwas nicht funktioniert hat. Was genau 1 und 2 etc. bedeuten hängt von der jeweiligen Funktion ab. Meist gilt, je grösser der Wert desto mehr ging daneben.

rsync

- Programm zum Abgleich von Dateien in verschiedenen Verzeichnissen (auch auf verschiedenen Systemen und über das Netzwerk)
- Eignet sich zum Sichern von Dateien, weniger für Datenbanken
- Beispiel:
  rsync -azv --delete-after -e ssh /data/ \
  hans@192.168.112.52:/save_data/
  (Sichert Dateien im Verzeichnis /data/ über ssh auf das System mit der angegebenen IP und dem Benutzer hans. Im Original gelöschte Dateien werden auf dem Zielrechner auch gelöscht)

24

rsync dient dem Synchronisieren von Verzeichnissen und eignet sich gut für Sicherungszwecke. Es gibt zahlreiche Parameter. Im Beispiel werden die Daten in einem Verzeichnis auf ein entferntes System übertragen. Dabei erfolgt die Übertragung verschlüsselt mit ssh (Secure Shell) und Dateien mit dem gleichen Datum/Uhrzeit werden nicht kopiert. Werden im ursprünglichen Verzeichnis Dateien gelöscht, werden dies auch im Fernen Verzeichnis gelöscht. (Mit dem \ wurde der Befehl auf 2 Zeilen verteilt, so ist die Schrift besser lesbar).

Dabei bedeutet azv:

-im Archiv-Modus mit Unterverzeichnissen

-Die Übertragung erfolgt komprimiert

-Im Verbose Modus werden die Dateien am Bildschirm angezeigt