Unix-Shell

Arbeiten mit der bash

Hilfe

1. Editieren in der bash
   Pfeiltasten, TAB, history, !aa
2. man 'command' $\Rightarrow$ man ls
3. Erklären: Optionen beginnen mit '-' command -h, command -?
4. apropos oder man -k
   alles über pdf $\Rightarrow$ apropos pdf
   auch: apropos gif
   (eigentlich: alles was in der Übersichtszeile den Suchstring enthält)
5. cd-Befehl
   Verzeichnisse mit '/' nicht mit '\'
   'cd /' , 'cd ..' , 'cd ' ($\Rightarrow$ HOME)
6. Erzeugen einer neuen leeren Datei (eigentlich Ändern der Zeitstempel)
   touch *filename*
7. Erzeugen einer Datei mit einer Zeile Inhalt
   echo "Inhalt" $>$ *filename*

Arbeiten mit der bash (Forts.)

Befehle

8. ls-Befehl:
   Ausgabe erklären, auch Zugriffsrechte
   ls -a , ls -A , ls -l, ls -al, ls -il (inode-Nr),
   (-help, -d, -r, -R, -S, -t)
   Wichtige Systemverzeichnisse mit: 'ls /bin' , ...
9. Erzeugen einer Datei mit einer Zeile Inhalt
   echo "Inhalt" > *filename*
10. Anzeigen des Dateiinhalts
    cat *filename*
11. Erzeugen einer Datei mit Inhalt von Tastatur (Ctl-D = Ende)
    cat > *filename*
    (Stoppen mit Ctl-C)
12. Anfuegen an Dateiende
    cat >> *filename*
13. less 'filename' (/ $\Rightarrow$ darin suchen mit "/*pattern*")

Arbeiten mit der bash (Forts.)

Befehle

14. head ; tail (Optionen später als Übung)
15. cp -R (recursive)
    cp -i (interactive)
    cp -p (preserve, wem gehören die Dateien? $\Rightarrow$ alter Besitzer)
16. ln (Hard-link, Link-Konzept erklären, ls -l)
    ln -s (Sym-link erzeugen)
17. Benutzer:
    who, whoami, id, groups
    finger, finger -l (Finger-Name ändern: chfn)
18. Zugriffsrechte, Besitzer, Gruppe ändern:
    chmod ug+rw /.bash_history
    chmod 600 /.bash_history

    chown (nur als Superuser)

    chgrp ("groups" zum Anzeigen der möglichen)

Arbeiten mit der bash (Forts.)

Editor und Umgebungsvariablen

19. nano bash (Editor)
    eingeben ⇒ echo 'Meine Bash ist gestartet'
    ^O ^X (speichern und beenden)
20. Welche bash wird aufgerufen???
    ausprobieren!
    ⇒ which bash

    Wie ändern? ⇒ ausführbar machen und Pfad angeben :
    chomd a+x bash; ⇒ "./bash"
21. Umgebungsvariable PATH ⇒ echo $PATH, echo "$PATH"
    ⇒ NICHT mit einfachen Anf.-Zeichen: echo '$PATH'
22. alle Umgebungsvariablen:
    env (⇒ USER, PATH, HOME, SHELL, TERM, ...)
23. neue Umgebungsvariable: ⇒ NEU="Hello World" (OHNE BLANKS!)
    echo $NEU
    bash ; echo $NEU
    ⇒ dann wdh. nach "export NEU"

Arbeiten mit der bash (Forts.)

Suchpfad und Umleitungen

24. Suchpfad anpassen:
    ⇒ export PATH=∼/bin/:$PATH
    KEINE BLANKS, SICHTBAR auch für SOHN-Prozesse
    (Vorsicht mit "." für aktuelles Directory ⇒ Sicherheitsproblem!)
25. einfügen in ∼/.bashrc ⇒ wird von allen bash-Shells beim Start ausgeführt
26. Welche Prozesse laufen?
    ⇒ ps, ps axw
    ⇒ top
27. grep *pattern filename*
    grep PATH ∼/.bash_history
    ⇒ Optionen?
28. stdin, stdout, stderr
    Umleitungen mit < > >> 2 >
    cat file >/dev/null
    nur Fehlermeldungen umleiten: befehl 2 >/dev/null

Prozesse (Forts.)

bash: Bourne Again Shell

29. Pipes "|" (stdout mit stdin des Folgebefehls verbinden)
    cat /etc/passwd | grep bash | grep pagnia
30. Mehrere Befehle nacheinander ausführen mit ";":
    ps axw | grep pagnia; ps auw | grep root
31. Hintergrundprozess:
    Beispiel mit "top"

    Prozess stoppen: Ctl-Z
    Prozess in den Hintergrund schicken: bg
    Prozess wieder in Vordergrund holen: fg
    Prozess direkt im Hintergrund starten: *befehl* &
32. Prozess abbrechen
    Ctl-C
    kill, kill -9 (kill ist eingebaute Funktion der der bash, nicht /bin/kill)

Arbeiten mit der bash (Forts.)

Shell Scripts

33. Ein einfaches Shell-Skript (myls) (Pfad zur bash mit "which bash")

```
#!/usr/local/bin/bash
echo Auflistung des Verzeichnisinhalts:
# Kommentarzeile
ls
```

34. ausführen mit ". myls"
oder "sh myl" oder "bash myls"
"chmod a+rx myls; ./myls"

35. Shell-Skript (cruise_list)

```
#!/bin/bash
for i in A E I O U Fertig
# oder mal ersetzen durch " for i in ` ls * ` "
do
echo "momentaner Wert von i = $i"
done
```

Arbeiten mit der bash (Forts.)

Shell Scripts

36 Shell-Skript (myparam):

```
#!/bin/bash
echo "aufgerufenes Programm:" $0
for i in $@
do
echo "Parameter: $i"
done
```

AUFRUFE: myparam SOWIE myparam 1 2 3

37 Shell-Skript (papagei):

```
#!/bin/bash
while (true) do
read w
case $w in
'klappe' ) echo '> NA GUT :-('; exit;;
* ) echo '>' $w | tr '[:lower:]' '[:upper:]'
esac
done
```

Arbeiten mit der bash (Forts.)

Archive, Suchen und Speicherplatz

38. Archive anlegen, komprimieren:
    tar cvf temp.tar *
    tar tf temp.tar
    tar xf temp.*
    tar czf temp.tgz * (mit Komprimierung)
    zip temp.zip * (Windows ZipFiles)
    unzip temp.zip
39. Finden von Dateien:
    find ~ -name myls -print 2>/dev/null
40. Speicherplatz-Infos:
    du
    du -s ~
    du -s ~/*

    df

Arbeiten mit der bash (Forts.)

sed und awk

41. String überall im Text ersetzen:
    sed -e s/abc/ABC/ file.txt (ersetzen von abc; 1x pro Zeile)
    sed -e s/a../ABC file.txt (ersetzen mittels regular expression)
42. awk (prima für Reports und Listen)
    awk '/root/ {print $0}' /etc/passwd
    (untersucht alle Zeilen, $0 = alle Spalten ausgeben)
43. Mit Pipes: alle Dateien im Arbeitsverzeichnis größer als 200 Bytes:
    ls -l | awk '$5 > 200'
44. Mehrwertsteuer-Rechner, Einlesen von Tastatur (Ende = Ctl-D):
    awk '$1 > 0 {print $1, "+ Steuer = ", $1*1.19, "\n"}'
45. als Datei 'steuer.awk':
    # Berechnung der MwSt
    $1 > 0 {print $1, "plus 19 % Steuer = ", $1*1.19, "\n"}

    Aufruf mit: awk -f steuer.awk

Arbeiten mit der bash (Forts.)

awk

46. formatierte Ausgabe:
    echo "3 4 5" | awk'{printf("hallo %s\n", $1);}' (Semikolon am Ende!)
47. Umlenken der Ausgabe auf Standardfehlerausgabe:
    printf ("\n") > "/dev/stderr"
48. eigene Variablen definieren (alle sind mit 0 initialisiert)
    z. B.: i, j oder sum

    bereits vordefinierte Variablen, u.a.:
    NR ⇒ akt. Zeilennummer
    NF ⇒ Anzahl der Felder dieser Zeile

    BEGIN { ... } ⇒ Block wird nur 1x am Anfang ausgeführt
    END { ... } ⇒ nur 1x am Ende

    Beispiel:
    # avg.awk: compute average of pos. inputs
    $1>0 {sum += $1; anz++}
    END {printf("AVG = %f \n", sum/anz);}

Arbeiten mit der bash (Forts.)

awk

49 auch bedingte Anweisungen: ⇒ if / else
auch Schleifen: ⇒ for ⇒ while
auch Funktionen: ⇒ function xy(*params*)
Beispiel:

```
# ggt.awk: compute gcd(a,b) recursively
function ggt(a,b){
    if (a == b)
        return a;
    if (a > b)
        return ggt(a-b,b);
    else
        return ggt(b,a);
}
BEGIN {
    printf("\na = ");
    getline a < "-";
    printf("b = ");
    getline b < "-";
    printf ("\n   ggt(%d,%d) = %d\n", a, b, ggt(a,b));
    exit
}
```