

2. Testat Skript:

- Lösung:


```
#!/bin/bash
# 2. Pflichtuebung Linux
# im Home des eingel. Users ein Verzeichnis Sicherung anlegen, falls es nicht
# existiert.
# im Verz. Sicherung die Dateien bak.hostts, bak_fstab und bak_passwd an-
# legen.
# die Inhalt der /etc/hosts, /etc/fstab /etc/passwd soll rein kopiert werden
# dabei soll die passwd so gesichert werden, das die Sicherungen sich nicht
# gegenseitig überschreiben (zeitabhängiger Dateiname)
```

```
HOME=/home/tux/test
```

```
if [ ! -e $HOME/sicherung ]
then mkdir $HOME/sicherung
fi
```

```
cd sicherung
```

```
touch bak{,_hosts,_fstab,_passwd}
```

```
cat /etc/hosts > bak_hosts
cat /etc/fstab > bak_fstab
cat /etc/passwd > bak_passwd_`date +%s`
```

- Sicherungsdatei anlegen:
 - `touch bak_hosts$(date +%D\ %H:%M:%S)`
 - → Fehlermeldung: Pfad nicht gefunden (Pfad 11/12/03 wird gesucht)
 - → `$(date...)` müsste nochmals geschützt werden
- prüfen, ob das Verz. backup bereits existiert / sonst anlegen:
 - `if (!test ~/backup) ...`
 - `!test` → testet auf Nicht-Existenz
 - Option fehlt worauf geprüft werden soll
 - Lösung: `if(test -d ~/backup) ...`
- Datei bak_hosts anlegen, falls es existiert, dann alles vorher sichern:
 - `touch bak_hosts || mv bak_hosts bak_hosts$(date ...)`
 - 1. Ausführen → com1 wird ausgeführt → Datei anlegen
 - x. Ausführen:
 - `touch machen (+Zeitstempel anpassen)`
 - `alles nach ODER wird nie ausgeführt, da 1.Bed. immer true`

1. ...Skripte:

- Kommando `expr`
 - interpretiert übergebene Strings als ganze Zahlen
 - `expr 2 + 1`
- Kommando `test...`
 - `test -eq` → testet auf Gleichheit von Ganzzahlen
 - `test -ge` → int1 >/= int2
 - `test -gt` → int1 > int2
 - `test -le` → int1 </= int2

- `test -lt int1 < int2`
- `test 5 -eq 4` → echo \$? (gibt Fehlercode aus) → 0 = richtig
- Kommando `[`
 - gleichbedeutend mit dem Kommando `test`
 - `[5 -eq 4]`
- Kommando `((...))`
 - übergebene Zeichen als Ganzzahlen interpretieren
 - `echo ((5 - 2))`

2. Übung:

- Skript schreiben, das vom User 2 Ganzzahlen abfragt und alle 4 Grundrechenarten durchführt:
 - Abfragen der Werte:

```

1  #!/bin/bash
2
3  echo "Bitte geben sie die erste Zahl ein: "
4  read Zahl1
5
6  test $Zahl1 -eq $Zahl1 ||
7  {
8      2 > /dev/null
9      echo "Gib eine Ganzzahl ein du Trottel !"
10     exit
11 }
12
13 echo "Bitte geben sie die zweite Zahl ein: "
...

```

- 3 → Ausgabe auf dem Monitor
- 4 → eingegebenen Wert in der Variable Zahl1 speichern
- 6 → Zahl mit sich selber vergleichen
- → gibt false zurueck wenn in Variable ein String steht
- → Bei False wird zweite Anweisung ausgefuehrt (Zeilen 7-11)
- 8/9 → Ausgabe auf dem Monitor
- 10 → Skript wegen fehlerhafter Eingabe verlassen
- 13... → Eingabe der zweiten Zahl...

- Berechnungen durchführen:

```

...
1  echo "Addition"
2  echo $Zahl1 + $Zahl2 = $((Zahl1 + Zahl2))
3
4  echo "Subtraktion"
5  echo $Zahl1 - $Zahl2 = $((Zahl1 - Zahl2))
6
7  echo "Multiplikation"
8  echo $Zahl1 \* $Zahl2 = $((Zahl1 * Zahl2))
9
10 echo "Division"
11 if test $Zahl2 -eq 0
12 then
13 {
14     echo "Division durch 0 ist nicht erlaubt!"
15     exit
16 }
17 else
18     echo $Zahl1:$Zahl2 = $((Zahl1 / Zahl2)), Rest $((Zahl1%Zahl2))
19

```

- 1 → Ausgabe am Monitor
- 2 → Berechnung der Addition
- 5 → Berechnung der Subtraktion
- 8 → Berechnung der Multiplikation („*“ muss geschützt werden!!!)
- 11 → prüfen, ob Zahl2 = 0 ist

- 12-16 → falls Zahl2 = 0, dann Fehlerausgabe + Beenden des Skripts
- 18 → Berechnung der Division und des Rests

3. Der Systemstart:

- /etc/inittab
 - bestimmt den Default-Runlevel: Runlevel5 (in etc/init.d/rc5.d)
 - verweist auf Startskripte
- /etc/init.d/boot
 - Skript, das die fstab aufruft
 - etc/fstab: mountet die eingetragenen File-Systeme im Lese-Schreib-Modus

4. Das X-Window-System:

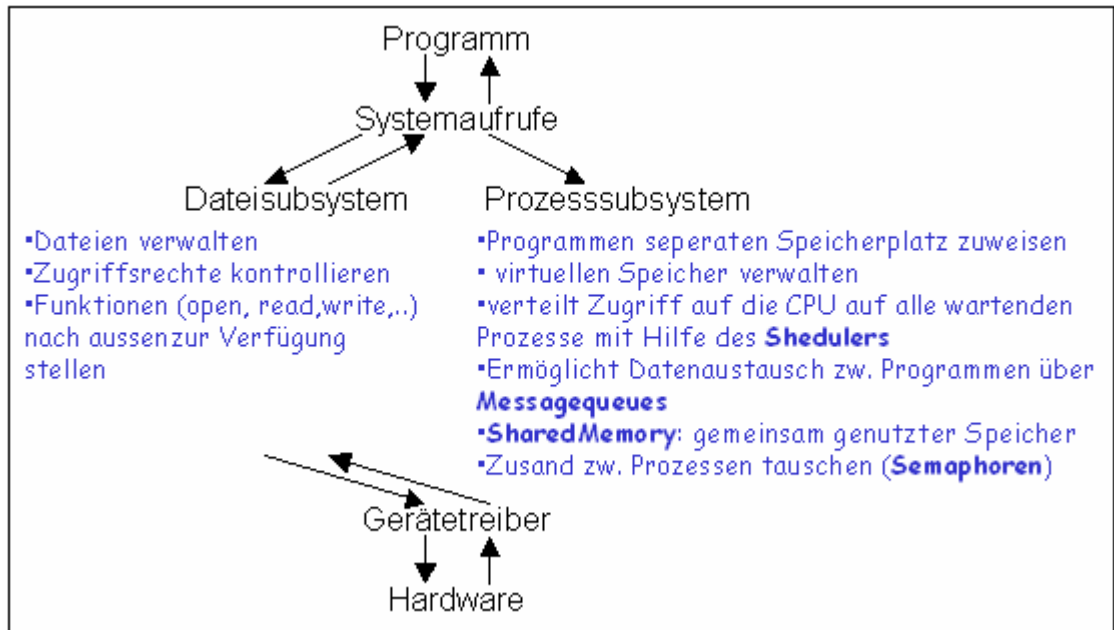
- Für die grafische Darstellung
- In verschiedenen Schichten aufgebaut
 - X (Basis): (X11R6, XFree) Grundlage für die Grafikfähigkeit des Systems
 - macht einfache grafische Funktionen ausführbar (Linien zeichnen, Text an einer bestimmten Stelle ausgeben, ...)
 - Windowmanager: (fvwm2, afterStep, openlook, ...)
 - Fenster in der Größe ändern, in Vorder-/Hintergrund stellen,...
 - Desktop (KDE, Gnome,...)
 - Startbuttons, Papierkörbe, Copy-Paste-Funktion,...
 - Netzwerkfähig (grafisches System kann von anderen Rechnern ausgeführt/geöffnet/... werden)

XServer	XClient
Programm, das die Grafikfähigkeit zu Verfügung stellt → X (Grafikkartentreiber)	Grafisches Programm, das die Dienste eines XServers benötigt, um angezeigt werden zu können z.B. xclock
Startet und steuert den XClient und zeigt die Ausgabe lokal an	Wird von remote (einem anderen Rechner) gestartet und die Ausgabe wird remote angezeigt
5. host (Kommando, das die Zugriffe auf den XServer verwaltet) 6. xhost + 143.53.17 +: freigeben 143...: eingeschränkt auf ... 7. Telnet-Verbindung aufbauen 8. > oneko –display eigeneIP → Ausg. auf eig. Rechner	XClient muss installiert sein (XClient-Programm: oneko)

- /usr/X11R6/bin/startX
 - startet den XServer
 - führt ~/.xinitrc aus
 - → startet min 1 XClient
 - Zeile: exec \$WINDOWMANAGER (startet den Windowmanager)
 - Letzter Eintrag - danach darf nichts mehr stehen, da nach Beenden des letzten Eintrages der XServer runterfährt (z.B. falls xclock letzter Eintrag wäre und man diese dann beendet, wird auch der XServer runtergefahren)

6. Hardware (Kernel, Treiber, Module):

- Kernel:
 - Hauptspeicherverwaltung
 - Zugriff auf Peripheriegeräte
 - Verwaltung des Plattenbedarfs
 - Programm-/Prozessverwaltung
 - Verwaltung der Zugriffsrechte
- Treiber:



- Module:
 - 3c59x.c (Treiber in C) → 3c59x.o (übersetzter Treiber)
 - /lib/modules/2.6.5-7.111.19-default/kernel/net → installierte Treiber
 - /etc/modules.conf
 - sagt welche Treiber geladen werden
 - sagt welche Treiber an welches Gerät gebunden sind
 - alles, was mit einem alias bezeichnet ist, wird auch geladen
 - Kommando **lsmod** (gibt an, welche Module geladen sind, mit Platzbedarf, wie oft sie genutzt werden und von welchen Modulen sie abhängen)