Musterlösung

Probeklausur
24.01.2018

1	2	3	4								Σ

Die Bearbeitungszeit der Probeklausur ist 45 Minuten; für die richtige Klausur haben Sie 90 Minuten Zeit. Entsprechend hat diese Probeklausur reduzierten Umfang (50 %). Bitte bearbeiten Sie alle Aufgaben. Es sind insgesamt 45 Punkte zu erreichen (richtige Klausur: 90 Punkte).

Tipp: Lesen Sie zunächst alle Aufgaben durch und entscheiden Sie, welche Fragen Sie am leichtesten beantworten können; starten Sie dann mit diesen Aufgaben.

Viel Erfolg!

1. Dateisysteme / Benutzerverwaltung (17 / 45 Punkte)

a) Neue Benutzer (2 P.)

Mit welchem Kommando fügen Sie einen neuen Benutzer `testuser` zum Linux-System hinzu und erzeugen dabei auch gleichzeitig ein Home-Verzeichnis am Standardort (`/home/testuser`)? Geben Sie das volle Kommando an.

useradd -m testuser

b) Platte vorbereiten (6 P.)

Auf einer Platte `sdc` mit klassischer Partitionstabelle (MBR) haben Sie die Partitionen `sdc1` (Typ: Linux Swap), `sdc2` (Typ: Linux) und `sdc3` (Typ: Linux) erzeugt.

- `sdc1` soll als Swap-Partition verwendet werden.
- `sdc2` soll ein Ext4-Dateisystem verwenden und nach `/test` gemountet werden – der Ordner `/test` existiert noch nicht.
- `sdc3` soll ein Ext4-Dateisystem verwenden und nach `/test/home` gemountet werden.

Geben Sie die Befehle an, die nötig sind, um die drei Partitionen zu formatieren (bzw. als Swap-Bereich einzurichten) und dann auch zu verwenden. Der erste Befehl (`mkdir /test`) ist schon vorgegeben. Sie können davon ausgehen, bereits mit Root-Rechten zu arbeiten.

1. `mkdir /test`
2. mkswap /dev/sdc1
3. mkfs.ext4 /dev/sdc2
4. mkfs.ext4 /dev/sdc3
5. swapon /dev/sdc1
6. mount /dev/sdc2 /test
7. mkdir /test/home
8. mount /dev/sdc3 /test/home

c) Tools zum Formatieren und Prüfen (4 P.)

(i) Welche Funktion hat das Programm `fsck.ext3`? Durch welchen Mechanismus in welcher Konfigurationsdatei wird sein automatischer Aufruf beim Systemstart gesteuert?

Ext3-FS überprüfen ~~anlegen~~. letzte Spalte in /etc/fstab

(ii) Mit welchem Programm können Sie ein FAT-Dateisystem erzeugen?

mkfs.vfat (oder: mkfs.msdos)

d) Partitionen und fdisk (5 P.)

(i) Bei Verwendung des Kommandos `n` (new) in `fdisk` erscheint folgende Rückfrage, welcher Partitionstyp erzeugt werden soll:

```
Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
```

Info in Klammern zeigt nicht jedes fdisk an!

Gibt es auf dieser Platte bereits a) primäre, b) erweiterte, c) logische Partitionen? (Die möglichen Anworten zu a, b, c sind jeweils „ja", „nein" oder „unbekannt".)

a) ~~unbekannt~~/ja b) ja c) unbekannt

(ii) Mit dem `fdisk`-Kommando `t` ändern Sie den Typ einer Partition. Was bewirkt eine solche Typ-Änderung?

Ändern der Typ-Kennung in der Part.-Tabelle

2. Shell-Nutzung (8 / 45 Punkte)

a) Betrachten Sie die folgenden Aussagen. Geben Sie zu jeder Aussage an, ob sie wahr oder falsch ist, und geben Sie eine kurze Begründung (in Stichworten).

- Alle in der Shell durch `VARNAME=wert` definierten Variablen sind auch in Kind-Shells sichtbar.
  falsch – nur die exportierten
- Der Befehl `rm -r DIRNAME` löscht rekursiv das Verzeichnis `DIRNAME` (mit allen enthaltenen Dateien und Unterordnern).
  richtig – r = rekursiv
- Mit dem Befehl `help PROGNAME` rufen Sie die Handbuchseite zu einem Programm auf.
  falsch – man PROGNAME
- Mit dem Umleitungsoperator `2>` leiten Sie die Standardausgabe in eine Datei um.
  falsch – 2> = Fehlerausgabe

3. Shell-Programmierung (7 / 45 Punkte)

a) Implementieren Sie eine For-Schleife (welche die Variable `i` von 3 bis 7 laufen lässt und im Rumpf der Schleife jeweils nur `echo $i` ausführt) auf drei unterschiedliche Arten. **(3 P.)**

```
for i in $(seq 3 7); do
  echo $i
done
```

```
for (( i=3; i<=7; i++)); do
  echo $i
done
```

```
i=3
while [ $i -ne 8 ]; do
  echo $i
  i=$[i+1]
done
```

b) Welche Funktion hat der Befehl `continue` in einer Schleife? **(1 P.)**

aktuellen Schleifendurchlauf abbrechen
(Schleife wird aber fortgesetzt)

c) Mit `inotifywait` können Sie ein Verzeichnis auf Änderungen überwachen; über die Option `-m` starten Sie das Programm im Monitor-Modus. Erläutern Sie, warum der Betrieb im Monitor-Modus notwendig ist, wenn Sie in einem Skript auf *alle* Ereignisse im überwachten Ordner reagieren wollen. Beschreiben Sie dazu einen problematischen Ablauf bei Verwendung ohne `-m`. **(3 P.)**

Ohne Monitor-Betrieb: Schleife mit inotifywait und Verarbeitung. Während der Verarb. kann ein Event auftreten, den man dann verpasst.

4. Shell-Programmierung: Beispielprogramm (13 / 45 Punkte)

Betrachten Sie das folgende Shell-Skript `raetsel.sh`:

```
#!/bin/bash
i=$1; k=0
while true; do
  echo $i
  ((k += i))
  ((i += $2))
  [[ $i -gt $3 ]] && break
done
echo s=$k
```

a) Simulieren Sie einen Skript-Lauf von `./raetsel 3 2 10` notieren Sie alle Ausgaben, die das Programm beim Aufruf mit diesen Argumenten erzeugt. Geben Sie außerdem an, welchen Inhalt die Variable `$i` am Ende hat. **(1 P.)**

b) Beschreiben Sie die Funktionalität des Skripts. **(2 P.)**

c) Ersetzen Sie die Zeile mit && durch einen besser verständlichen If-Then-Else-Block. **(1 P.)**

d) Erstellen Sie eine angepasste Version des Skripts, welche $1, $2 und $3 in sinnvoll benannte Variablen einliest und dann nur noch diese Variablen verwendet. Durch den Einsatz der Variablen soll das Skript ohne weitere Kommentierung direkt verständlich sein. **(3 P.)**

e) Das Skript funktioniert nur wie gewünscht, wenn der mittlere Parameter positiv ist. Geben Sie an, welches Problem auftritt, wenn Sie einen negativen Parameter verwenden. **(2 P.)**

f) Lösen Sie das Problem aus Aufgabe **e)**, indem Sie eine angepasste Version des Skripts bereitstellen, die auch mit negativen mittleren Parametern sinnvoll umgeht. **(4 P.)**

a) 3, 5, 7, 9, s=24

b) raetsel a b c ⇒ 1) Schleife von a bis c, Schrittweite b
2) Ausgabe s= Summe der vorher ausgeg. Werte

c)
```
if [[ $i -gt $3 ]]; then
  break
fi
```

d)
```
zähler=$1; inkrement=$2; ende=$3
summe=0
while true; do
  echo $zähler
  (( summe += zähler ))
  (( zähler += inkrement ))
  [[ $zähler -gt $ende ]] && break
done
echo s=$summe
```

e) Abbruchbedingung wird nie erfüllt

f)
```
[[ $inkrement -gt 0 &&
   $zähler -gt $ende ]] && break
[[ $inkrement -lt 0 &&
   $zähler -lt $ende ]] && break
```