Abteilung für INFORMATIK

HTL ST. PÖLTEN

Titel: Labor 4

Klasse: 3BHIF

Name: Haiden

Gruppe: 01

Aufgabe: 27.11.2019 Abgabe:10.12.2019

Inhaltsverzeichnis

1 Dateisystem mounten

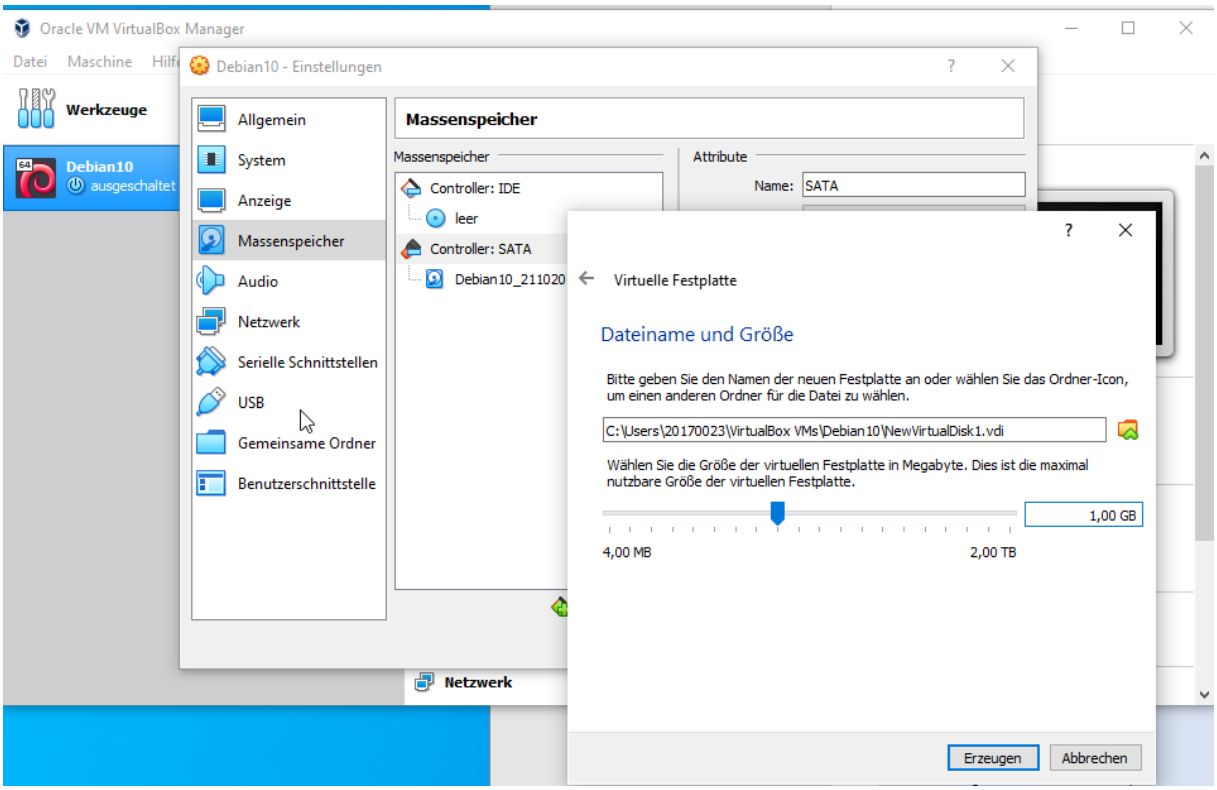
1.1 Welche Bedeutung hat die Datei /etc/fstab?

In der Datei /etc/fstab gibt man alle Datenträger an, die beim Systemstart, d.h. wenn Linux startet, gemount werden sollen. Dabei muss man verschiedene Optionen einzeln angeben.

- UUID: Als erstes muss man die UUID des Dateisystems angeben, das ist ein Unique Identifier für die Festplatte. Dieser ist einzigartig und dient dazu, um die Festplatte identifizieren zu können. Er muss als erstes in der FSTAB Datei angegeben werden.
- Mount Point: Als nächstes muss man einen Mount Point angeben, das kann ein Ordner auf jeder beliebigen Festplatte sein. So kann z.B. in seinem Home-Folder die Datenpartition einhängen und hat so einen einfachen Zugriff auf die Partition.
- Type: Hier gibt man den Typ des Dateisystems an, also ob es ext4, NTFS, usw. ist.
- Options: Hier kann man noch spezielle Optionen angeben, z.B. ob auf diesem Datenträger gelesen und geschrieben werden darf, ob darauf Hardware-Dateien vorkommen kommen dürfen, oder ob ein normaler User das Verzeichnis von einem normalen User gemountet werden darf oder nicht.

1.2 Legen Sie eine zusätzliche Festplatte (1GB) an

Zuerst wird, wie im Screenshot dargestellt, eine Festplatte für die virtuelle Maschine erzeugt, die man dann in der Debian VM benutzen kann.



Mit dem Befehl `sudo fdisk /dev/<FESTPLATTE>` kann man auf einer Festplatte eine neue Partition erstellen. Dabei wird, falls die Festplatte noch komplett leer, also ohne Partitionstabelle und ähnliches war, je nach Größe eine sogenannte MBR oder GUID Partitionstabelle erstellt.

```
schueler@Debian10nvs:~$ sudo fdisk /dev/sdb


Welcome to fdisk (util-linux 2.33.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


The old ext4 signature will be removed by a write command.


Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x62b3f4a7.


Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default
2097151):


Created a new partition 1 of type 'Linux' and of size 1023 MiB.


Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.


schueler@Debian10nvs:~$ sudo fdisk -l
#Disk /dev/sda: 16 GiB, 17179869184 bytes, 33554432 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xbb484e51

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 31457279 31455232   15G 83 Linux
/dev/sda2       31459326 33552383  2093058 1022M  5 Extended
/dev/sda5       31459328 33552383  2093056 1022M 82 Linux swap /
Solaris



Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x62b3f4a7

Device     Boot Start     End Sectors  Size Id Type
/dev/sdb1        2048 2097151 2095104 1023M 83 Linux
schueler@Debian10nvs:~$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 261888 4k blocks and 65536 inodes
Filesystem UUID: 7879facd-dd7a-40b9-8523-424bb6748e0d
Superblock backups stored on blocks:
     32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
schueler@Debian10nvs:~$ sudo mount /dev/sdb1 /mnt/backup/
```

1.3 Ist das Verzeichnis für den User schueler (niklas) schreibbar?

```
niklas@niklasUbuntuEoan:/mnt$ cd backup

niklas@niklasUbuntuEoan:/mnt/backup$ echo "Hello World">>hello.txt

bash: hello.txt: Keine Berechtigung
```

Wie man an der obigen Ausgabe sehen kann, hat der User schueler (in dem Fall niklas da andere Maschine), keine Berechtigungen eine Datei auf die Festplatte zu schreiben.

1.4 Machen sie es schreibbar/nicht schreibbar.

Um das Verzeichnis auch für normale User schreibbar zu machen, führt man folgendes aus:

```
niklas@niklasUbuntuEoan:/mnt$ sudo chown -R niklas:niklas
/mnt/backup/

niklas@niklasUbuntuEoan:/mnt$ cd backup/

niklas@niklasUbuntuEoan:/mnt/backup$ echo "Hello World">>text.txt

niklas@niklasUbuntuEoan:/mnt/backup$ cat text.txt

Hello World
```

Wie man sieht, wird bei dem echo Befehl keine Fehlermeldung ausgegeben und der Inhalt wurde erfolgreich in die Datei geschrieben.

Um das Ganze rückgängig zu machen, macht man einfach folgendes:

```
niklas@niklasUbuntuEoan:/mnt$ sudo chown -R root:root /mnt/backup/

niklas@niklasUbuntuEoan:/mnt$ echo "Hello World">>text2.txt

bash: text2.txt: Keine Berechtigung
```

Dies entzieht dem normalen User jegliche Rechte und nur mehr der root-Benutzer hat Zugriff darauf.

1.5 Erstellen Sie einen Eintrag in fstab

Als erstes wird die UUID der Partition bzw. Festplatte eingetragen, welche man sich mit blkid als root-Nutzer ermitteln kann. Als nächstes der Mount-Punkt, in dem Fall /mnt/backup. Dann wird das Dateisystem der Partition angegeben, darauf folgend ein paar Optionen, in dem Fall dürfen normale User das Laufwerk ein und aushängen, es darf gelesen und geschrieben werden und es dürfen Programme von der Partition ausgeführt werden.

```
UUID=1c1e9de8-6784-4ae1-a3de-12dddc4cf4f0       /mnt/backup     ext4
user,rw,exec 0 0
```

2 Prozessverwaltung

2.1 Erzeugen Sie ein Programm cmd (sh, C, ...), dass alle 2 Minuten „Hallo PID“ ausgibt

```
schueler@Debian10nvs:~$ cat cmd.sh
#!/bin/bash
#Schleife, die alle 2 Sekunden die Prozess-ID des Prozess ausgibt
while true
do
   echo $$
   sleep 2
done
```

2.2 Starten Sie CMD

```
schueler@Debian10nvs:~$ ./cmd.sh
6483
6483
…
```

2.3 Stoppen (Strg-Z) Sie cmd

Das Programm wird bei diesem Vorgang nicht gestoppt, es wird bloß in eine Art Ruhezustand versetzt, d.h. dass sich das Programm immer noch im Hauptspeicher befindet, allerdings nicht gerade keine CPU Zeit verbraucht.

```
^Z
[1]+  Angehalten                ./cmd.sh
schueler@Debian10nvs:~$
```

Man erkennt daran, dass ein Programm in den Ruhezustand versetzt wurde, weil die Shell sagt, dass es angehalten wurde, wie bei der obigen Terminal – Ausgabe.

2.4 Schicken Sie das Programm in den Hintergrund

Mit dem Befehl `bg <JOBNUMMER>` kann man einen Prozess in den Hintergrund schicken. Das heißt, er blockiert nicht die Shell, sodass man weiter Befehle eintippen kann. Schließt man allerdings die Shell, in der das Programm gerade ausgeführt wird, beendet.

```
schueler@Debian10nvs:~$ bg 1
[1]+ ./cmd.sh &
schueler@Debian10nvs:~$ 6483
6483
6483
6483
```

2.5 Holen Sie das Programm in den Vordergrund

Mit dem Befehl `fg <JOBNUMMER>` holt man das aktuell im Hintergrund oder angehaltene Programm in den Vordergrund, dabei wird die Shell blockiert und nur die Ausgabe ausgegeben.

```
niklas@niklasUbuntuEoan:~$ ./cmd.sh
2759
2759
^Z
[1]+  Angehalten              ./cmd.sh
niklas@niklasUbuntuEoan:~$ fg 1
./cmd.sh
2759
2759
2759
2759
```

2.6 Beenden Sie das Programm

Mit der Tastenkombination STRG + C wird das Programm beendet. Das wird in der Shell durch ein ^C angezeigt.

```
2759
2759
2759
^C
niklas@niklasUbuntuEoan:~$
```

2.7 Starten Sie das Programm im Hintergrund

Um einen Prozess im Hintergrund zu starten, ruft man das Programm ganz normal auf und schreibt dahinter ein käufmännisches &. Damit läuft das Programm im Hintergrund. Ein Einsatzzweck hierfür wären Daemons, z.B. Webserver, die im Hintergrund weiterlaufen sollen, auch wenn die Shell zu ist.

```
niklas@niklasUbuntuEoan:~$ ./cmd.sh &
[1] 2900
```

2.8 Zeigen Sie die Jobs an

Mit dem Befehl `jobs` zeigt man alle Jobs an, die derzeit im Hintergrund laufen bzw. Angehalten sind, wie im Beispiel zu sehen ist.

```
niklas@niklasUbuntuEoan:~$ jobs
[1]+  Angehalten              ./cmd.sh
```

2.9 Anzeigen der Prozessliste

Mit dem Befehl `ps` kann man sich die Prozessliste anzeigen lassen. Dabei gibt es verschiedene Optionen, die die Ausgabe erweitern.

Führt man ps ohne jegliche Parameter aus, werden nur Prozesse angezeigt, die sich in der Shell, in der es aufgerufen wurde, abspielen.

```
niklas@niklasUbuntuEoan:~$ ps
   PID TTY          TIME CMD
  3230 pts/2    00:00:00 bash
  3237 pts/2    00:00:00 ps
```

Mit dem `-a` Parameter kann man sich die laufenden Prozesse des gesamten Benutzers anzeigen lassen (vgl. obige Ausgabe).

```
niklas@niklasUbuntuEoan:~$ ps -a
   PID TTY          TIME CMD
  1429 tty2     00:00:09 Xorg
  1438 tty2     00:00:00 gnome-session-b
  2900 pts/0    00:00:00 cmd.sh
  2983 pts/1    00:00:00 cmd.sh
  2984 pts/1    00:00:00 sleep
  3138 pts/1    00:00:00 man
  3148 pts/1    00:00:00 pager
  3242 pts/0    00:00:00 sleep
  3243 pts/2    00:00:00 ps
```

Um eine detailliertere Ausgabe zu bekommen, kann man den Parameter `-u` verwenden. Man bekommt ein paar Extra-Informationen, wie z.B. die Startzeit des Prozesses oder die verbrauchte Prozessorzeit angezeigt.

```
niklas@niklasUbuntuEoan:~$ ps -u
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
niklas     1426  0.0  0.1 172696  6568 tty2     Ssl+ 22:16   0:00
/usr/lib/gdm3/
niklas     1429  0.8  1.9 293148 79824 tty2     Sl+  22:16   0:10
/usr/lib/xorg/
niklas     1438  0.0  0.3 200864 15996 tty2     Sl+  22:16   0:00
/usr/lib/gnome
```

Mit dem `-x` Parameter kriegt man alle Benutzerprozesse angezeigt, das heißt, alle Prozesse, die sich im User-Space befinden.

```
niklas@niklasUbuntuEoan:~$ ps -x
   PID TTY      STAT   TIME COMMAND
  1401 ?        Ss     0:00 /lib/systemd/systemd --user
  1402 ?        S      0:00 (sd-pam)
  1414 ?        S<sl   0:01 /usr/bin/pulseaudio --daemonize=no
  1417 ?        Sl     0:00 /usr/bin/gnome-keyring-daemon --
daemonize --login
  1422 ?        Ss     0:00 /usr/bin/dbus-daemon --session --
address=systemd: --nofork --nopidfile --systemd-activation --syslog-
only
  1426 tty2     Ssl+   0:00 /usr/lib/gdm3/gdm-x-session --run-script
env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd
--session=ubuntu
```

2.10 Starten Sie Cmd mit einer Priorität von 5

Man kann die Priorität von Prozessen nicht selbst bestimmen, man kann nur einen sog. Nice-Wert angeben, d.h., es ist ein Wunsch an das Betriebssystem, wie ein Prozess mit Priorität 5 behandelt zu werden, wobei das Betriebssystem dies nicht erfüllen muss.

```
niklas@niklasUbuntuEoan:~$ sudo nice -5 ./cmd.sh
[sudo] Passwort für niklas:
3937
…
niklas@niklasUbuntuEoan:~$ ps ax -o pid,ni,cmd
…
  3937   5 /bin/bash ./cmd.sh
…
```

Mit dem ps Befehl kann man sich durch angeben von spezifischen Spalten den Nice-Wert holen, und wie man hier sehen kann, wurde das Skript mit einem Nice-Wert von 5 gestartet.

2.11 Starten Sie den Prozess mit Priorität auf -15

```
niklas@niklasUbuntuEoan:~$ sudo nice --15 ./cmd.sh
[sudo] Passwort für niklas:
4207
niklas@niklasUbuntuEoan:~$ ps ax -o pid,ni,cmd
4207 -15 /bin/bash ./cmd.sh
```

2.12 Starten Sie cmd mit nohup (Ausgabe umleiten)

Mit dem Befehl `nohup <BEFEHL>` kann man einen Befehl starten, damit er nach Ab und Anmeldung weiterläuft. Dabei schreibt der Prozess in eine Datei namens nohup.out.

```
niklas@niklasUbuntuEoan:~$ nohup ./cmd.sh
```

ABMELDEN & ANMELDEN