Linux Administation Grundlagen

(mit Ingo Wichmann und Dominik George (nik@velocitux.de))

Author Robin Heydkamp

Tipps und Tricks

- Um eine Befehlszeile mit # Auszukommentieren `ESC + #`
- Um den letzten Parameter zu kopieren `Alt + .`
- Programm `etckeeper`, Macht aus dem /etc/ ein Repository welches man dann entsprechend reverten könnte.

Spannende Webseite

- https://lab.linuxhotel.de/customers/
- https://flathub.org/
- https://snapcraft.io/
- https://hub.docker.com/
- https://wiki.lab.linuxhotel.de/doku.php

nik@velocitux.de

Tag 1

Verwendete Befehle:

```
xargs echo useradd-s /bin/bash -m < users
xargs -n1 echo useradd-s /bin/bash -m < users
watch 'ls -ld /tmp/*-backup?'s
```

Xargs um Argumente zu übergeben, `-n1` um das mit jeder Zeile einzeln durchzuführen.

Nutzer anlegen:

```
xargs -n1 echo useradd -s /bin/bash -m < users
```

Passwörter setzen:

```
    pwgen -B 8 10 >> users
    vim users
    chpasswd < users
```

> ssh robin@ubuntu1.lxht.de
> `mkdir subdir`
> cd `.+Alt` (Zum verwenden des letzten Arguments aus der Histroy

Grundlegende Befehle
SSH Key erstellen und auf VHost kopieren. Testen
Grundlegende Datei Verwaltung. Datein Erstellen und kopieren.

Output Kanäle der Bash

Kanal	Bezeichnung	Steuerzeichen	Alternative (an Datei anhängen)
0	STDIN	<	
1	STDOUT	>	>>
2	STDERR	2>	2>>

Beispiele:

```
    wc -l < /etc/shadow
```

Die Pipe | (Befehle Verknüpfen)

Der Inhalt vor der Pipe wird dem Befehl nach der Pipe entgegen geworfen.

```
cut -d : -f 7 /etc/passwd | sort | uniq -c | sort -n
```

über `mkfifo` kann eine Pipe "benannt" werden. Dann steht diese Pipe als "temporärer Speicher" zur verfügugn. One time Read Only.

TAR Archive

Erstellen

- `tar -c ~/* > share.tar` um ein Archiv zu erstellen \
- `tar -cC /usr share/ | gzip > share.tar.gz`
- `tar -cC /usr share/ | zstd > share.tar.zst` Nimmt den neuen "Facebook" Komprimierungsgrad
- `tar -caC /usr share/ -f share.tar.zst`

Statt `>` lässt sich auch `-f` (File) verwenden.

Validieren

- `tar -t < share.tar` um ein Archiv zu "öffnen" bzw. auszulesen welche Dateien vorhanden sind. \
- `du -sh share.gz`

Dekomprimieren

- `zcat share.tar.gz | tar -x` Nimmt das Tar und streamt es "dekomprimiert" weiter. In dem Fall an `tar -x` womit es dann entpackt wird.
- `tar -cC /usr share/ > share.tar.gz` Datei so dekomprimiert
- `tar -xf share.tar.gz`
- `tar -xf share.tar.zst` Archiv automatisch dekomprimieren. -f kann WIRKLICH nur Dateien dekomprimieren. KEINE Streams.

Datein auf einem Server Archivieren, komprimieren und anschließend übertragen.

```
ssh robin@ubuntu1.lxht.de 'tar -cC /usr share/ | zstd' | pv >
ubuntu1_usr_share.tar.zst
```

Wichtig: Durch die Anführungszeichen beim Tar und zstd Befehl werden die Datein erst AUF DEM SERVER komprimiert und Archiviert. DANN erst Übertragen.

Task: Archiv entschlüsseln und auf dem Server im Home-Order hinterlegen.

Datein öffnen, Dann auf den Server kopieren und zu guter letzt natürlich noch dekomprimnieren und "ent-Archivieren" `cat ubuntu1_usr_share.tar.zst | ssh robin@ubuntu1.lxht.de 'zstdcat | tar -x'`

Task: Backup Pushen

Den Lokalen Ordner /usr/share komprimieren und archivieren. Anschließend auf den Server pushen

```
tar -cC /usr/share doc/ | zstd | ssh robin@ubuntu1.lxht.de 'cat >
doc.tar.zst'
```

- `tar -cC /usr/share doc/` Archiv erstellen
- `zstd` Arcvhiv komprimieren
- `ssh robin@ubuntu1.lxht.de` Verbindung zum Server aufbauen
- `'cat > doc.tar.zst'` DatenStream auf dem Server annehmen und dann den STDOUT in eine Datei kopieren

Task: Restore from Backup

```
ssh robin@ubuntu1.lxht.de cat doc.tar.zst | zstdcat | tar -xC doc/
```

- `ssh robin@ubuntu1.lxht.de` - Verbindung zum Server
- `cat doc.tar.zst` - Komprimiertes Archiv "öffnen"
- `|` - Bash-Sonderzeichen zum beenden der SSH Verbindung und annehmen des "STDOUT" lokal
- `zstdcat` - geöffnetes Archiv dekomprimieren
- `tar -xC doc/` - geöffnetes, dekomprimiertes Archiv entpacken nach Ordner doc/

RSYNC

- `rsynch -a /usr/share share`
- `rsynch -a --del /usr/share share` - `--del` damit gelöschte Dateien in der Quelle auch im Ziel gelöscht werden.
- `rsynch -a --del /usr/share robin@ubuntu01.lxht.de:share`
- `rsynch -az --del /usr/share robin@ubuntu01.lxht.de:share`

Task: Von der entfernten Maschine den gesicherten Ordner Doc ins Home verschieben

```
rsync -az --del robin@ubuntu1.lxht.de:share ./doc
```

Benutzermanagment

> Folgende Befehle laufen quasi alle unter Sudo `sudo -i`

```
touch /tmp/vorher
useradd -m -s /bin/bash robin
find / -xdev -newer /tmp/vorher -ls
```

Befehl	Parameter	Erklärung
find	/	Sucht alle Datein im Ordner "/"

Befehl	Parameter	Erklärung
-xdev		Ausschluss von "Devices", Mount-Points werden ausgeschlossen
- newer	/tmp/vorher	Neuer Daten als Vorher
-ls		Formatierung

`(sudo) useradd -m -s /bin/bash robin`

Wichtige Benutzer werden unter "/etc/passwd/ oder /etc/shadow/ gespeichert

`(sudo) passwd -e robin` *Erklärung*

- `passwd` zum erstellen eines Passwortes.
- `-e` damit er nach dem ersten Login ein neues Passwort verlangt,
- `robin` ist der Benutzer

- `(sudo) userdel robin` - Account löschen (nicht empfohlen da Neue Benutzer eventuell Dateien vom gelöschtem Benutzer bekommen kann)
- `(sudo) passwd -l robin` - Account "Sperren", es wird vor das Passwort in der Shadow Datei ein ! geschrieben. Dadurch ist der Login nicht möglich.
- `(sudo) passwd -d robin` - "Löscht" das Passwort des Benutzers, ACHTUNG dabei wird der Account "Entsperrt"
- `(sudo) passwd -u robin` - Reaktiviert einen Benutzer (entfernt das ! im Hash der Shadows)
- `cat /etc/passwd` - Listet nur die lokal hinterlegten Benutzer auf.
- `getend passwd` - Listet alle "echten" Benutzer im System auf. Auch die, die über ein LDAP oder ActiveDirectory kommen.

Dateibereichtigungen

Befehl	Parameter	Erklärung
r	4	Lesbar, schreibbar und ausführbar für alle Benutzer.
w	2	Wie 1., aber ohne Schreibrecht für andere Benutzer (z.B. bei PHP-Skript).
x	1	Wie 1., aber ohne Schreibrecht für andere Benutzer und Benutzer der eigenen Gruppe.

Berechtigung	Numerisch
Lesen + Schreiben + Ausführen	7
Lesen + Schreiben	6
Schreiben + Ausführen	5
Lesen	4
Lesen + Ausführen	3
Schreiben	2

Berechtigung	Numerisch
Ausführen	1

umask - Defaults für Dateien und Ornder

Defaultbereichtignugen werden über die `umask` definiert. Standardmäßig lautet diese "`0002`". Für Dateien wird die Umask vom Standard "666" (für jede Kategorie einzeln) abgezogen. Für Dateien wird die Umask von 777 abgezogen.

Task: Unmask umsetzen

Neue Dateien sollten rw- r-- -w- Neue Verzeichnisse sollen rwx r-x -w- bekommen

```
umask 0024
umask 0025
```

Tag 2

vewendete Befehle

- `chattr +i /etc/hosts` - change Attribut, Immutable, Unverundbar. Nicht editierbar. Nicht löschbar. Auch von Root nicht mehr, Rückgängig mit `-i`
-
-

Projekt Gruppenverzeichniss

- `id <User>` gibt die Informationen über den aktuellen Benutzer aus.
- `groupadd <groupname>` Erzeuge eine Gruppe
- `chgrp <Group> <Ordner>` / `chown :<Group> <Ordner>` Ornder einer Gruppe zuordnen
- `chmod g+w <ProjektOrdner>` Ändere die Berechtigung für Gruppen
- `gpasswd -a <User> <Group>` Weise User einer Gruppe zu

Wenn ein User Schreibrechte auf ein ORDNER hat, darf er in diesem Ornder auch Dateien neu anlegen oder löschen. Auch wenn diese Dateien eigentlich einem anderem User oder einer anderen Gruppe zugeordnet sind.\

- `pkill -u praktikant`
- `pkill -kill -u praktikant`

sendet ein "Kill" Signal an alle Prozesse des Praktikanten. Dies führt auch zu einem Logout z.B. bei einer SSH Verbindung. Nötig um wirklich die neuen Gruppen/Benutzer-Berechtigungen zu aktivieren. Laufende Prozesse werden sonst nicht mit den aktualisierungen versorgt.

Besondere Rechte

	suid	sgid	Sticky
Dateien	s	s	(n/a)
Verzeichnisse	(n/a)	s	t

- Kleine Buchstaben (s,t) bedeuten dass das dahinterliegende Bit ebenfalls gesetzt ist, Also z.b. Ausführen. \
- Große Buchstaben (S,T) bedueten dass das dahinterliegende Bit nicht gesetzt ist. Also z.b. kein Ausführen gesetzt ist.

Suid Bit (suid)

Suid Bit -> In den Bereichtungen, rws. s zählt quasi Als Ausführen und "suid". Andere Benutzer dürfen diese Datei dann als "Besitzer" ausführen. Beispiel passwd:

```
robin@ubuntu1:~$ type passwd
passwd is /usr/bin/passwd
robin@ubuntu1:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 72056 May 30  2024 /usr/bin/passwd
```

Passwd darf also als "root" ausgeführt werden von Benutzer die der Gruppe "root" oder "Other" zu gehören mit ihren entsprechenden Berechtigungen (r+x, r+x).

Alle Dateien finden die das Suid-Bit haben.
`find / -xdev -user root -perm /u+s -ls`

> Berechtigungen würden nach einem Update wieder ihre, vom entiwckler gewünschten Rechte bekommen. > Ein entferntes Suid-Bit z.B. würde dann also wieder erscheinen. Um Berechtigungen Updatesicher zu > setzen kann man die Berechtigungen über dpkg dauerhaft setzen. \ `dpkg-statoverride --update --add root root 0755 /bin/ping`

Set-Group-Id (sgid)

Durch das sgid wird einem Verzeichniss mitgeteilt das alle Verzeichnisse und Dateien die in diesem Verzeichniss erstellt werden automatisch der selben Gruppe zugewiesen werden die das "Parten-Verzeichniss". Also das Verzeichniss, dem das sgid zugvewiesen wurde. Untere Verzeichnisse erhalten ebenfalls das sgid, entsprechend werden hier auch die Gruppen weitervererbt.

Sticky Bit

Mit dem Sticky Bit im Verzeichnis-Berechtigung lässt sich das "Write" leicht einschränken. Alle Benutzer die der Gruppe zugeorndet sind (beim Ornder `/tmp` wären es "Other"), dürfen entsprehcne DAtein und Ornder anlegen. Andere benutzer dürfen diese allerdings nicht löschen auch wenn sie im Ordner und auf den Dateien schreib Rechte hätten.

Task: Dateirechte testen

Eine Datei anlegen mit konkreten Rechten:

- `mkdir -m 777 /srv/open`\
- `chmod 356 /srv/open/datei356`

```
#!/bin/bash
echo executable
```

Die Datei hat also folgende Rechte:

	Berechtigung
Besitzer	- w x
Gruppe	r - x
Andere	r w -

Wie können wir nun auf die Datei zugreifen:

	Mitlglied	**nicht Mitglied**
Besitzer	- w -	- w -
Nicht besitzer	r - x	r w -

Warum kann der Besitzer die Datei nicht ausführen? Im Fehlt das "read" Attribut, Wenn er nichts lesen kann, kann er die Datei auch nicht ausführen. Solange es keine Binary-Datei ist.

Capabilities

zum Wiki

Für Prozesse lassen sich die Fähigkeiten mit anderen Prozessen zu interagieren beschränken. Z.b. dürfte ein Prozess nicht auf die Drucker zugreifen. Oder anderen Prozesse sehen, beenden (kill).
Zum Einsehen der Capabilitys über:

`systemd-analyze security <Programm.Service>`

DNS Schreibt in die /etc/resolv.conf dhclient -sf /usr/bin/env

Sudo / Root

brauchbare Befehle:

- `su - root` - "switch user"
- `su -c id - root`
- `sudo -u robin -l` - Recht als Benutzer Robin ausführen.
- `sudo -l` - Sudo-Befehle die ich aktuell ausführen darf anzeigen.
- `sudo -l` - Sudo-Befehle die ich aktuell ausführen darf anzeigen.
-

Sudoers Datei

Zum Wiki

Beispiel Eintrag; `%sudo ALL=(All:All) ALL`

- `%sudo` = GRUPPE Sudo
- `ALL` = Welcher Rechner
- `(ALL:ALL)` = Als welcher Benutzer, Als welche Gruppe darf der Befehl ausgeführt werden
- `ALL` = Welcher Befehl darf ausgeführt werden

Ändern der Sudoers Datei immer mit `visudio -f <Datei>` (Default wird genommen, wenn nicht `-f` eine Datei angegeben wurde)

Task MTU ändern und Seperaten Benutzer erlauben diese Änderung auch durchzüfhren.

- `ip -c link show dev enp4s0`
- `ip -c link set mtu 1480 dev enp4s0`

```
DATEI: /etc/sudoers.d/netzadmin.tmp
robin ALL=(root) /usr/sbin/ip -c link set mtu 1480 dev enp4s0 # MTU auf
1480 Setzen
robin ALL=(root) /usr/sbin/ip -c link set mtu 1500 dev enp4s0 # MTU auf
1500 setzen
$sudo ALL=(ALL) NOPASSWD: ALL  # Sudo Befehle ohne Passwort checken!
praktikant ALL=(root) /usr/bin/vim /etc/hostname # Praktikant erlauben
"etc/hostname" mit Vim zu öffnen. ACHTUNG ULTRA GEFÄHRLICH da Vim auch
andere Dateien öffnen kann, besser wäre
praktikant ALL=(root) sudoedit /etc/hostname
```

Jobs und Prozesse

- Strg + Z -> Prozess stoppen
- Strg + C -> Prozess abbrechen

bg/fg gestoppten Prozess im Hintergrund oder Vordergrund weiterlaufen lassen.

Prozesse mit & am Ende werden automatisch im Hintergrund gestartet.

Task BG und FG

- `sleep 10 &` wartet 10 Sekunden "im Hintergrund"
- `wait` wartet bis alle Jobs abgearbeitet sind
- `top` - Prozess Liste "Table of Processes"
  - `h` um zwischen threads und Jobs zu wechseln
  - `k` Um Prozesse zu "killen"
  - `r` für "renice", neuen Nice Wert setzen, (-20 , +19)

- `grep aux`- Alle Prozesse mit allen Benutzern anzeigen
- `pgrep -u <Benutzer>` - Alle Prozesse des Benutzers

- `pgrep gzip | wc -l` - Alle Prozesse von gzip anzeigen bzw. anzeigen wie viele Ergebnisse kommen (Word Count - Line)
- `pgrep -la gzip` Zeigt alle Prozess-IDs von GZIP Prozessen.
- `pkill gzip` - Beende alle Prozesse von GZIP

- `lsof` list open files
- `lsof -ac bash -au nutzer52 +D /home/nutzer52/` - Zeige alle Bash Prozesse die einem Nutzer gehören und im Home Ornder stattfinden. Wichtig ist das "-a" bei den Parameter, dadruch werden die Parameter mit "AND" Verknüpft. Ansosnten werden diese mit "OR" verknüpft.

Speicherverwaltung

- `free - h` - Zeige Ram Speicher Verwaltung
- `lsblk` - list block devices

```
nvme0n1     259:0    0 465,8G  0 disk
├─nvme0n1p1 259:1    0 977,6M  0 part /boot
├─nvme0n1p2 259:2    0   500M  0 part /boot/efi
├─nvme0n1p3 259:3    0 413,7G  0 part
├─nvme0n1p4 259:4    0    20G  0 part /
└─nvme0n1p5 259:5    0  30,6G  0 part [SWAP]
```

p1-p5 sind die Partitionen

Spaß mit fdisk

Mit `lsblk` können wir die Festplatten und aktuelle Partitionen einsehen. Jetzt wollen wir die Partitionen neu gestallten.

- Starten von fdisk mit `fdisk`
- `m` für Manual
- `n` für Neue Partition
- `w` für "schreiben" und Beenden.

> Partitionen könnten zum Beispiel zum Sichern von Daten verwendet werden. Um Fesplatten zu kopieren eignen sich natürlich Tools wie rsynch. Zum initialem Kopieren lässt sich aber auch `dd` verwenden.

Dateisysteme

`mkfs.` - Mögliche bekannte Dateisysteme anzeigen. **mkfs.ext4** ist hierbei das, unter Linux, am weitesten Verbreitem. xfs ist das Standard-Dateisystem unter RedHat-Linux. Es ist für High-Perfomance Systeme geeignet.

Verwendung:

- `lsblk` - Alle Block-Devices anzeigen
- `mkfs.ext4 /dev/<Device Bezeichung / Partition>` - ext4-Dateisystem einrichten

- `blkid /dev/nvme0n1p10` - UUID auslesen.
- `mount -m /dev/nvme0n1p10 /mnt/ext4` - `-m` Um automatisch den Mount-Point zu erzeugen (Ornder), Quelle - Mount-Point
- `tail -n 1 /proc/mounts >> /etc/fstab` -> Kopiert die ersten Informationen in die FSTAB. Wir brauchen trotzdem noch die UUID in der fstab. Sollte dann So aussehen:
    - > UUID=8b632fcc-3ad5-4db8-b347-61780175fbb4 /mnt/ext4 ext4 rw,nosuid,nodev 0 0
- `mount -av` - Alle Mounts aus der FSTAB einhä#ngen.
- `df -h` - schauen wo das Laufwerk eingebunden wurde.

Tag 3

LVM - Logical Volume Managment

- Logical Volume - kann größer oder kleiner werden.
- Volume Group - Kollektiv von einem Volume,
- Physical Volume - Festplatten, SSDs, USB-Stick

Snapshots können von Logical Volumes angefertig werden. Snapshots sind nicht direkt ein Backup, aber es ist kopierbar/spiegelbar und man kann daraus ein Backup erstellen.

Snapshot ist erstmal einfach nur eine weitere Ansicht auf die selben Daten. Für den Snapshot werden zusätzliche Blöcke im Volume resserviert. WEnn dann Daten "geschrieben" werden, werden diese nicht in dem bisherigen Blöcken geschrieben, sondern in dem, vom Snapshot reservierten Bereich umgelenkt. Dieses Verfahren nennt man "`Copy on Write`"

Man macht ein Snapshot von dem aktuellen Stand der daten weil diese dann Konsistenz zu einem speziellem Zeitpunk sind. Während eines potenziellen Kopiervorggang, z.B. Via dd oder cp oder rsync können die Daten dann weitergeschrieben werden, während wir die Daten vom Snapshot sichern.

Task: aus einer Partition ein LVS erzeugen und damit spielen

Zum Wiki

2 Partitionen erscheinen die als LVM getaged sind.

```
nvme0n1                   259:0    0 465,8G  0 disk
├─nvme0n1p1               259:1    0 977,6M  0 part /boot
├─nvme0n1p2               259:2    0   500M  0 part /boot/efi
├─nvme0n1p4               259:3    0    20G  0 part /
├─nvme0n1p5               259:4    0  30,6G  0 part [SWAP]
├─nvme0n1p10              259:5    0    80G  0 part /mnt/ext4
├─nvme0n1p11              259:6    0    80G  0 part /var/log
├─nvme0n1p12              259:7    0    80G  0 part
│ └─robins_system-lv_test 252:0    0     5G  0 lvm
├─nvme0n1p13              259:8    0    80G  0 part
├─nvme0n1p14              259:9    0    80G  0 part
└─nvme0n1p20              259:11   0  13,7G  0 part
```

Linux Dateiverzeichniss

Zum Wiki

- `/boot/` - Alles zum Boooten, Grub-Boot-Loader, vm_linuz sind die Kernel-Datein
- `/dev/` - Devices, Maus, Festplatten
- `/etc/` - Konfigurations-Dateien
- `/home/` - Home-Verzeichnisse aller User
- `/lib/` - Libarys (usr/lib)
- `/mnt/` - mount-folder.
- `/media/` - Speichermedien, USB-Sticks, autmatisch gemountet
- `/proc/` - Prozess-IDs und entsprechende Meta-Daten, ist eigentlich leer, wird nur beim auslesen temporör gefüllt
    - `cat cpuinfo` - Gibt Informationen über die CPU auslastung
- `/run/` -
- `/sys/` - Analog zu /proc/ allerdings etwas sauberer und geordneter.
- `/usr/` - Unique Static Resources, Programm Datein,
    - `bin/` - Binarys
    - `sbin/` - systemBinarys
    - `libs/` - Libarys
    - `local/` - Skripte und Anwendungen die wir selber anlegen und nicht aus einem Repository oder sowas kommen. Daten hingegen lieber auf `/srv/`
    - `share` -
- `/opt/` - Verzeichniss für *schlecht* portierte Software...
- `/var/` - Variabele Datein
    - `tmp/` - Wird nicht beim rebooten gelöscht.
- `/tmp/` - Temporäre Dateien die beim rebooten gelöscht werden.
- `/srv/` - Nur für selber angelegt Daten - quasi wie /usr/local/ aber nur für Skripte sondern für Daten

Booten

- UEFI + NVRAM (Non-Volatile-Ram) -> shim -> Bootloader (GRUB - Grand Universal Bootloader)
- BIOS -> MBR (MasterBootRecord) -> Bootloader

UEFI kann nur mit dem Bootloader kommunizieren über einen Zertifiziertes Programm von Windows. Damit das möglichst günstig war, und Linux nicht den kompletten GRUB lizensieren lassen wollten, haben sie shim entwickelt der, zertifiziert, nichts anderes tut ausser den GRUB zu starten.

`efibootmgr -v`

Bootloader

Binary-Programm um weitere Binarys nachzuladen. Zuerst "kennenlernen" der grafischen Oberfläche, des Dateisystems, und setzen der primar Partition mit der er arbeiten soll. Bsp. 'hd0,gpt1'. Anschließend wird vom Root-Ordner "vmlinuz" gestartet. Hier lassen sich auch Parmeter mitgeben. Sowas wie `init=/bin/bash` um direkt eine root-Bash zu starten. Wir booten allerdings nur im "read-Only"-Modus.

Der Befehl `update-grub` rennt gegen die Datein im `/etc/grub.d/`. Hier werden die Skripte nacheinander ausgeführt und pushen ihre Ausgabe anschließend nach `/boot/grub/grub.cfg`. Einfache Änderungen können auch direkt nach `/boot/grub/grub.cfg` kopiert werden, die Änderungen werden allerdings beim nächsten Kernel-Update wieder überschrieben.

Kernel

Verwendete Befehle

- `update-initramfs`
- `update-initramfs -u -k all`
- `lsinitramfs`

Wenn wir z.B. ein Image von einem Rechner auf einer anderen Maschine installieren wollen werden dort im init-rd die entsprechenden Konfig Datein hinterlegt... Auf anderen Maschinen kann das aber zu Problemen führen da die Hardware konfiguration anders ist. Über `update-initramfs -u -k all` könnten wir dann die initrd neu schreiben lassen mit der aktuellen Hardware.

Die initrd (initial ramdisk) hat quasi die Aufgabe die SystemPartition verfügbar zu machen. Quasi `mount /dev/nvme0np4 /`. Dazu werden Datein aus dem Dateisystem in den Ram geschreiben.

Was ist der Kernel? Sämtliche Kommunikation mit der Hardware läuft über den Kernel. Programme greifen nicht direkt auf die Datein zu, sondern bitten den Kernel die entsprechenden Signale zu senden damit diese die Datein verarbeiten können.

- `modinfo amdgpu` - Inforamtionen über Treiber sammeln
- `lsmod`
- `modprobe` - Befehl zum laden von Treibern

Zeigt Informationen an mit der der Treiber der Hardware (in dem Falle eine amd-GPU) geladen werden kann. Parameter können angepasst werden. Aber auch verwendet Firmware kann angezegit werden oder auch nachinstalliert werden.

- `dpkg -S raven_gpu_info` - Sucht nach Informationen zu raven_gpu_info (eine Firmware für amdgpu)

Treiber erkennen

- `lspci` - Alle Treiber für die Schnittstellen ausgeben
- `lspci -s 00:04.0` - Informationen über eine konkrete Schnittstelle rausfinden
- `lspci -s 05:00.0 -k` - Treiber informationen anzeigen
- `lsusb`
- `ls /sys/bus/...`
- `dmesg -Tw`

```
modprobe -r dummy # Dummy Treiber Stoppen
modprobe dummy numdummies=5 # Dummy Mit 5 Dummies starten
ip -c link show # Wir sehen 5 Netzwerk Dummies
```

> ## Fun Fact
>
> Unter Ubuntu ist nicht ALLES eine Datei. Netzwerkkarten z.B. nicht.

Task

Dummy Netzwerkinterface dummy0 in intern0 umbenennen

Zum Wiki

Mit UDev lassen sich Regel für die interaktion mit Hardware erstellen. Z.B. das beim einstecken eines USB-Sticks automatisch ein Backup Programm gestartet wird. Oder das eine Netzwerkkarte mit spezieller Konfiguration (Name, IP-Adresse) eingebunden wird.

System-D

(dejy-vu!)

Der erste Prozess ist immer "/sbin/init". Immer mit der PID 1 und vom user root. Über `systemctl status apache2.service` können die Informationen über services angezeigt werden.

Services können enabled oder disabled werden. Services die Enabled sind, lassen sich beim Start direkt booten. Streng genommen werden Services die enabled sind lediglich in dem Ordner: `/etc/systemd/system/multi-user.target.wants/` hinterlegt.

- `systemctl set-default multi-user.target` - Grafische Oberfläche deaktiviert. Nach dem Booten sind wir automatisch in einer Shell.
- `systemctl set-default graphical.target`
- `systemctl list-dependencies graphical.target`

- `systemctl stautus zeigt degraded`
- `systemctl list-units --failed`

Wir finden raus das nginx nicht läuft da Apache2 den Port bereits belegt hat.

- `systemctl disabale --now nginx`
- `systemctl reset-failed`

VIM - Its Magic!

`vimtutor` - startet den VIM Tutor

- `h j k l` - Navigieren , links, unten, oben, rechts
- `x` - Aktuelles Zeichen löschen
- `u` - undo . Rückgängig machen

- `i` - um in den Input-Modus zu wechseln
- `A` - um in den Input, am ENDE der Zeile zu springen.
- `d` - Löschen :
  - `dd` - die komplette Zeile
  - `dw` - bis zum Beginn des nächsten Wortes OHNE dessen erstes Zeichen.
  - `e` - zum Ende des aktuellen Wortes MIT dessen letztem Zeichen.
  - `d$` - (JAA Dollar!) Löscht bis zum Ende der Zeile.
- `2w` Anfang des 2 Wortest
- `4e` Ende des vierten Wortes
- `0` anfang der Zeile
- `d2w` - Lösche 2 Wörter
- `/` - suchen, eintippen
  - `n` - nächstes Ergebnis
  - `N` - vorheriges Ergebnis
- `g` - Zum Anfang der Datei
- `G` - Zum Ende der Datei
- `.` - Wiederhole die letzte Eingabe

Tag 4

Vim: Nach den Änderungen vergessen die Datei mit Sudo zu öffnen. Was kann ich machen? `:w !sudo tee /etc/hostname` - write SUDO tee (Verbindet die Asugabe in eine Datei) Dateinamen den man gerade editieren will.

Softwareverwaltung

Microsoft

- Bezugsquelle
  - Original?
  - Vertrauenswürdig
- Installationsprogramm
  - Was
  - Wohin
  - Abhänigkeigten
    - Bundled
    - automatische nachinstallieren
    - manuell nachinstallieren
  - Deinstallationsprogramm
    - Was
    - Von Wo
  - Updates
    - Eingebauter Update Manager

Task Apt benutzen

1. Finde ein Spiel mit apt, das wie Super Mario funktioniert.

    - `apt search supertux`
2. instaliere das Paket
    - `apt install supertixkart`
3. Was verrät euch `apt show` über das Paket
4. Deinstalliere das Paket wieder.
    - `apt remove supertuxkart`
    - `apt autoremove supertuxkart` - Löscht supertuxkart UND alle Abhänigkeiten

Beim Installieren fällt auf das apt automatisch updates und upgrades macht. Anschließend werden alle abhänigkeiten geprüft und installiert. Das Spiel selbst wird auch installiert.

> `apt show` liefert ergebnisse über den Hersteller, abhänige Pakete, Version, Maintainer und Original-Maintainer, Homepage, Section (Kategorie). Unterschied zwischen Recomend und Suggest:
> Recommend wird standardmäßig mitinstalliert. Über ein parameter lässt sich das aber auch deaktivieren.
> Suggest ist umgekehrt. Standardmäßig wirds nicht mitinstalliert.

Tasks für dpkg

- Was war in dem SuperTuxKart den alles drin?
  `dpkg -L supertuxkart` Zeigt wohin ein Programm installiert wurde, inkl. aller unterdatein.
- `.deb` Datein sind quasi komprimierte Programm-Dateien. mit `dpkg -i` könnte man auch .deb Dateien entpacken bzw. "installieren".
- `dpgk` kümmert sich um alles was schon auf dem System ist. Also welche Programme installiert sind oder wenn .dep Dateien bereits auf dem Rechenr sind. Es kann aber keine "unbekannten" Dateien "finden" oder verwalten. Dafür ist `apt` da. Auch kann es keine Abhänigkeiten installieren.
- `apt remove supertuxkart` löscht lediglich das Programm. Nicht die Abhänigkeiten. Diese könnten anschließend mit `apt autoremove` entfernt werden. Dabei werden alle automatisch installierten Pakete (recommend/suggest) entfernt.
- `apt list` Zeigt alle verfügbaren Pakete an die installiert werden könnten.

> Bei Red Hat
>
> - `dnf` kümmert sich um Repositoy
> - `rpm` lokales System

Ein Repository einbinden

Beispiel PostgreSQL.

```
# Import the repository signing key:
sudo apt install curl ca-certificates # Curl und Ca-Certifiacates
installieren um den PGP-Public-Key runterzuladen und zu verifizieren.
sudo install -d /usr/share/postgresql-common/pgdg # Ordner erstellen.
sudo curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --
fail https://www.postgresql.org/media/keys/ACCC4CF8.asc # PGP-public-Key
```

```
runterladen

# Create the repository configuration file:
. /etc/os-release # ? touch? `source`, läd quasi die os-release variablen
in die Shell. Die Variabele Version_Codename wird in diesem beispiel mit
"nobel" gefüllt
sudo sh -c "echo 'deb [signed-by=/usr/share/postgresql-
common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $VERSION_CODENAME-pgdg main' >
/etc/apt/sources.list.d/pgdg.list" # Wir schreiben in die sources.list, das
wir der Webseite postgresql "vertrauen" und wir uns mit dem runtergeladenem
Key authentifizieren können/wollen.

# Update the package lists:
sudo apt update

# Install the latest version of PostgreSQL:
# If you want a specific version, use 'postgresql-17' or similar instead of
'postgresql'
sudo apt -y install postgresql
```

Signaturen

Dinge durch Verschlüsselung Absicher -> Kryptografie.

- Symetrische Verschlüsselung
  - Der Schlüssel ist zum ver UND entschlüsseln identisch.
- Asymmetrische Verfahren
  - Schlüssel zum ver und entschlüsseln sind unterschiedlich.
  - Nachrichten können mit einem Public Key verschlüsselt werden
  - Nachrichten können mit dem Private Key entschlüsselt werden. Ausschließlich mit dem Private
- Signieren
  - Bob verschlüsselt mit seinem PRIVATE Key eine Nachricht.
  - Wenn ich sie mit dem Public Key Entschlüsseln kann, weiß ich, das diese wirklich von Bob kommt
  - Die Message ist also mit dem Private Key signiert.

> **Beispiel**: Mirrors können auch via https aufgerufen werden und sind dann erstmal "gültig" sofern das SSH-Zertifikat gültig ist. Um aber zu prüfen ob auch das runtergeladene Paket, dem entspricht was der eigentliche Entwickler entwickelt hat, werden die Pakete zusätzlich Signiert. Daruch können Pakete auch von Mirrors verifiziert werden.

In welchen Branch wird Software einsortiert?

	supported	**unsupported**
open source	main	univers
non open source	restricted	multiverse

Wie entsteht open source Software unter Debian?

Binary Dateien bereit stellen. Der Hersteller ist verantwortlich das die Lizenz wirklich open source ist. Libarys müssen alle open Source bleiben. Telemetrie muss standardmäßig deaktiviert sein. Im Zweifel muss die Software angepasst werden.

Wenn das Paket dann fertig ist, kann ich das Paket in ein Repository hochladen welches sich "`debian unstable`" nennt. Hier kann die Software dann ausgiebig getestet werden. Für die Produktiv-Umgebugnen wird aber nur "`debain stable`" verwendet. In der Regel verlässt ein Paket nach 5 Tagen `debian unstable` in den Branch `debian testing`. Ab Testing sollte keine neuen Features entwickelt werden, lediglich Bugs gefixxt.

Speicher aufräumen und Ubuntu Upgrade machen

`du -hxd1 /usr/ | sort -rh` - Feststellen welches Ornder unter /usr/ wie viel Platz verbraucht `apt do-release upgrade` - Release upgrade ausführen. Führt eventuell zu einer Meldung das

Docker installieren

```
apt install docker.io # docker installieren
docker run -it bash # bash Container starten
cat /etc/os-release # beispiel Befehle
exit  # docker Container wieder beenden.
```

Mit Docker ein Mini Debian einrichten

`debootstrap trixie /root/minidebian` -

- `debootstrap` = Befehl,
- `trixie` = Version von Debian,
- `root/minidebian` = Pfad

> `chroot /root/minidebian/ /bin/bash` - Neue Shell starten bin /root/minidebian als Root-Verzeichnis. \ `chroot` sagt nur das ein Programm mit einem neuem Pfad als Root Verzeichniss starten sollen. In diesem Falle starten wir das Programm `/bin/bash`.

Mit Docker einen eigenen httpd-Container erstellen.

```
mkdir -p /srv/docker/httpd/htdocs # htdocs erzeugen
cd /srv/docker/httpd  # ordner wechseln
```

`vim /srv/docker/httpd/htdocs/index.html` # index.html erzeugen.

```
    <html>
      <head>
        <title>Hello World!</title>
      </head>
      <body>
```

```
        Hello World
      </body>
    </html>
```

Ein Docker Container erstellen mit dynamischen Konfiguration.

```
docker run -d --name hello-httpd -p 8888:80 -v
'/srv/docker/httpd/htdocs:/usr/local/apache2/htdocs/' httpd:2.4
```

- `docker run` - Docker Starten
- `--name hello-httpd` - Container Name
- `-p 8888:80` - Portforwarding mitgeben.
- `-v '/srv/docker/httpd/htdocs:/usr/local/apache2/htdocs/'` - Dateipfad
- `httpd:2.4` - Verwendetes Image, Kommt in diesem Falle von Docker selbst. Zu suchen über: Docker Hub

> Unterschied zwischen docker run und docker build mit docker build wird lediglich ein Image erzeugt. Mit Docker Run ein Container gestartet. Zum prüfen ob Container via docker laufen kann man mit `docker ps` den Status aller Container einsehen.

Backups

Warum Backups? Um sich vor Angriffen zu schützen oder Datenverlust (Plattenausfall) zu schützen.

Full-Backups

Welche Ordner lohnen sich für ein Full-Backup

- `boot`
- `etc` - Konfigurationsdatein
- `home` - Daten
- `opt` - Programme
- `srv` - Nutzdaten von Serverdiensten
- `usr/local/` - Nutzdaten vom System
- `var`
- `mount --bind / /mnt/system` - Alle Datein unter / auch auf /mnt/system mounten
- `dpkg --get-selection > /tmp/pakete` - Installierten Pakete anzeigen und nach "tmp/pakete" sicher. NICHT EMPFOHLEN!
- `dpkg --set-selection < /tmp/pakete` - Pakete wieder installieren NICHT EMPFOHLEN!

Alternative für VOLLEs Backup
`dd if=/dev/nvme0n1 of=/mnt/backup/datentrager.img bs=4M conv=sparse`

- `if` = Input File
- `of` = Output File
- `bs` = Block-Size
- `conv=sparse` = Sparse Dateien komprimieren (?)

Snapshots werden verwendet damit die Daten weiter bearbeitet werden können und weiterhin konsistente Daten haben können fürn die Dauer des Snapshots.

partielles Backup mit rsynch

Damit nicht alle Datein permanent kopiert und gesichert werden, kann man mit rsynch die Verzeichnisse einfach synchronisieren. So wird unnötiger Datenverkehr vermieden.

```
rsync -aSH --acls --xattrs --numeric-ids --del /mnt/system/
root@notebook51:/mnt/ext4/robin
```

- `rysnch` - Programm
- `-aSH` -a = Archive Mode, -S = turn sequences of nulls into sparse blocks, -h = preserve hard links
- `--acls` = ACL's Berechtigungen
- `--xattrs` = Preserve extende Attributes,
- `--numeric-ids` = don't map uid/gid values by user/group name

Tag 5

Topic-List

- Distributionen
- Netzwerk
- ssh
- Login

Distributionen

Eine Kurze Übersicht über einige Linux Distribiutionen

- Debian
  - Ubuntu
  - Mint
  - Knoppix
  - Raspian
- Fedora
  - CentOS
    - RedHat Enterprise
  - Open SUSE
    - Open SUSE Thumbleweed
      - SUSE Enterprise
      - Open SUSE leap
- Archlinux

    - SteamOs
- Alpine
- Gentos - wird immer selber kompiliert, besteht lediglich aus Build-Skripten
    - ChromeOs
- Android

Netzwerk

- `ping linuxhotel.de` - Sendet ein "Echo-Request" an einen Server.
- `ping -4 linuxhotel.de` - Verwendet IPv4.

Was passiert dabei?

Zuerst DNS aufschlüsselung. Dazu `getent hosts linuxhotel.de`, Wenn wir die IP haben müssen wir das Gateway fragen ob wir im Intranet oder ins Internet gehen müssen. Mit `ip route get 49.12.11.242` erkennen wir das die Route nicht nur über den Gateway sondern auch über ein DNS läuft. Der DNS kann uns die passende MAC-Adresse bereitstellen (das ist etwas unsicher). Die MAC Adresse ist die, vom Router bzw. Gateway. Das Gateway entpackt das "Ethernet-Frame", entdeckt weil die MAC ja passt. Er erkennt das die IP aber nicht die richtige ist. also fängt er von vorne an und versucht das Paket weiter zu leiten mit der selben IP aber der nächsten MAC (vom DNS, rausgefunden über `ip route get 49.12.11.242`). TraceRoute pingt den Zielhost mit einem Paket mit minimaler TTL (steigernd, nach jeder Fehlermeldung). Jeder Zwischenhost sendet dann eine Fehlermeldung zurück und so kann TraceRoute dann erkennen mit welchen Servern er kommuniziert.

Der Unterschied zwischen einem Router und einem "host" ist lediglich, das ein Router verscuht Paket, die nicht für ihn bestimmt sind, weiterzuleiten.

Netzwerkbefehle

- `host linuxhotel.de`
- `getent hosts linuxhotel.de` - präferierte Adresse finden
- `getent ahosts linuxhotel.de` - alle Adressen finden
- `nslookup linuxhotel.de` -
- `ip route get 49.12.11.242` - Route zu 9.9.9.9 finden
- `ip neighbor show` / `mtr 9.9.9.9` (mtr-tiny) - bekannte Link-Layer-Adressen anzeigen
- `ip a` / `ip n` - Netzwerkkarten-Informationen anzeigen.
- `sysctl net.ipv4.ip_forward` - ergibt das wird IP Pakete "forwarden" können.
- `ip address add dev enp4s0 192.168.29.252/24` - Wir vergeben uns eine weitere IP-Adresse. (zum löschen `del`). Dies ist nur Temporär (bist zum nächsten booten)
- `ip route add 9.9.9.9/32 via 192.168.29.251` - Wir Fügen eine Route hinzu und sagen das die IP 9.9.9.9 über den Host erreichbar sein soll (das ist die Kiste von Nik)
- `cat /etc/nsswitch.conf`
- `resolvectl` - DNS-Konfiguration mit systemd.-resolved
- `nmtui` - Grafische Oberfläche
- `ip address show` - eigene Adressen auflisten

einen Router bauen

Auf dem Router:

```
sysctl net.ipv4.ip_forward=1
ip address add dev enp4s0 192.168.29.251/24
iptables -P FORWARD ACCEPT
iptables -t nat -I POSTROUTING -j MASQUERADE
```

Auf den Hosts:

```
ip address add dev enp4s0 192.168.29.xxx/24
ip route add 9.9.9.9/32 via 192.168.29.251
```

Netzwerkkonfigurationsdateien:

Datei	Funktion
/etc/nsswitch.conf	Namensdatenbanken
etc/resolve.conf	DNS-Server#
gai.conf	IPv4, IPv6

SSH

SSH steht für secure shell. Eine Shell die auf einem anderem Rechner läuft. Viele Tools basieren auf ssh bzw. nutzen SSH. Ansible, rsynch z.B.. SSH möchte sich nicht darauf verlassen das das Netzwerk sicher und sauber ist. Es möchte eine weitere Verifizierung für den Host haben. Dazu nutzt SSH den sogenanntent "Fingerprint". Grundsätzlich schickt uns der Server eine Signierte Nachricht und seinen Public-Key. Mit dem Public-Key können wir diese Nachricht entschlüsseln.

`ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key.pub` - Hier steht mein Public Key. Der selbe Key, den SSH bekommt wenn eine anderen Maschine versucht auf mich zu connecten. Diesen könnte ich jetzt über andere Methoden (Telefon, E-Mail) übertragen bzw. verifizieren.

```
ssh nutzer07@notebook07 # Befehl
The authenticity of host 'notebook07 (192.168.1.207)' can`t be established.
ED25519 key fingerprint is
SHA256:aEZjOtbJDRqC6Wwkev3pqa2TKl0/yRRjGoySYcIfETo. # Public Key des Hosts
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

task

Überkreuz einloggen auf dem Server des jeweils anderen.

Zum Wiki

```
ssh-keygen -t ed25519 -C "Kommentar"
ssh-copy-id <nutzerdeshosts>@<hostname>
```

> **~/.ssh/authorized_keys**
>
> restrict,command="/usr/bin/who" ssh-ed25519 @

SSH-Tunnel

SSH kann auch Netzwerk-Trafic selbst tunneln. `ssh -L 8080:127.0.0.1:80 nutzer51@notebook51` - Wir Tunneln den Port 8080 automatisch auf die IP des Zielservers

Um einen Jump-Server zu benutzen: `ssh -J nutzer07@notebook07,nutzer48@notebook48 nutzer51@notebook51` - "`jump`" zu NB7 -> weiter zu NB48 und das Ziel ist NB51.

Logging

Syslog ist ein Prozess der Logs schreibt. Zwar kann man grundsätzliche Kateogrien eintippen (z.B. Mail) um die Logs zu bündeln. Aber Syslog wird nicht mehr weiter verwend

less auth.log

Beispiel um Failed Password aus der auth.log zu finden. Die 5 IP Adressen sortiert nach den häufigsten Angriffen: `grep "Failed password" auth.log | grep -E -o "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" | sort | uniq -c | head -n 5`

Loggin mit journalctl

Task

Alle Logmeldungen des ssh-Dienstes anzeigen

- `journalctl /usr/sbin/sshd`
- `journalctl -u ssh.service`

alle LogMeldungen ab 14:00 Uhr

- `journalctl -S "Jun 27 14:00:00"`
- `journalctl --since "14:00:00"`

alle LogMeldungen von SSH ab 14:00 Uhr.

- `journalctl -S "14:00:00" /usr/sbin/sshd`

Alle Logmeldungen die mindestens das Level Warning haben

- `journalctl -p 4`

`journalctl` loggt nach Möglichkeit in die Datei: `var/log/journal` Unter Ubuntu ist diese Datei defaultmäßig vorhanden. Unter RedHat z.B. müsste diese manuell eingeloggt werden.

- `apt install systemd/journal-remote`
- `vim /etc/systemd/journal-upload.conf` -> Hier haben wir die IP des Ziel-Hosts eingetragen
- `systemctl enable --now systemd-journal-upload.service` -> starten und enablen
- `systemctl status systemd-journal-upload.service` -> Service prüfen, Bugfixing mit `journalctl -u systemd-journal-upload.service`

Author: Robin Heydkamp