

# Ubuntu: Dies und das...

Basiswissen für Vergessliche. 😊

## Der UNIX-Editor VI

Auf nahezu jedem Unix-Betriebssystem ist der zu Hause. Wichtig ist zu verstehen, dass der VI in verschiedenen Modi läuft. Diese Modi wären:

- der Befehls-Modus
- der Einfüge-Modus
- der Kommandozeilen-Modus.

Folgendes ASCII-Bild aus [Wikipedia](#) sollte es etwas veranschaulichen, wie die Modi aufgerufen werden und wie sie zu verstehen sind:

		Start mit   vi <dateiname> v	
+-----+ -----+		+-----+ -----+	
Kommandozeilen-	<-----	Befehls-Modus	----->
Einfüge-Modus			
Modus	„:“		„i“, „a“,
		Verwendung von:	„o“ usw.
z. B. „wq“, „q!“		yy, p, dd, J	
Normales Editie-			
oder komplexe		usw.	
ren, Pfeiltasten			
Befehle wie			
Bildscrollen usw.			
„Suchen und	[Enter]	(Der vi startet	[Esc]
Ersetzen“	----->	in diesem Modus)	<-----
+-----+ -----+		+-----+ -----+	
Beenden mit		Beenden mit	
v „wq“ oder „x“		v „ZZ“	

## Konvertieren von Zeichensätzen (vim)

```
:set ff=unix # UNIX-Format
:set ff=dos  # DOS-Format
:set ff ?    # zeigt aktuelles Format
```

## Springen innerhalb einer Datei

```
:1 [enter]      # Anfang
:X [enter]      # Springe zu Zeile X
[Shift] + [G]   # Ende
```

## Exkurs: Den Standard-Editor ändern

Mittels des folgenden Befehls kann der Standard-Editor geändert werden:

```
# update-alternatives --config editor
Es gibt 6 Auswahlmöglichkeiten für die Alternative editor (welche
/usr/bin/editor bereitstellen).
```

Auswahl	Pfad	Priorität	Status
-----			
0	/usr/bin/vim.nox	40	automatischer Modus
1	/bin/ed	-100	manueller Modus
2	/bin/nano	40	manueller Modus
3	/usr/bin/mcedit	25	manueller Modus
4	/usr/bin/vim.basic	30	manueller Modus
* 5	/usr/bin/vim.nox	40	manueller Modus
6	/usr/bin/vim.tiny	15	manueller Modus

Drücken Sie die Eingabetaste, um die aktuelle Wahl[\*] beizubehalten, oder geben Sie die Auswahlnummer ein:

## Hostname ändern via CLI

Vorab sei gesagt, dass es in der Regel reicht, die Datei `/etc/hosts` anzupassen und den Server dann neuzustarten.

Es müssen ansonsten zwei Dateien im Ordner `/etc` editiert werden:

- `hostname`
- `hosts`

## Schritt 1: NetBIOS-Name

```
# echo "MeinHostname" > /etc/hostname
```

## Schritt 2: FQDN persistent

[/etc/hosts](#)

```
127.0.0.1 localhost.localdomain localhost
::1 localhost.localdomain localhost
127.0.1.1 MeinHostname MeinHostname.MeineDomain.de
88.77.66.55 MeinHostname MeinHostname.MeineDomain.de
```

## Schritt 3: NetBIOS-Name neu einlesen

```
# hostname -F /etc/hostname
```

Dieser Befehl setzt den Hostnamen zur Laufzeit.

## Wie kann ich mich ohne Passwort auf einem anderen System einloggen?



Wir gehen hier davon aus, dass sowohl Client als auch Server Linux-Maschinen sind bzw. mit OpenSSH arbeiten!

Durch Austausch von Public und Private Keys! Mal angenommen, wir wollen uns von „Client“ zu „Server“ ohne Passwort einloggen können:

```
Client> # ssh-keygen
Client> # ssh-copy-id username@server
Client> # ssh username@server
Last login: Tue Dec  2 10:38:51 2001 from 89.125.78.5
Have a lot of fun...
Server> # less .ssh/authorized_keys
```

Die Tools lauten:

- ssh-keygen → Es erstellt ein Private Keyfile für unseren User, der sich einloggen möchte.
- ssh-copy-id → Es tauscht den Public Key des Servers mit dem Private des Client aus. Es wird eine Art Vertrauensstellung erzeugt.

Wenn der SSH-Dämon einen Nicht-Standard-Port nutzt:

```
ssh-copy-id "user@host" -p 4711
```

Was ist jetzt konkret geschehen? - Wir haben zunächst einen Private Key und einen Public Key für unseren User auf dem Client erstellt (ssh-keygen). Dieser Schritt erzeugt Dateien in /home/username/.ssh wie folgt:

- id\_rsa
- id\_rsa.pub

Der Inhalt der Datei id\_rsa wurde vom Tool ssh-copy-id nach /home/username/.ssh/authorized\_keys auf dem Server kopiert. Des Weiteren wurde der Public Key des Servers in die /home/username/.ssh/known\_hosts aufgenommen. Der Public Keys des Clients ist das Gegenstück zum Private, der jetzt auch auf dem Server liegt. Daran kann der Benutzer eindeutig identifiziert und eingeloggt werden.

## Screen-Sessions

### Sitzung erstellen

```
# screen -S sitzung1
```

### Sitzung wieder aufnehmen

```
# screen -r sitzung1
```

### Sitzung trennen

```
# screen -d sitzung1
```

## Mailserver-Handling

### Mailqueue

Anzeigen:

```
# mailq
```

Erneut abarbeiten:

```
# sendmail -q
```

## Schweizer Taschenmesser für Kopiervorgänge: rsync

r	(copy) recursive
t	(keep) timestamps

L	copy-links
v	verbose
h	human-readable
delete	delete (extraneous files from dest dirs)

```
rsync -rtLvH --stats --log-file=copy.log --partial --delete /var/files/  
/srv/files/
```



Wird an das Quellverzeichnis ein / angehängt, so wie im vorherigen Beispiel, dann kopiert rsync nur den Inhalt nicht aber das Verzeichnis selbst.

## Boot-Partition läuft voll

Oftmals hilft ein Aufräumen mittels Deinstallation alter Kernel-Dateien über

```
apt-get autoremove
```

Bleiben aber immer noch zu viele alte Kernel-Pakete stehen, dann kann sich sich zunächst mal anzeigen lassen, welche Kernel-Versionen auf dem System installiert sind:

```
dpkg --get-architecture | grep ^ii  
ii linux-image-4.15.0-135-generic          4.15.0-135.139 amd64  
Signed kernel image generic  
ii linux-image-4.15.0-136-generic          4.15.0-136.140 amd64  
Signed kernel image generic  
ii linux-image-4.15.0-137-generic          4.15.0-137.141 amd64  
Signed kernel image generic  
ii linux-image-generic                    4.15.0.137.124 amd64  
Generic Linux kernel image
```

Der aktuelle Kernel ist mittels folgendem Befehl zu ermitteln. Dieser darf nicht entfernt werden!

```
uname -a
```

Der Befehl „autoremove“ schlägt auch bei nicht aufgelösten Abhängigkeiten fehl. Wir entfernen aus der `initrd.img` zunächst alte Kernel mittels:

```
update-initramfs -d -k 4.15.0-135-generic
```

Jetzt ist wieder Platz auf `/boot` zum Arbeiten!

Dann muss dieser Kernel noch sauber über die Paketverwaltung entfernt werden:

```
dpkg --purge linux-image-4.15.0-135-generic
```

Der Befehl schlägt unter Umständen fehl, da noch Abhängigkeiten bestehen. Die abhängigen Pakete

müssten dann im Vorfeld entfernt werden, zum Beispiel:

```
dpkg --purge linux-image-4.15.0-135-generic linux-image-extra-4.15.0-135-generic
```

Alle nicht zu Ende installierten Pakete erneut installieren:

```
dpkg --configure -a
```

alternativ geht auch

```
apt-get -f install
```

<https://help.ubuntu.com/community/RemoveOldKernels>

## CHMOD

Berechnungstabelle Oktalsystem:

	<b>Owner</b>	<b>Group</b>	<b>Other</b>
none	0	0	0
read	1	1	1
write	2	2	2
execute	4	4	4

Berechnungsbeispiel:

	<b>Owner</b>	<b>Group</b>	<b>Other</b>
none	-	-	-
read	X	X	X
write	X	-	-
execute	X	X	X
<b>chmod</b>	<b>1+2+4=7</b>	<b>1+4=5</b>	<b>1+4=5</b>

Weitere Infos: <https://ss64.com/bash/chmod.html>

## Einfaches Dateibackup

### Backup-User anlegen

```
useradd backupuser -d /BACKUP/backupuser -M -s /bin/bash
chown -R backupuser.backupuser /BACKUP/backupuser
passwd backupuser
```

# .bashrc

## Aliases

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
    "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -alFh'
alias la='ls -A'
alias l='ls -CF'
```

## Apt-get E: Could not get lock /var/lib/dpkg/lock

Wenn Apticron mal hängen bleibt, dann kann ein manuell angetriggertes „apt-get“ nicht ausgeführt werden. Meist hängt dann ein Prozess namens „apt-get -qq update“.

```
ps aux | grep -i apt
kill -9 82657
```

From:

<https://wiki.sebastianhetzel.net/> - Sebastians IT-Wiki

Permanent link:

<https://wiki.sebastianhetzel.net/ubuntu:diesunddas?rev=1688141157>

Last update: **2023/06/30 18:05**

