Nützliche Befehle & Links

Große Ordner sortiert auflisten

`du -h /dein/direcotory/ | sort -rh | head -n 20`

Auf welchem Port läuft etwas?

```
lsof -i :PORT -S


root@server:/# lsof -i :9201 -S
COMMAND   PID   USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
java    81378 user   21u  IPv6 5091945      0t0  TCP *:9201 (LISTEN)
```

Hostname ändern

Quelle: https://www.tecmint.com/set-hostname-permanently-in-linux/

- mit SystemD: `hostnamectl set-hostname NEW_HOSTNAME`

Zeilennummern herausfinden

Print all lines between two line numbers

awk: https://www.commandlinefu.com/commands/view/9889/print-all-lines-between-two-line-numbers

sed: https://www.commandlinefu.com/commands/view/9890/print-all-lines-between-two-line-numbers

Get line number from pattern

https://unix.stackexchange.com/questions/182015/find-the-line-number-which-contains-the-pattern-using-custom-regex-delimiter

```
grep -n 'pattern' file


Ausgabe:
23: pattern
```

Bash Ausgabe farbig markieren

Wenn man in mehreren Dateien einen bestimmten String austauschen will dann testet man vorher seinen Befehl und möchte sehen ob es funktioniert. Die farbige Markierung der Ausgabe ist dabei sehr hilfreich um die Stellen des geänderten Codes zu finden und überprüfen zu können.

- Stichwort ANSI Escape Codes

Hier ein Beispiel mit dem Texteditor **sed**:

SED - Texteditor

```
sed ''/iteritems/s//`printf "\033[31miteritems->items\033[0m"`/'' {icinga/*.sls,icingaweb/*.sls,apache/*.sls,mariadb/*.sls,php/*.sls}
```

Beispiel Ausgabe Auszug:
{% for module,attr in salt['pillar.get']('modules:enabled').iteritems->items() %}

Festplatte voll

Quelle: https://www.unixe.de/no-space-left-device/

Kopie des Blogbeitrages:

Unser erster Schritt sollte darin bestehen herauszufinden, **welches** `device` eigentlich gemeint ist; ist eine eingehängte reine Datenpartition voll, so muss in der Regel schlicht aufgeräumt und ausgemistet werden. Schwieriger wird es jedoch, wenn das System gar keine Datenpartitionen hat beziehungsweise die `root`-Partition betroffen ist — was nun?

```
$ df -h
Dateisystem            Grösse Benutzt Verf. Verw% Eingehängt auf
/dev/mapper/lda-root    6,4G    6,1G  8,0K  100% /
none                    4,0K       0  4,0K    0% /sys/fs/cgroup
udev                    487M     12K  486M    1% /dev
tmpfs                   100M    568K   99M    1% /run
none                    5,0M       0  5,0M    0% /run/lock
none                    497M       0  497M    0% /run/shm
none                    100M       0  100M    0% /run/user
/dev/mapper/lda-temp    488M    556K  478M    1% /tmp
/dev/mapper/lda-boot    249M     69M  168M   29% /boot
/dev/sdb1                12G    514M   11G    5% /var
```

Ersichtlich ist, dass das Root-File-System zu 100% belegt ist; oft liegt das an Log-Files, die in aller Regel in `/var/log` abgelegt werden, aber wie wir hier sehen können ist `/var` eine eigene Partition, auf der noch jede Menge Platz ist. Was also ballert das System so zu? Fangen wir mal klein an.

Kernel, Header und Pakete

Eine wahre Pest können auf einem Ubuntu-System alte Pakete sein, und insbesondere die `linux-image`-Daten können (in `/usr/src`) eine Menge Platz fressen. In einem ersten Schritt können also nicht mehr benötigte Abhängigkeiten aus dem System entfernt werden:

```
$ apt-get autoremove
$ apt-get clean
```

Anschließend betrachten wir uns näher, welche `linux-images` installiert sind und ob die wirklich alle benötigt werden; `ii` bedeutet, dass das Paket vollständig installiert ist, `rc` zeigt uns, dass das Paket zwar deinstalliert wurde, die Konfigurationen jedoch noch im System herumsausen.

```
$ dpkg -l|grep linux-image
...
rc  linux-image-3.13.0-36-generic       3.13.0-36.63
amd64        Linux kernel image for version 3.13.0 on 64 bit x86 SMP
rc  linux-image-3.13.0-39-generic       3.13.0-39.66
amd64        Linux kernel image for version 3.13.0 on 64 bit x86 SMP
ii  linux-image-3.13.0-52-generic       3.13.0-52.86
amd64        Linux kernel image for version 3.13.0 on 64 bit x86 SMP
ii  linux-image-3.13.0-57-generic       3.13.0-57.95
```

```
amd64        Linux kernel image for version 3.13.0 on 64 bit x86 SMP
rc  linux-image-extra-3.13.0-36-generic 3.13.0-36.63
amd64        Linux kernel extra modules for version 3.13.0 on 64 bit x86 SMP
rc  linux-image-extra-3.13.0-39-generic 3.13.0-39.66
amd64        Linux kernel extra modules for version 3.13.0 on 64 bit x86 SMP
ii  linux-image-extra-3.13.0-52-generic 3.13.0-52.86
amd64        Linux kernel extra modules for version 3.13.0 on 64 bit x86 SMP
ii  linux-image-extra-3.13.0-57-generic 3.13.0-57.95
amd64        Linux kernel extra modules for version 3.13.0 on 64 bit x86 SMP
ii  linux-image-generic                 3.13.0.57.64
amd64        Generic Linux kernel image
```

Die nicht benötigten können nun mit Hilfe des Paketmanagers entfernt werden:

```
$ apt-get remove linux-image-3.13.0-36-generic
...
$ apt-get remove linux-image-3.13.0-39-generic
...
```

In aller Regel wird durch diese Maßnahmen schon erstaunlich viel Platz geschaffen; sollte das noch nicht ausreichen, muss weiter nach der Ursache geforscht werden.

Das größte Paket

Du kannst suchen, welches die größten im System installierten Pakete sind; vielleicht brauchst du das eine oder andere davon ja nicht mehr und kannst es mittels des Paketmanagers entfernen?

```
$ dpkg-query -W -f='${Installed-Size;8}\t${Status;1}\t${Package}\n' \
> | grep -v "\sd\s" \
> | sort -n \
> | cut -f1,3-
...
   61937    linux-headers-3.13.0-73
   61938    linux-headers-3.13.0-74
   61938    linux-headers-3.13.0-76
   74304    libreoffice-common
  113706    libreoffice-core
```

```
...
```

Große Ordner (z.B. > 1GB) aufspüren

Welcher ist der fetteste Ordner im System? Du kannst von `/` aus iterieren und große Datenmengen systematisch aufspüren.

```
$ cd /
$ du -h --max-depth=1 / | grep '[0-9]G\>'
4.4G	/var
4.0G	/usr
9.5G	/
```

Große Files (z.B. > 1GB) aufspüren

Oftmals sind auch einzelne immer riesiger werdende Files die Übeltäter, die das System verstopfen — zum Beispiel Log-Files, die aus irgendwelchen Gründen nicht nach `/var/log` geschrieben werden. Auch nach ihnen kann systematisch gefahndet werden:

```
$ find / -name '*' -size +1G
```

Out of Inodes

Richtig fies wird es, wenn *eigentlich* noch hinreichend viel freier Platz auf dem Device ist, das System dennoch `No space left on device` meldet; das kann zum Beispiel an einer richtig großen Anzahl ganz winzig kleiner (oder gar leerer) Dateien liegen. Speicherplatz ist dann vorhanden, die verfügbaren Inodes sind jedoch aufgebraucht.

```
$ df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/xvda            2080768 2080768       0  100% /
tmpfs                  92187       3   92184    1% /lib/init/rw
varrun                 92187      38   92149    1% /var/run
varlock                92187       4   92183    1% /var/lock
udev                   92187    4404   87783    5% /dev
```

```
tmpfs                  92187       1   92186    1% /dev/shm
```

Hier wird es ersichtlich: auf / sind die Inodes aufgebraucht. Wir können die Suche nach den Verursachern damit beginnen, dass wir im File-System nach einer unüblich hohen Anzahl an Dateien innerhalb eines Ordners suchen. Auch hier wird sich von / aus in Richtung des Ordners iteriert.

```
$ for i in /* ; do echo $i ; find $i | wc -l ; done
```

Jetzt können die Dateien (gegebenenfalls archiviert und dann) gelöscht werden; die Zahl verfügbarer Inodes ist nun signifikant höher als zuvor.

SSH

Port Weiterleitung

```
ssh -L Port_lokal:localhost:Port_remote username@remoteserver.com
Beispiel:
ssh -L 443:localhost:443 username@remoteserver.com
```

Revision #17
Created 14 August 2018 11:45:14 by David
Updated 20 October 2022 06:36:04 by David