Allgemeines

- Anzahl der Cores/Prozessoren herausfinden
- Lustige Sachen mit Linux
- Zeilen in Felder zerlegen
- Zeichenfolgen von Sondertasten ermitteln
- Harten Reset durchführen
- Arbeiten mit sed
- Reguläre Ausdrücke (regex) mit sed, perl, etc
- bind-Mount
- Alternatives System
- boot Prozess
    - System bootet nicht mehr - LILO
    - System bootet nicht mehr - hängt Anzeige "GRUB" links oben

- Nervige Piepstöne der Konsole abstellen
- Windows-Freigabe in Linux mounten
- letztes Logfile durch logrotate via Mail senden lassen
- Uhrzeit ist falsch und der ntpd setzt auch nicht die richtige Zeit
- GroupID (GID) eines Prozesses anzeigen
- ps aux zeigt nur die uid (userid) statt dem usernamen an
- dmesg Timestamp in lesbare Form umwandeln
- Mini-Screen-Howoto
- Verzeichnis zu ISO-Image machen
- Userverwaltung anlegen-löschen-ändern
- Langlaufende (hängende) Prozesse beenden
- etckeeper mit gitlab verwenden
- Fork-Bomb und andere feine Sachen
- Laufzeit eines Cronjobs begrenzen (Timeout)
- Mehr Infos aus htop herauskitzeln
- Swapfile anlegen und einrichten
- Cronjob jeweils am letzten Tag des Monats ausführen
- Zombie-Prozesse
- Blocks berechnen
- Seit wann läuft ein Prozess?

Anzahl der Cores/Prozessoren herausfinden

Mit den folgenden Befehlen lässt sich einfach die Anzahl „echter“ Prozessoren und deren Cores herausfinden.

Anzahl der physikalischen CPUs:

```
grep "^physical id" /proc/cpuinfo | sort -u | wc -l
```

Anzahl der Cores pro CPU:

```
grep "^core id" /proc/cpuinfo | sort -u | wc -l
```

Dieses Kommando zeigt die Gesamtanzahl der Prozessoren im System, auch der virtualisierten.

```
grep "^processor" /proc/cpuinfo | sort -u | wc -l
```

`lscpu` aus dem util-linux Paket zeigt die Prozessorinfos übersichtlich an:

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
Address sizes:       46 bits physical, 48 bits virtual
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  1
Core(s) per socket:  8
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               63
Model name:          Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
Stepping:            2
CPU MHz:             2600.073
BogoMIPS:            5200.04
Hypervisor vendor:   Xen
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
```

```
L2 cache:            256K
L3 cache:            35840K
NUMA node0 CPU(s):   0-7
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush acpi mmx fxsr sse sse2 ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl
cpuid pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer
aes xsave avx f16c rdrand hypervisor lahf_lm abm cpuid_fault invpcid_single pti ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt flush_l1d
```

Lustige Sachen mit Linux

Linux Systemverwaltung

- doppelt vergebene UIDs finden

```
awk -F: '{print $3}' /etc/passwd | sort -n |uniq -d
```

- normale Dateien älter als 2 Tage anzeigen

```
find . -type f -mtime +2 -printf "%f %Tc\\n"
```

- normale Dateien älter als 2 Tage finden und löschen

```
find . -type f -mtime +2 -exec rm -f {} \;
```

- normale Dateien älter als 2 Tage, deren Dateiname nicht mit .gz endet, mit GZIP packen

```
find . -type f -mtime +2 ! -name "*.gz" -exec gzip {} \;
```

- normale Dateien in Unterverzeichnis g*/* mit *2005* im Namen und nicht mit der Endung *.gz„ (also nicht nicht gezippt) mit gzip packen

```
find g*/* -type f -name "*2005*" -a ! -name "*.gz" -exec gzip {} \;
```

- normale Dateien finden größer als 1MB

```
find . -type f –size +1024k
```

- alte (noch ungezippte) Logs, älter als 2 Tage finden und zippen

```
find /opt/fast/datasearch/var/log/querylogs.backup -type f -mtime +2 ! -name "*.gz" -exec gzip {
```

- Inhalt inkl. Unterverzeichnisse eines Ordners löschen, der zu groß für ein rm -rf ist

```
find /opt/vollesVerzeichnis/ -mindepth 1 -maxdepth 1 -exec rm -rf {} \;
```

- Dateien durchsuchen, auch wenn man den genauen Pfad oder den Dateinamen nicht kennt

```
find /verzeichnis -exec grep -s -l "sucheintrag" {} \;
```

- Files nach Inhalt durchsuchen und z.B. löschen wenn Übereinstimmung (nützlich um mailqueues zu leeren)

```
find . -exec grep -q "spamdomain.net" '{}' \; -exec rm {} \;
```

- Verzeichnisse rekursiv durchsuchen und jeweils die ältesten 5 Dateien ausgeben

```
find . -type d -exec bash -c 'echo "next dir: ${1}" ; ls -lt "$1" | grep ^- | head -n 5' bash {} \;
```

- alle Sonderzeichen in Dateien anzeigen

```
cat -A datei | less
```

- alle Dateinamen im aktuellen Verzeichnis in Kleinbuchstaben umwandeln

```
for name in * ; do mv -i $name `echo $name | tr [A-ZAeOeUe] [a-zAeOeUe]` ; done
```

- Erweiterung einer Datei ändern (das Beispiel benennt .TXT in .txt um)

```
for a in *.TXT ; do echo mv -i $a `basename $a .TXT`.txt ; done
```

- Kommandozeile verschleiern - verhindert die Anzeige des ausgeführten Programms in der Prozessliste

```
doexec EigentlichesProgramm Schleier Parameter
```

- SSH Tunneln

```
ssh -g -L3389:192.168.110.17:80 myserver.de
```

- Bootsektor einer Festplatte überschreiben (/dev/cciss/c0d0 durch das richtige Laufwerk ersetzen, grub.conf)

```
cat /etc/grub.conf
dd if=/dev/zero of=/dev/cciss/c1d0 bs=512 count=1
```

- Datei mit einer bestimmten Größe erstellen (hier z.B. 20MB groß)

```
dd if=/dev/zero of=/tmp/blafasel bs=1024k count=20
```

- Eine Zeile mit awk parsen/ausgeben - Leerzeichen als Feldseparator, Feld 1 ausgeben

```
cat <datei> | awk -F" " '{ print $1 }'
```

- access.log analysieren, nur Requests ausgeben, die Dateien größer als 150 MB runterluden (1024*1024*150)

```
tail -f access.log | awk '$6 > 157286400'
```

- ein .patch-File erstellen

```
diff -aur file-alt.c file-neu.c
```

- DNS-Server reverse-zonefile neu laden

```
rndc reload 180.34.212.in-addr.arpa.
```

- alle mit a beginnenden Dateien und Verzeichnisse inklusive Unterverzeichnisse auflisten ? unübersichtlich

```
ls -a*
```

- nur noch Dateien und Verzeichnisse zeigen OHNE die Unterverzeichnisse

```
ls -d a*
```

- Scriptautomatisierung - Texte ausgeben mit yes

Manche Programme verlangen eine Bestätigung der Eingabe mit z.B. y oder no. Dies kann automatisch passieren durch das Programm yes

```
yes | unzip bla.zip
```

- den Kommandostack / die History der Bash löschen

```
export HISTSIZE=0
```

- eigene IP-Adresse rausfinden (nützlich bei Einwahlverbindung) für ipv4 und ipv6

```
env LC_ALL=C /sbin/ifconfig eth0 | sed -n '/inet addr:/s/ [^r]*..//gp'
env LC_ALL=C /sbin/ifconfig eth0 | sed -n '/inet6 addr:/s/ [^r]*..//gp'
```

- SWAP-Speicher erweitern (z.B. nach Arbeitsspeichererweiterung)

```
dd if=/dev/zero of=/var/tmp/swapfile bs=2GB count=1  # erzeugt ein 2GB Swapfile
mkswap /var/swap/swapfile  #schreibt die Swapsignatur
```

in /etc/fstab folgende Zeile ergänzen

```
/var/swap/swapfile  swap swap  pri=1000 0 0
```

und nun mit swapon -a den neuen Speicher einschalten

- VLC-Videostream mit Logitech Quickcam (HTTP mit Authentifizierung) Server-Modus

```
vlc v4l:/dev/video0:norm=0:frecuency=0:channel=0 --no-audio --no-sout-audio --sout-http-user use
```

- schnell ganz viele User anlegen

```
for i in `seq -w 1 65`; do echo "useradd -c \"Test-Account $i\" test$i; echo `pwgen -1 -c -n -s
```

Backup bezogen

- Backup eines Servers erstellen

exclude.txt (hier kommen die zu übergehenden Verzeichnisse rein)

```
/proc
/dev
/lost+found
/tmp
/sys
```

```
tar czf /tmp/backup.tgz --exclude-from=excludes.txt /
```

- tar-Standardausgabe via SSH auf Zielsystem schicken

```
tar cf - `cat /verzeichnisliste.txt` | ssh -c blowfish remote.meinserver.de "cd /zielverzeichnis
```

- Verzeichnis via rsync auf anderes System spiegeln

```
rsync -rlvpogDt --size-only --partial --stats --progress -e "ssh -c blowfish" root@<quellserver>
```

HAHA

- Star Wars Beep auf der Console

```
beep -l 350 -f 392 -D 100 --new -l 350 -f 392 -D 100 --new -l 350 -f 392 -D 100 --new -l 250 -f
```

- lecker Bier:

```
[magenbrot@brot ~]$ cd /pub
[magenbrot@brot pub]$ more beer
                  .OOOOOOOOO.
              .OOOOOOOOOOOOOOOOO
            OOOOOOOOOOOOOOOOOOOOOOO
           OOOOOOOOOOOOOOOOOOOOOOOOO
           oOOOOOOOOOOOOOOOOOOOOOOOoOO
          |OoooooOOOOOOOOOOOOOOooooO|O
   _______|OOOOOoooooOOOOOoooooOOOOO|O
  /        OOOOOOOOOoOOOOOoOOOOOOOOO|O
 /  .------+OOOOOOOOOoOOOOOoOOOOOOOOO|
/  /       |OOOOOOOOOoOOOoOOOOOOOOOO|
|  |       |.OOOOOOOOOoOOOoOOOOOOOOO.|
|  |       |.....OOOOOOoOOOoOOOOO.....|
|  |       |...........OOOo..........|
 \  \       |...........OOOo..........|
  \  '------+............Oo...........|
   \_______ ..........................|
           |..........................|
          /............................
         ooo........................ooo
             oooo..............oooo
                 OOOOOOOOOOOOOO
```

- schwanzersatz.sh:

```
echo `uptime|grep days|sed 's/.*up \([0-9]*\) day.*/\1\/10+/'; \
cat /proc/cpuinfo|grep "cpu MHz"|awk '{print $4"/30 +";}'; free|grep '^Mem' \
|awk '{print $3"/1024/3+"}'; df -P -k -x nfs | grep -v 1k \
| awk '{if ($1 ~ "/dev/(scsi|sd)"){ s+= $2} s+= $2;} END \
{print s/1024/50"/15+70";}'`|bc|sed 's/\(.$\)/.\1cm/'
```

Zeilen in Felder zerlegen

Gelegentlich benötigt man lediglich ein bestimmtes Feld einer Zeile. Mit dem 'cut' Kommando läßt sich ein einzelnes Feld leicht extrahieren.

Das Kommando

```
uptime | cut -d, -f3
```

Zerlegt die Ausgabe von uptime anhand des Seperators „,“ und liefert das dritte Feld - also die Anzahl momentan angemeldeter Anwender.

Versucht man allerding mit dieser Methode die aktuelle Last auf dem System zu ermitteln, so schlägt die Ausgabe gelegentlich fehl. Das Uptime Kommando liefert Text, der für einen menschlichen Leser und nicht zur Weiterverarbeitung gedacht ist. Die Anzahl der Felder ändert sich durch zusammenfassen von Ausgaben zur besseren Lesbarkeit.

Die Lastangabe ist jedoch immer das letzte Feld. Leider kann das 'cut' Kommando nur von Links zählen, das 'letzte' Feld ist aber identisch mit 'das erste - von rechts gezählt'.

Mit dem 'awk' Kommando kommt man zum Ziel.

```
uptime | awk -F, '{print $NF}'
```

Liefert das letzte Feld einer Zeile die durch ',' getrennt ist.

```
uptime | awk -F, '{print $(NF-1)}'
```

entsprechend das vorletzte.

Zeichenfolgen von Sondertasten ermitteln

um die Zeichenfolge (Tastencode) einer Sondertaste herauszufinden, wird cat folgendermaßen aufgerufen:

```
cat -vu
```

Jede gedrückte Taste gibt nun den entsprechenden Code aus. Beenden kann man diesen Modus mit STRG+D.

Alternativ kann man STRG+V voransetzen, z.B. STRG+V F1.

Harten Reset durchführen

Hängt der Rechner klappt der Reboot im allgemeinen aber nicht mehr – er kann keinen weiteren Prozess starten, aber dieser würde für den Reboot benötigt.. Über das Interface „/proc/sys/kernel/ctrl-alt-del“ kann man dieses Verhalten allerdings umkonfigurieren. Ein

```
echo "1" > /proc/sys/kernel/ctrl-alt-del
```

führt dazu, daß der Rechner sofort neu startet wenn die Tastenkombination gedrückt wird- und zwar immer.

Arbeiten mit sed

Mit einem 'normalen' Editor ist das Bearbeiten von Dateien etwas aufwändig. Datei laden, Position suchen, Text aendern, Datei speichern. Wie aufwändig dies ist, sieht man erst im Vergleich zum Streameditor sed.

Wenn wir z.B. die ersten drei Zeilen einer Datei loeschen wollen, gehen wir so vor:

```
sed -e '1,3D' eingabe.txt
```

Wenn du dieses Beispiel mit einer echten Datei testest, wird dir vermutlich nicht gefallen, dass die Ausgabe nur auf dem Bildschirm erscheint. Also leiten wir die Standardausgabe auf eine Datei um:

```
sed -e '1,3D' eingabe.txt > ausgabe.txt
```

Die Datei eingabe.txt bleibt unverändert. In ausgabe.txt wird der Inhalt von eingabe.txt ohne die ersten drei Zeilen geschrieben. Wenn Sie die letzte Zeile löschen wollen:

```
sed -e '$D' eingabe.txt > ausgabe.txt
```

Das Zeichen $ steht als für die letzte Zeile. Dieser Befehl würde also alles ab der 2. Zeile löschen:

```
sed -e '2,$D' eingabe.txt > ausgabe.txt
```

Statt der Zeilennummern können wir aber die Grenzen des Arbeitsbereichs auch mit regulären Ausdrücken festlegen. Der folgende Befehl löscht alles ab dem Auftreten von 'BEGIN' bis zum Auftreten von 'END':

```
sed -e '/BEGIN/END/D' eingabe.txt > ausgabe.txt
```

Würde 'END' nicht gefunden, so würde sed bis zum Dateiende weiter löschen.

Zu guter Letzt wollen wir die Auswahl umkehren - es sollen z.B. alle Zeilen gelöscht werden, nur die letzte nicht:

```
sed -e '$ ! D' eingabe.txt > ausgabe.txt
```

Das geht natürlich auch mit regulären Ausdrücken:

```
sed -e '/BEGIN/END/D' eingabe.txt > ausgabe.txt
```

Statt mit der Umleitung (>) kann sed auch inline (direkt in der Datei) editieren. Dazu einfach den Schalter -i verwenden:

```
sed -i -e '1,3D' eingabe.txt
```

Einer Zeile einen String hinzufügen, die mit einem bestimmten String anfängt:

```
sed '/^User_Alias ADMINS=/ s/$/,newadmin/' /etc/sudoers
```

Zeile mit pattern +X nachfolgende Zeilen löschen:

```
sed -e '/pattern/,+5d' datei.txt
```

Rekursive Verarbeitung verschiedenster Dateien:

```
find -type f -name "*.conf" -print0 | xargs -0 sed -i 's#http_certificate = "30,14"#http_certifi
```

Einen String mit vorgestelltem Prefix „verdoppeln“:

```
# Beispielliste mit Domainnamen. domainliste.txt enthält:
1lik28pvitfl8q9o0jh.de
li816ifohpxhwfuj0d5.de
v6rhv3hwx3w9cgd4syi.de
gxxkz832p71j6hujron.de
aen9nldoy5wds640yg5.de
okzi8954w6ihsbcvai8.de
```

Mit diesem Kommando wird jede Zeile verdoppelt, mit Leerzeichen getrennt und dem ersten Eintrag ein www. vorangestellt:

```
cat domainliste.txt | sed 's/\<.*\>/www.& &/'

www.1lik28pvitfl8q9o0jh.de 1lik28pvitfl8q9o0jh.de
www.li816ifohpxhwfuj0d5.de li816ifohpxhwfuj0d5.de
www.v6rhv3hwx3w9cgd4syi.de v6rhv3hwx3w9cgd4syi.de
www.gxxkz832p71j6hujron.de gxxkz832p71j6hujron.de
www.aen9nldoy5wds640yg5.de aen9nldoy5wds640yg5.de
www.okzi8954w6ihsbcvai8.de okzi8954w6ihsbcvai8.de
```

Mit tr läßt sich noch der Zeilenumbruch in ein Leerzeichen umwandeln. Nützlich um z.B. aus einer Domainliste einen Eintrag für die /etc/hosts zu generieren:

```
cat domainliste.txt | sed 's/\<.*\>/www.& &/' | tr '\n' ' '

www.1lik28pvitfl8q9o0jh.de 1lik28pvitfl8q9o0jh.de www.li816ifohpxhwfuj0d5.de li816ifohpxhwfuj0d5
```

Reguläre Ausdrücke (regex) mit sed, perl, etc

Hier ein paar Beispiele für nützliche reguläre Ausdrücke:

Logfiles anonymisieren

- Dieses Beispiel ersetzt die IP-Adressen aus einem Logfile durch die IP 123.123.123.123:

```
sed 's/\([0-9]\{1,3\}\)\.\([0-9]\{1,3\}\)\.\([0-9]\{1,3\}\)\.\([0-9]\{1,3\}\)/123\.123\.123\.123
```

bind-Mount

Ein sogenannter bind-mount bindet ein Unterverzeichnis an einer anderen Stelle im Verzeichnisbaum ein. Dies ist z.B. nützlich wenn man etwa in seiner ursprünglichen /var/log-Partition nicht genügend Platz hat, aber an anderer Stelle noch viel frei ist:

```
mount -o bind /mnt/httpd /var/log/httpd
```

mountet das Verzeichnis /mnt/httpd nach /var/log/httpd. Man sollte VORHER den Inhalt aus dem Zielverzeichnis nach /mnt/httpd verschoben haben.

Ein Eintrag in der fstab sieht dann so aus:

```
/mnt/httpd              /var/log/httpd          none    rw,bind         0 0
```

Alternativ wäre ein symbolischer Link (ln -s) wahrscheinlich ressourcenschonender (hab ich noch nicht gemessen).

Alternatives System

Beispiel java runtime environment:

eigenes java-binary bei alternatives registrieren:

```
alternatives --install /usr/bin/java java /usr/java/jre1.6.0_01/bin/java 2
```

Voreinstellung ändern

```
alternatives --config java
```

boot Prozess

System bootet nicht mehr - LILO

- L [error code]

siehe LILO-Error-Codes möglicherweise ist der Datenträger defekt oder weist eine fehlerhafte Geometrie auf.

- LI

möglicherweise befinden sich die LILO Dateien nicht dort, wo sie hingehören Dies kann verursacht werden durch eine fehlerhafte Platten-Geometrie oder durch Verschieben von /boot/boot.b ohne Neuinstallation von LILO.

- LIL

Bootsector ist möglicherweise korrupt oder der Bootsector entspricht nicht der physikalischen Eigenschaft der Festplatte (Bad hard disk/floppy disk)

- LIL?

Der zweite Level von LILO hat eine falsche Adresse geladen. Möglicherweise kann dies durch erneutes Ausführen von lilo oder durch Anpassen der Festplattenparameter behoben werden

- LIL-

Beschädigte discriptor table. Meistens durch erneutes Ausführen von lilo zu beheben.

- LILO

Keine Fehler

- LILO Error Codes

0x00 - Internal Error
0x01 - Illegal command
0x02 - Address mark not found
0x04 - Sector not found
0x06 - Change line active
0x08 - DMA Overrun
0x09 - DMA Attempt accross 64k Boundary
0x0c - Invalid Media
0x10 - CRC error
0x20 - Controller error
0x40 - Seek failure
0x80 - Disk timeout

boot Prozess

System bootet nicht mehr - hängt Anzeige "GRUB" links oben

System hängt beim Booten. BIOS wird noch durchlaufen, danach hängt das System mit der Anzeige „GRUB“ im linken oberen Teil des Bildschirms.
? den GRUB-Bootloader hats zerschossen. Das kann passieren durch Kernel-Updates / Plattenfehler / Anwenderfehler.

1. Runterladen einer aktuellen Version von Linux (Ich habe Fedora Core 6 verwendet, das System muss nur fähig sein in einen Prompt zu starten und die vorhandenen Partitionen zu mounten).
2. Von der CD booten
3. Am Begrüßungsbildschirm „linux rescue“ eingeben und mit Enter bestätigen
4. Sprache/Tastaturbelegung entsprechend auswählen
5. Netzwerkdevices brauchen nicht gestartet zu werden
6. Bei der folgenden Auswahl „Continue“/„Weiter“ auswählen, die gefundenen Partitionen werden nun nach /mnt/sysimage gemountet
7. nochmal mit Enter bestätigen
8. Am Prompt jetzt mit „chroot /mnt/sysimage“ auf das installierte System wechseln
9. Mit „fdisk -l“ den Namen der Bootplatte rausfinden, sofern er nicht bekannt ist (/dev/sd* oder /dev/hd*)
10. Mit „grub-install /dev/<bootplatte>“ den GRUB in den Bootsektor installieren
11. nach zweimaliger Eingabe von „exit“ sollte das System rebooten und wieder normal starten (CD aus dem Laufwerk nehmen)

Nervige Piepstöne der Konsole abstellen

An diversen Stellen sind beim Drücken einer „falschen“ Taste oder bei gewissen Events Piep-Töne zu hören. Im Büro oder an der Uni sind diese Klänge eher lästig und nerven. Es piepst, wenn der Gnome-Anmeldebildschirm erscheint, wenn man mit der „Backspace“-Taste in der Kommando-Zeile zu viele Zeichen löscht oder versucht mit „Tab“ etwas zu ergänzen.

Konsole

Die Datei /etc/inputrc muss wie folgt angepasst werden. Bei folgende Zeile muss die Auskommentierung (#) gelöscht werden.

```
set bell-style none
```

Gnome

Unter System ? Administration ? Anmeldebildschirm auf den Reiter „Barrierefreiheit“ wechseln und die Option „Klänge“ / „Anmeldebildschirm bereit“ deaktivieren.

Unter Umständen kann dieses Verhalten auch gewünscht sein, wenn akustisch signalisiert werden soll, wenn das System bereit.

Unter System ? Einstellungen ? Audio dann den Reiter 'Systemglocke' wählen und 'Systemglocke aktivieren' deaktivieren.

Alternative Möglichkeiten

- Neukomplieren des Kernels ohne „PC speaker support“
- Das pcspkr-Kernel-Module auf die Schwarze Liste setzen.

Die Datei /etc/modprobe.d/blacklist um folgenden Eintrag ergänzen

```
#Nerviger Speaker-Beep deaktivieren
blacklist pcspkr
```

Windows-Freigabe in Linux mounten

Um eine Windows- oder Samba-Freigabe schon beim Booten in Linux zu mounten ist folgendermaßen vorzugehen:

Installation der cifs-utils:

```
apt-get update

apt-get install cifs-utils
```

Eintrag in /etc/fstab (eigener User hat uid und gid 500):

```
//winserver/freigabe /home/user/windows/freigabe cifs user,credentials=/etc/fstab.credentials,ui
```

nach /etc/fstab.credentials kommt ein Eintrag in der Form:

```
username=windowsuser
password=windowspasswort
```

letztes Logfile durch logrotate via Mail senden lassen

Logrotate bietet zwar auch so eine Funktion an, jedoch wird dabei das Logfile nicht auf dem Server archiviert sondern nur via Mail verschickt und danach gelöscht.

Mit folgendem kleinen Script kann man das Logfile versenden und es dennoch auf dem Server belassen. Dieses Beispiel versendet das maillog:

```
#!/bin/sh

REC="empfaenger@domain.com"

/bin/gzip -c /var/log/maillog > /tmp/maillog.gz
/bin/echo -e "Hallo,\nAnbei das maillog von `date`.\nViele Grüße, Ihr Server" | mutt -s
"Logfile von `date`" -a /tmp/maillog.gz $REC
/bin/rm -f /tmp/maillog.gz
```

logrotate muss dann nur noch so konfiguriert werden, dass es das gewünschte Logfile gesondert behandelt. Für das maillog würde das dann bedeuten, dass der Eintrag aus /etc/logrotate.d/syslog entfernt und eine neue Datei /etc/logrotate.d/maillog mit folgendem Inhalt angelegt wird:

```
/var/log/maillog {
    rotate 365
    olddir /var/log/maillog.old
    daily
    sharedscripts
    prerotate
        /bin/gzip -c /var/log/maillog > /tmp/maillog.gz
        /bin/echo -e "Hallo,\nAnbei das maillog von `date`.\nViele Grüße, ihr netter Server" | m
        /bin/rm -f /tmp/maillog.gz
    endscript
    postrotate
 /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
 /bin/kill -HUP `cat /var/run/rsyslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

Achtung: das Logfile auf dem Server wird so logischerweise täglich rotiert, via Mail versendet, in das (vorher anzulegende) Unterverzeichnis /var/log/maillog.old verschoben und 365 Tage aufgehoben.

Uhrzeit ist falsch und der ntpd setzt auch nicht die richtige Zeit

Hier sind wahrscheinlich falsche Informationen zur Zeitzone gesetzt. Folgende Maßnahme schafft Abhilfe:

```
service ntpd stop
rm /etc/localtime
ln -s /usr/share/zoneinfo/Europe/Berlin /etc/localtime
ntpdate 0.ubuntu.pool.ntp.org
hwclock --systohc
service ntpd start
```

GroupID (GID) eines Prozesses anzeigen

Dass sich mit „ps aux“ sämtliche Prozesse des Systems mit dem Usernamen/UID in der ersten Spalte anzeigen lassen, ist bekannt.

Um auch die Gruppe, unter der ein Prozess läuft, zu sehen, hilft dieses Kommando:

```
ps -eo "%U %G %p %a"

USER     GROUP      PID COMMAND

root     root      2063 /sbin/getty -8 38400 tty1
root     root      2671 /usr/lib/upower/upowerd
root     root      3176 lightdm --session-child 12 19
userx    gruppe    6331 xscreensaver -no-splash
userx    gruppe    6333 xfce4-session
```

Man kann sich noch viel viel weitere Felder ausgeben lassen. Welche das sind findet man in der ps-manpage unter „AIX FORMAT DESCRIPTORS“ und „STANDARD FORMAT SPECIFIERS“

ps aux zeigt nur die uid (userid) statt dem usernamen an

„ps aux“ zeigt in der CentOS 5.3 Version (procps-3.2.7-9.el5.rpm) bei manchen Usern statt des Usernamens nur die UID an. Ich habe nun rausgefunden, dass ps bei Usernamen länger als 8 Zeichen nur die UID anzeigt:

```
[root@server etc]# ps axu |grep hal
root     20188  0.0  0.0  61124   716 pts/2    S+   15:13   0:00 grep hal
68       22664  0.0  0.0  28980  3448 ?        Ss   14:51   0:00 hald
root     22665  0.0  0.0  21648  1020 ?        S    14:51   0:00 hald-runner
68       22693  0.0  0.0  10624   748 ?        S    14:51   0:00 hald-addon-acpi: listening on a
68       22835  0.0  0.0  10624   724 ?        S    14:51   0:00 hald-addon-keyboard: listening
root     22979  0.0  0.0  10196   660 ?        S    14:51   0:00 hald-addon-storage: polling /de
```

Beim Haldaemon ist das aufgefallen, der wurde in früheren CentOS-Versionen noch unter dem User root gestartet.

dmesg Timestamp in lesbare Form umwandeln

Mit dem Kommando dmesg kann man sich den Kernel Ringbuffer anzeigen lassen, also alle Kernel-Meldungen seit Systemstart. Die Zeit wird dabei als Sekunden seit dem Systemstart ausgegeben.

Ab Debian Wheezy kann man einfach über 'dmesg -T' eine lesbare Zeit ausgeben lassen.

Braucht man die genaue Zeit, kann man die Meldungen mit lesbarer Zeit entweder in /var/log/kern.log finden oder sich die Zeit über einen kleinen Umweg konvertieren lassen (in diesem Fall einfach über einen alias Befehl):

```
alias kmsg='dmesg | perl -ne "BEGIN{\$a= time()- qx!cat /proc/uptime!}; s/\[\s*(\d+)\.\d+\]/loca
```

Der Output sieht dann so aus:

```
Fri Apr  5 11:06:50 2013 ip_tables: (C) 2000-2006 Netfilter Core Team
Fri Apr  5 11:06:50 2013 init: plymouth-stop pre-start process (2024) terminated with status 1
Fri Apr  5 11:06:51 2013 eth0: no IPv6 routers present
```

Mini-Screen-Howoto

Screen ist ein Fenstermanager zur Verwendung mit textbasierten Eingabefenstern (Textkonsole) und kommt typischerweise auf UNIX-ähnlichen Betriebssystemen zum Einsatz. Hierbei ist es möglich, innerhalb eines einzigen Zugangs (zum Beispiel über ein Terminal oder eine Terminalemulation) verschiedene virtuelle Konsolensitzungen zu erzeugen und zu verwalten. Darüber hinaus können Sitzungen in den Hintergrund geschoben und später fortgeführt werden.

Steuerung von Screen

screen starten

```
$ screen <optional zu startendes Programm>
```

Die verschiedenen Funktionen in screen werden über ein Tastenkürzel „STRG-a + <x>“ (STRG und a werden zusammen gedrückt und losgelassen, danach folgt der Befehl) aufgerufen.

Hier die wichtigsten Kürzel:

```
STRG-a + c    # neues Window innerhalb der Session erstellen
STRG-a + leer # zum nächsten Window wechseln (oder STRG-a + n)
STRG-a + p    # zum vorherigen Fenster wechseln
STRG+a 0-9    # zum Fenster mit der entsprechenden Nummer wechseln
```

Beendet wird eine Screen-Session wenn das letzte Fenster geschlossen wird (Ausloggen mit STRG+d).

Soll die Session im Hintergrund weiterlaufen, kann man sie mit diesem Kürzel einfach abhängen (detach):

```
STRG-a + d
```

mit diesem Befehl kann eine Session aus dem Hintergrund hervorgeholt werden:

```
screen -list # laufende Screen-Sessions anzeigen

screen -r <pid> # auf eine abgehängte Session verbinden
oder
screen -x  <pid> # auf eine NICHT abgehängte Session verbinden. Damit kann man z.B. eine Shell m
```

Es reichen übrigens die Anfangszahlen einer PID (bis sie nicht mehr mit anderen Sessions verwechselt werden kann:

```
There are screens on:
 18209.pts-18.server (01/11/2013 11:11:23 AM) (Detached)
 19097.pts-24.server (01/11/2013 11:11:07 AM) (Detached)
2 Sockets in /var/run/screen/S-user.

$ screen -r 19 ## holt den Screen mit der PID 19097 zurück
```

Scrollen in Screen

Shift+Bild-hoch/runter funktioniert im Screen leider nicht. Dafür gibts den „Copy Mode“. Dieser wird über „STRG+a ESC“ aktiviert. Jetzt kann man mit Bild-hoch/runter arbeiten. Zum Verlassen des Copy Mode „ESC“ drücken.

In meiner Screenrc ist eine Option, die das Scrollen mit BILD HOCH/RUNTER wieder aktiviert. Allerdings wird dieser Terminal-Buffer zw. allen offenen Fenstern geteilt (Im Gegensatz zum Screen-internen Buffer).

gemeinsame Screensession / Shared Screen

Es ist möglich einen Screen gemeinsam zu bedienen. Das ist z.B. für Fernsupport nützlich oder um gemeinsam an einem Problem zu arbeiten.

Dazu startet user1 einen Screen und aktiviert den Multiuser-Mode:

```
STRG-a :multiuser on
STRG-a :acladd user2
```

Der user2 kann sich nun mit dem Screen verbinden (er muss screen nicht nochmals starten):

```
# screen -x [screen-id]
```

Sind die User die gleichen (also z.B. root auf einem Server) kann man sich das acladd sparen.

screenrc

Mit einer screenrc lassen sich einige Optionen setzen, die das ganze noch etwas komfortabler machen. Die Screenrc kann entweder als .screenrc im Homeverzeichnis oder als /etc/screenrc global abgelegt werden. Hier ist meine aktuelle .screenrc:

```
#
# magenbrot screenrc
#
# $Id: screenrc 125 2013-01-10 09:59:41Z magenbrot $
#

# disable startup message
startup_message off

# the status line and window caption
hardstatus alwayslastline "%{yb}  [ $LOGNAME@%H ]  %{gb} %0c:%s | %d.%m.%Y %{rb} %l %{wb} %w "

#defbce on
deflogin off

# scrollback history
defscrollback 10000

# visual bell off (screen flicker)
vbell off
bell_msg ""

# detach on disconnect
autodetach on

# make search case-insensitive
ignorecase on

# display time for warnings and messages
#msgminwait 10
msgminwait 0

# buffer emulator enables scrollback with PAGE-UP/DOWN (with some drawbacks, all windows share t
termcapinfo xterm|xterms|xs|rxvt ti@:te@
```

Verzeichnis zu ISO-Image machen

Manchmal ist es nützlich aus einem Verzeichnis und seinem Inhalt ein ISO-Image zu machen. Ich nutze das z.B. für Debian-Installation über Dell-Drac bei Servern, die keinen Zugang zum Internet haben. Das Image läßt sich natürlich auch sehr einfach auf CD brennen.

Der Befehl ist ganz einfach:

```
mkisofs -V <LABEL> -r <DIRECTORY> > cdrom.iso
```

LABEL ist dabei ein Bezeichner für das ISO und DIRECTORY das gewünschte Verzeichnis.

Userverwaltung anlegen-löschen-ändern

Ich zeige hier die Userverwaltung anhand der Tools adduser, deluser, addgroup und delgroup.

User anlegen/löschen:

```
adduser [--uid XXXX] myname groupname
adduser --system myname # einen Systemuser anlegen (uid < 1000)
deluser [--remove-home] [--remove-all-files] [--backup] [--backup-to DIR] myname
```

Bei adduser werden dabei auch zusätzlich noch die Finger-Infos abgefragt (GECOS).

User einer Gruppe hinzufügen/aus einer Gruppe entfernen:

```
deluser myname groupname
```

Eine Gruppe anlegen/löschen:

```
addgroup [--gid XXXX] groupname
addgroup --system groupname # eine Systemgruppe anlegen (gid < 1000)
delgroup groupname
```

Noch ein paar nützliche Tipps:

adduser - Parameter

- –home verzeichnis # Home-Verzeichnis festlegen, wenn anders als /home/$user
- –no-create-home # kein Home-Verzeichnis anlegen
- –shell # Shell festlegen, default /bin/bash
- –ingroup / –gid # Hauptgruppe festlegen
- –disabled-password # User ohne Passwort anlegen, ein Login per SSH-Key ist damit z.B. möglich
- –disabled-login # User deaktivieren, kein Login möglich

deluser - Parameter

- –remove-home # Home-Verzeichnis löschen
- –remove-all-files # alle Dateien, die diesem User gehören, vom System löschen (inkl. Home)
- –backup] –backup-to verzeichnis # erstellt ein .tar.gz der Dateien des Benutzers und legt diese im angegebenen Verzeichnis ab

Für adduser und deluser lassen sich viele Parameter über die Konfigurationsdateien /etc/adduser.conf und /etc/deluser.conf vordefinieren.

Langlaufende (hängende) Prozesse beenden

Ich hatte schon öfter das Problem, dass sich manche Prozesse aufhängen (kein Wait, kein Timeout, aber auch keinerlei Ausgabe im strace) und sich nicht mehr regulär beenden. Das kam z.B. bei PHP-FPM-Children vor oder bei der Konvertierung von HTML-Dokumenten in PDF mittels pdflib. Eine genaue Ursache konnte ich leider nicht finden, daher habe ich einen dieser Cronjobs eingerichtet, der „alte“ Prozesse einfach killt.

Möglichkeit 1: Killen über „killall –older-than“ (Nachteil, funktioniert nicht bei Prozessen mit Leerzeichen oder als Match auf Programmparameter)

```
# pdflatex haengt gerne mal (Kille stündlich alle Prozesse älter als 30 Minuten)
0 * * * * killall --older-than 30m pdflatex
```

Möglichkeit 2: Prozess über find suchen, genau filtern und per xargs kill beenden

```
# das hier killt "www pool" Prozesse (z.B. bei PHP-FPM) die älter als eine Stunde sind.
# Bei PHP-FPM sollte pm.max_requests ungleich 0 sein, z.B. 50)
*/5 * * * * find /proc -maxdepth 1 -user deploy -type d -mmin +60 -exec basename {} \; | xargs p
```

etckeeper mit gitlab verwenden

gitlab vorbereiten

- Installation: https://about.gitlab.com/downloads/#debian8
- Gruppe „etckeeper“ in gitlab angelegt
- 1 neues Projekt pro Server in dieser Gruppe angelegt
- 1 dedizierten User für Serverdinge angelegt
- alle SSH-Pubkey der Server in diesem User hinterlegt.

Installation

```
apt update && apt install -y etckeeper git
```

globale git Konfiguration

```
git config --global user.name "Oliver Völker"
git config --global user.email "info@ovtec.it"
git config --global core.editor "vim"
git config --global push.default simple
```

etckeeper Konfiguration

/etc/etckeeper/etckeeper.conf

```
VCS="git"
AVOID_SPECIAL_FILE_WARNING=1
PUSH_REMOTE="origin"
```

ersten Commit einchecken

```
cd /etc

# start clean?

# rm -rf .git

git init --initial-branch=main --object-format=sha1

# oder falls das schon die Branch 'master' existiert: git switch --create main

git remote add origin git@gitlab.ovtec.it:etckeeper/`hostname -f`.git

# oder fish: git remote add origin git@gitlab.ovtec.it:etckeeper/$(hostname -f).git

git push --set-upstream origin main

etckeeper commit "initial commit"
```

Dateien ignorieren (Beispiel)

/etc/.gitignore

```
git rm --cached openvpn/OVTEC/status.txt
git add .gitignore
```

automatische Commits bei Logout

Diese Datei z.B. in /usr/local/share/logout_trap.sh ablegen und beim Login jedes Users z.B. durch einen Eintrag in ~/.profile sourcen lassen.

```
[ `id -u` -eq 0 ] || return

_USER=""
_HOSTNAME=`hostname`
if [ -n "$SUDO_USER" ]; then
    _USER="$SUDO_USER"
else
    # try to check tty ownership, in case user su'd to root
    _TTY="$(tty 2>/dev/null || true)"
    if [ -n "$_TTY" ] && [ -c "$_TTY" ]; then
        _USER="$(find "$_TTY" -printf "%u")"
    fi
fi

_etckeeper()
{
    if [ -x /usr/bin/git ] && [ -d /etc/.git ]; then
        (
        cd /etc

        if [ $(git status --porcelain | wc -l) -eq 0 ] ; then
            return 0
        fi

        if ! git add --all; then
            echo "warning: git add --all" >&2
        fi

        etckeeper commit "Automated commit at exit"
        )
    fi
}

if [ -n "$_USER" ]; then
    if [ -z "$GIT_AUTHOR_NAME" ]; then
        [ -f /home/$_USER/.gitconfig ] && GIT_AUTHOR_NAME=$(git config -f /home/$_USER/.gitconfi
        [ -z "$GIT_AUTHOR_NAME" ] && GIT_AUTHOR_NAME="$_USER"
        export GIT_AUTHOR_NAME
    fi

    if [ -z "$GIT_AUTHOR_EMAIL" ]; then
        [ -f /home/$_USER/.gitconfig ] && GIT_AUTHOR_EMAIL=$(git config -f /home/$_USER/.gitconf
        [ -z "$GIT_AUTHOR_EMAIL" ] && GIT_AUTHOR_EMAIL="$_USER@$_HOSTNAME"
        export GIT_AUTHOR_EMAIL
    fi

    if [ -z "$GIT_COMMITTER_EMAIL" ]; then
        export GIT_COMMITTER_EMAIL=`whoami`"@$_HOSTNAME"
    fi

    trap _etckeeper EXIT
fi
```

Fork-Bomb und andere feine Sachen

Klassische Fork-Bomb. Es werden immer weitere Prozesse gespawned bis der Kiste der Speicher ausgeht (dann schlägt zwar meist der OOM-Killer zu, kann aber auch nicht mehr viel ausrichten):

```
:(){ :|:& };:
```

Kernel Sysrq nutzen:

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

kaputtes Kernel-Modul bauen und laden (ungetestet):

```
mkdir /tmp/kpanic && cd /tmp/kpanic && printf '#include <linux/kernel.h>\n#include <linux/module
```

Laufzeit eines Cronjobs begrenzen (Timeout)

Um zu verhindern, dass Cronjobs mehrfach parallel laufen (da die Ausführung zu lange dauert) gibts viele Möglichkeiten. Die Beliebteste ist der Einsatz von Lockfiles (beim Start eines Jobs wird geprüft, ob eine bestimmte Datei vorhanden ist, falls ja, wird die Ausführung ausgesetzt).

Eine weitere Möglichkeit ist das Tool „timeout“. Über Parameter wird festgelegt, wie lange ein Job laufen darf und mit welchem Signal er beim Timeout abgebrochen wird.

Dazu wird dem auszuführenden Job einfach der Aufruf von timeout vorangesetzt:

```
 * * * * * /bin/timeout --preserve-status -s SIGINT 10 /mein/cronjob.sh
```

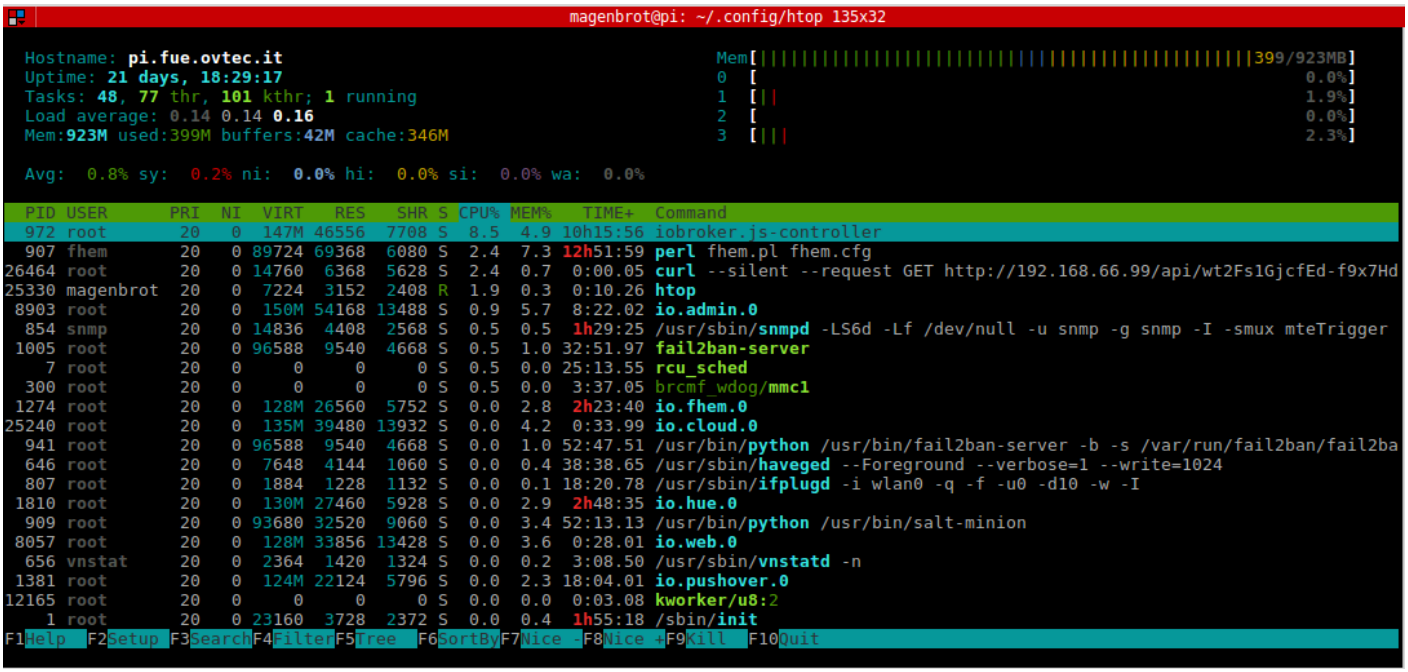
- *–preserve-status*: damit sendet timeout den gleichen Exitstatus wie das aufgerufene Script bei Beendung
- *-s SIGINT*: das Signal was an den Prozess geschickt werden soll (auch als Zahl, siehe unten)
- *10* (timeout): nach XX Sekunden wird das Signal geschickt

Folgende Signale stehen zur Verfügung:

```
~# kill -l
 1) SIGHUP  2) SIGINT  3) SIGQUIT  4) SIGILL  5) SIGTRAP
 6) SIGABRT  7) SIGBUS  8) SIGFPE  9) SIGKILL 10) SIGUSR1
11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP
21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR
31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

Mehr Infos aus htop herauskitzeln

Hier ist meine htop-config, die für die folgende Übersicht sorgt:



Die Config muss in `~/.config/htop/htoprc` abgelegt werden:

~/.config/htop/htoprc

```
# Beware! This file is rewritten by htop when settings are changed in the interface.
# The parser is also very primitive, and not human-friendly.
fields=0 48 17 18 38 39 40 2 46 47 49 1
sort_key=46
sort_direction=1
hide_threads=0
hide_kernel_threads=0
hide_userland_threads=0
shadow_other_users=0
show_thread_names=1
highlight_base_name=1
highlight_megabytes=1
highlight_threads=1
tree_view=0
header_margin=1
detailed_cpu_time=1
cpu_count_from_zero=1
update_process_names=0
account_guest_in_cpu_meter=1
color_scheme=0
delay=15
left_meters=Hostname Uptime Tasks LoadAverage Memory Blank CPU
left_meter_modes=2 2 2 2 2 2 2
right_meters=Memory AllCPUs
right_meter_modes=1 1
```

Swapfile anlegen und einrichten

Swapfile auf gemounteter Disk anlegen und verwenden:

```
dd if=/dev/zero of=/var/swapfile.1 bs=1M count=2048

chmod 0600 /var/swapfile.1

mkswap /var/swapfile.1

echo "/var/swapfile.1 none swap sw 0 0" >> /etc/fstab

swapon -a
```

Damit ist der Swap eingerichtet. Prüfen mit:

```
free -m
             total       used       free     shared    buffers     cached
Mem:         14009       8416       5593          8         20       4158
-/+ buffers/cache:       4237       9772
Swap:         2047          0       2047
```

Cronjob jeweils am letzten Tag des Monats ausführen

Mit Cron Bordmitteln läßt sich der letzte Tag des Monats nicht direkt ermitteln. Dafür nutzen wir einfach einen Bash-Test:

```
  1 9 28-31  *   *    [ $(date +\%d -d tomorrow) = 01 ] && /usr/local/sbin/billing.sh | mail -s "Disk Usage" -c billing@example.com
```

Zombie-Prozesse

Was ist ein Zombie-Prozess?

Prozesstabelle ankucken mit: ps ax

```
2081    pts/2    Z    0:00    [wait <defunct>]
```

An dem Status Z kann man erkennen, dass es sich um einen Zombie-Prozess handelt. Zombie-Prozesse entstehen, wenn sich ein Kindprozess beendet hat, ohne das der Elternprozess auf sie wartet. Oder etwas genauer:

Prozesse verwenden zum Beenden die return-Anweisung oder rufen die Funktion exit() mit einem Wert auf, der an das Betriebssystem zurückgeliefert wird. Das Betriebssystem lässt den Prozess so lange in seiner internen Datentabelle eingetragen, bis entweder der Elternprozess des Prozesses den zurückgelieferten Wert liest oder der Elternprozess selbst beendet wird.

Ein Zombie-Prozess ist in diesem Sinne ein Prozess, der zwar beendet wurde, dessen Elternprozess den exit-Wert des Kindprozesses aber noch nicht erhalten hat. Erst wenn der Elternprozess beendet wird, wird auch der Zombie-Prozess aus der Prozesstabelle des Betriebssystems entfernt.

Blocks berechnen

Wieviele Blocks wievielen MB entsprechen wird folgendermaßen berechnet:

zuerst muss man die Blocksize des Dateisystems herausfinden. Für ext2 geht das z.B. so:

```
# dumpe2fs /dev/sdb3 | grep -i 'Block size'
Block size:               4096
```

Um jetzt etwa ein Quota von 10MB zu setzen muss z.B. 40960 eingetragen werden.

Seit wann läuft ein Prozess?

Manchmal ist es nützlich herauszufinden seit wann ein bestimmter Prozess eigentlich schon läuft. Mit diesem Einzeiler ist das kein Problem:

```
ps -o stime,time <pid>


# Beispiel
ps -o stime,time 2863529
STIME     TIME
Mai01 21:15:43
```