

Starting & stopping

shutdown -h now
Shutdown the system now and do not reboot.

halt Stop all processes.

shutdown -r 5
Shutdown the system in 5 minutes and reboot.

reboot Stop all processes and reboot.

Filesystem

fdisk -l List the partition tables.

mount -t vfat /dev/sda1 /mnt/usb
Mount a USB-Stick.

df -h --sync Report filesystem disk space usage.

du -sh /home/user Estimate file space usage.

du -x --max-depth=0 * | sort -rg
File space usage of all directories in current.

stat file Report file status (-z) for SELinux.

Cryptoloop

- **modprobe cryptoloop && lsmod | grep cryptoloop**
Load the necessary kernel module.
- **modinfo /lib/modules/2.6.6-1.435/kernel/crypto/***
List the available crypto modules.
- **dd if=/dev/zero of=bs=1M count=639 of=/root/.cryptofs**
Create the container file with the desired size.
- **losetup -e twofish /dev/loop0 /root/.cryptofs**
Prepare the loop-device.
- **mkfs.ext3 /dev/loop0**
Create the filesystem.
- **mount -t ext3 /dev/loop0 /mnt/crypto/**
Mount the loop-device

For more security:

- **losetup -e twofish /dev/loop0 /root/.cryptfs**
- **losetup -e blowfish /dev/loop1 /dev/loop0**
- **mount -t ext3 /dev/loop1 /mnt/crypto/**

Using Cryptoloop on DVDs

- **dd if=/dev/zero of=image-crypt.ext3 bs=1M count=650 (or count=4069)**
- **losetup -e blowfish /dev/loop0 image-crypt.ext3**
- **mkfs.ext3 -v -j -b 1024 /dev/loop0**
- **tune2fs -m 0 /dev/loop0**
- **tune2fs -c 0 /dev/loop0**

For checking the fs: **fsck.ext3 -f /dev/loop4**

- **losetup -d /dev/loop0**
- **mount image-crypt.ext3 /mountdir -o loop,encryption=blowfish**
(do copy actions)
- **umount /mountdir/**
- **cdrecord -vv speed=10 -data image-crypt.ext3**
Burn CD
- **growisofs -Z /dev/hdc=image-crypt.ext3**
Burn DVD
- **mount /dev/hdc /mountdir -o encryption=blowfish**
mount

Create, edit, delete, move, copy, show, list, search files

touch file, echo "" > file Create an empty file.

vi file Edit file.

h/l left/right, **j/k** up/down, **:w** write, **:q** quit
:wq! write,quit,override, (**ESC**) **u** undo, **Ctrl+r** redo, **dd** delete line, **dw** delete word, **Ctrl+g** show stats, **Shift+g** go to eof, **\$** go eol, **Shift+a** append to eol.

cat > file <<
Write to **file** until "." on a line by itself.

rm -rf *.* Remove recursively, forced.

rm -i ./-file remove file beginning with a "-"

shred -u *.* A more secure way, -u truncate & remove.

mv -v -b dir1/* dir2/ Move files/subdirs from **dir1** to **dir2**, create backup from overwritten files.

cp -pR dir1/ dir2/ Copy **dir1** to **dir2** recursively, preserving the attributes.

less file Displays file, commands are based on both **more** and **vi**, searching **/pattern**, **<space>** next screen, **b** screen before.

cat file Print file on the standard output.

zcat file The same for compressed files.

zcat file.gz | wc -c Show uncompressed filesize.

tail -f file Output the last part of **file**, following appended data.

head file Output the first part of **file**.

ls -alt List directory contents, newest first.

ls -lt | head List newest 10.

ls -lSHr

file file Determine file type.

find / -type f -empty
Find empty files.

find / -type d -maxdepth 3
List all directories, descend max. 3 levels.

find . -type f -follow -exec md5sum {} 2>/dev/null \; | sort | uniq -w 32 -D
Find duplicate files in current directory.

whereis php
Locate the binary, source & manual page for a command.

locate file Search for **file** on your system

grep -Hirs pattern * Search for a **pattern** in files.

for file in *; do iconv -f ISO-8859-1 -t UTF-8 \$file > ./UTF/\$file; done

iconv Convert encoding of given file(s)

Directories

mkdir -p dir/subdirA/subdirB
Create missing parent directories as needed.

rmdir -p dir/subdirA/subdirB/subdirC
Remove all directories in the path.

Installing, uninstalling

rpm -ivh *.rpm
Install all rpm-packages in current dir.

rpm -Fvh *.rpm
Freshen all installed packages if needed.

rpm -qa | grep kernel

rpm -q -i kernel

rpm -qf /etc/php.ini

rpm -qa --queryformat "%{N} %{V}-%{R} %{INSTALLTIME:date}\n" | grep php

rpm -aq --last | more

rpm -qd List documentation files in rpm.

rpm -e package Remove package(s).

rpm2cpio paket.rpm | cpio -i -d extract rpm in current directory

tar & similar

tar -czvf file.tar.gz /dir/* --totals
Create a gzipped tar-file, showing total bytes written.

tar -uvf file.tar * --totals
Update, add newer files.

tar -xzvf file.tar.gz Extract gzipped tar-file.

Date & Time

date -d '1970-01-01 GMT 1110013633 seconds'
Show date in human readable format.

date +%s Show seconds since 1970-01-01.

(burning) CD, DVD

mkisofs -o file.iso -r /dir/subdir/
Prepare a iso-file for burning.

mkisofs -R -J -T -v /dir/subdir/ | aespipe -H sha256 -e aes256 -T >image.iso
generate a crypted iso-file with aespipe

growisofs -Z /dev/hdc=/path/to/image.iso
burn to DVD

Mounting the crypted CD/DVD

losetup /dev/loop4 /dev/hdc

cryptsetup -c aes -s 256 -h sha256 create cdrom /dev/loop4

mount /dev/mapper/cdrom /mount/to/dir

cdrecord -vv speed=8 dev=X,Y,Z -data .cryptofs driveropts=burnproof
Burning .cryptofs container-file.

dd if=/dev/hdc | cdrecord -v speed=16 dev=0,0,0 fs=16m -data -
Copy cd on the fly.

tar -czvf - /etc/home/user | cdrecord -ejct dev=1,0,0 -
Burn on the fly.

Processes & services

ps -eaf Report process status.

kill -9 pid
terminate the process with pid

fuser /mnt/cdrom
show processes that using a file/device

/dev/null, /dev/zero and other devices

command > /dev/null 2>&1

command &> /dev/null
Redirecting **stderr** and **stdout** to **/dev/null**.

dd if=/dev/random bs=6 count=1 | uencode -m /dev/stdout generate random password

Security

poor man's tripwire

- **find /dir1 /dir2 -type f > filelist.txt**
- **cat < filelist.txt | xargs md5sum > md5filelist.md5.txt**
- **md5sum -c md5filelist.md5.txt > differences.txt**

Check for modified files

rpm -qV package

rpm -qf file

SSH

ssh -c blowfish -C user@192.168.0.1
login via ssh, using blowfish cipher, compressing all data

ssh -C -c blowfish user@192.168.0.1 -N -f -g -L 1080:192.168.0.1:8080
Forward local port 1080 to remote port 8080, using ssh encrypt, decrypt files with openssl

openssl bf -in file.clear -out file.crypt -e -salt
encrypt, blowfish

openssl bf -in file.crypt -out file.clear -d -salt
decrypt, blowfish

encrypt, decrypt files with gpg

gpg -c --cipher-algo BLOWFISH -a file.txt
encrypt, blowfish

gpg -a --cipher-algo BLOWFISH -d file.txt.asc

Miscellaneous, combining commands

(cd /src/dir && tar cfls - .) | (cd /dst/dir && tar xfp -)
coping with tar

- **tar -cf - files | ssh user@server "tar -xf -"**
- **tar czvf - /dir | ssh user@server "cat > file.tar.gz"**
copy to server
- **ssh user@server "tar -cf - files" | tar -xf -**
ssh user@server "cat file.tar.gz" | tar zpvxf -
copy from server

copy from server and burn local DVD

- **ssh user@server 'mkisofs -joliet /path/*.ext' | growisofs -Z /dev/dvd=/dev/stdin**

find / -type f -size +10000k -iname "*.pdf" -exec gzip -9 \{\} \;
find all pdf-files bigger than 10000k & gzip them

find -name ".*.1" | while read f; do mv "\$f" `basename "\$f" .1`.2 ; done
as shell-script mmv: **mmv jpg jpg.sik**

For more information on commands/options

man -a command display all on-line manual page, show next manual with "q"

info command

pinfo command

apropos command or **man -k command**

DVD

Copy a DVD to disk

```
vobcopy -i /dev/cdrom -o /target/directory/ -v
```

Rip a DVD

```
mencoder dvd://1 -chapter 2-2 -vf scale -zoom -xy 640 -alang de -o  
dvd-title-1-chapter2.avi -ovc lavc -lavcopts  
vcodec=mpeg4:vhq:vbitrate=1200 -oac mp3lame
```

vi

example of a transparent gpg-encryption with vim
put in the .vimrc

```
" Transparent editing of gpg encrypted files.  
" By Wouter Hanegraaff <wouter@blub.net>  
" enhanced by Egon Leitner  
augroup encrypted  
au!  
" First make sure nothing is written to ~/.viminfo while editing  
" an encrypted file.  
" viminfo doesn't have local value, set global value instead  
autocmd BufReadPre,FileReadPre,BufNewFile *.gpg,*.asc set viminfo=  
" We don't want a swap file, as it writes unencrypted data to disk  
autocmd BufReadPre,FileReadPre,BufNewFile *.gpg,*.asc set noswapfile  
"autocmd BufReadPre,FileReadPre *.gpg, *.asc set uc=0  
" Switch to binary mode to read the encrypted file  
"autocmd BufReadPre,FileReadPre *.gpg set bin  
autocmd BufReadPre,FileReadPre *.gpg,*.asc let ch_save = &ch|set ch=2  
autocmd BufReadPost,FileReadPost *.gpg,*.asc '[,']!sh -c 'gpg --decrypt 2> /dev/null'  
"autocmd BufReadPost,FileReadPost *.asc '[,']!sh -c 'gpg --decrypt 2> /dev/null'  
" Switch to normal mode for editing  
"autocmd BufReadPost,FileReadPost *.gpg set nobin  
autocmd BufReadPost,FileReadPost *.gpg,*.asc let &ch = ch_save|unlet ch_save  
autocmd BufReadPost,FileReadPost *.gpg,*.asc execute ":doautocmd BufReadPost " . expand("%:r")  
" Convert all text to encrypted text before writing  
autocmd BufWritePre,FileWritePre *.gpg '[,']!sh -c 'gpg --default-recipient-self -e -a 2> /dev/null'  
autocmd BufWritePre,FileWritePre *.asc '[,']!sh -c 'gpg --default-recipient-self -e -a 2> /dev/null'  
" Undo the encryption so we are back in the normal text, directly  
" after the file has been written.  
autocmd BufWritePost,FileWritePost *.gpg,*.asc u  
augroup END
```