Unix Memos Collection

Marc Zonzon

Sep 03, 2019

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

This Unix Memo is composed of some notes on Unix, or mainly Linux, administration. Some are very fragmentary, some are obsolete, or at least not up to date, probably they contain some erroneous content, to sum up they are unreliable.

If nevertheless you chance up over some useful content, I'm pleased to share it. And if you want to improve the memo you are welcome to submit a pull request in the GitHub repository.

CHAPTER

TWO

LINUX COMMANDS MEMO

This is a linux command line reference for common operations. The network commands are in a separated *Network Commands Memo*.

2.1 basic

apropos whatis	Show commands pertinent to string.
man -t ascii | ps2pdf - > ascii.pdf	make a pdf of a manual page
which command	Show full path name of command
time command	See how long a command takes
time cat	Start stopwatch. Ctrl-d to stop.

2.2 files

ls -la > dirlist 2>&1	Redirect both stdout and stderr to a file.
rename 's/.jpeg$/.jpg/' *.jpeg	Mass rename with perl rename command
rename .jpeg .jpg *.jpeg	Mass rename with util-linux rename command
find ./ -type f -print | xargs chmod 640	Change permissions to 640 for all files in subtree.
find ./ -type d -print | xargs chmod 751	Change permissions to 751 for all sub-directories.
shred -u private.txt	delete a file from disk after securely erasing the content.

2.3 dir navigation

cd -	Go to previous directory
cd	Go to $HOME directory
(cd dir && command)	Go to dir, execute command and return to current dir
pushd *dir*	Put *dir* on stack so you can **popd** back to it
pushd +3	Rotate the dir stack, putting third entry at top

2.4 File searching

ls -lt	List files by date, newest first
ls /usr/bin | pr -T9 -W$COLUMNS	Print in 9 columns to width of terminal
find -maxdepth 1 -type f -print0 | xargs -0 ls -lS –block-size=1k	List files by decreasing size
find -size +1M -ls	List files bigger than 1 Megabyte.
find -name '*.[ch]' | xargs grep -E 'expr'	Search 'expr' in this dir and below.
find -type f -print0 | xargs -r0 grep -F 'example'	Search all regular files for 'example' in this dir and below
find -maxdepth 1 -type f | xargs grep -F 'example'	Search all regular files for 'example' in this dir
find -maxdepth 1 -type d | while read dir; do echo $dir; echo cmd2; done	Process each item with multiple commands (in while loop)
find. -xtype l	Find broken links
find -type f ! -perm -444	Find files not readable by all (useful for web site)
find -type d ! -perm -111	Find dirs not accessible by all (useful for web site)
locate -r '*file*.txt'	Search cached path index for names.
locate -r 'file[^/]*.txt'	Search cached path index for names. 'file' must be in last component.

2.5 disk space

ls -lkS	Show files by size in kb, biggest first.
ls -lt	sort by modification time, newest first
du -sh * | sort -k1,1rh | head	Show larger directories in current dir.
sudo du -hs /home/* | sort -k1,1h	Sort paths by increasing use
du -ah –max-depth=0 * | sort -k1,1rh | head -n 15	Show 15 larger directories or files in current dir.
df -h	Show free space on mounted filesystems
df -i	Show free inodes on mounted filesystems
sudo sfdisk -l /dev/sda	Show disks partitions sizes and types (MBR part)
sudo sgdisk -p /dev/sda	Show disks partitions sizes and types (GUID part)
dd bs=1 seek=2TB if=/dev/null of=ext3.test	Create a large sparse test file (taking no space).
>| file	truncate data of file or create an empty file

2.6 text handling

tr -dc '[:print:]' < /dev/urandom	Filter non printable characters
echo "(33) 06.61 62-63+84" | tr -d [:blank:][:punct:]	clean a phone number string
tr -s '[:blank:]' 't' </proc/diskstats | cut -f4	cut fields separated by blanks
tr -s '[:blank:]' </proc/diskstats | cut -d' ' -f4	cut fields separated by blanks
wc -l file	count lines (`w` words, `-b` bytes)
cut -d: -f1 /etc/passwd | sort	Lists all usernames in alphabetical order.
dd if=/dev/urandom count=1 | base64 -w 0 | cut -c 1-16	generate random 16 chararacters password
openssl rand -base64 16 | cut -c 1-16	generate random 16 chararacters password
tr -dc '[:alnum:]&~#|_@=+$%*<>,?;.:/!-' < /dev/urandom \| head -c${1:-16}; echo	generate random 16 chararacters password
date +%s | sha1sum'|:coreutils:'cut -f1 -d' '	generate new 4O alphanumeric chars password
paste -d ',:' file1 file2 file3	Merges given files line by line
mount | column -t	table of mounted filesystems
join -t'0' -a1 -a2 file1 file2	Union of sorted files
join -t'0' file1 file2	Intersection of sorted files
join -t'0' -v2 file1 file2	Difference of sorted files
join -t'0' -v1 -v2 file1 file2	Symmetric Difference of sorted files
column -s, -t <tmp.csv	pretty print csv
printf "%03o\n" "'%"	octal code of ascii character `%`
printf "Ox%02x\n" "'%"	hexacimal code of ascii character `%`
printf "%d\n" "'%"	decimal code of ascii character `%`
iconv -f ISO8859-1 -t UTF-8 -o file.utf8 file.txt	convert encoding
iconv -l	List known coded character sets
sha1sum file	checksum of a file (use also other sums: `sha256sum`, `sha512sum`, `md5sum`)
sha1sum -c checksumlist	check the sums against the files

2.7 encryption

gpg -c file	Encrypt file. More commands in the *GnuPG Memo*.
gpg file.gpg	Decrypt file.
openssl -h	Help including available ciphers
openssl list-cipher-commands	long list of available ciphers
openssl enc -aes-256-cbc -salt -a	encrypt stdin to stdout using 256-bit AES in CBC mode, and encode in base64
openssl enc -aes-256-cbc -salt -in file.txt -out file.enc	encrypt to *binary* file.enc using 256-bit AES in CBC mode
openssl enc -d -aes-256-cbc	decrypt binary data on stdin
openssl enc -d -aes-256-cbc -a -in file.enc	decrypt base64 encoded file
openssl enc -aes-256-cbc -salt -a -pass file:/path/to/password.txt	encrypt stdin to stdout, provide password in a file

2.8 archives and compression

tar -cjf dir.tar.bz2 dir/	Make bzip2 compressed archive of dir/
tar -jxf dir.tar.bz2	Extract archive (replace **j**, by **z** for gzip, or lzip)
tar -cxf dir.tgz –exclude '*.o' –exclude '*~' dir/	
tar -xf dir.tgz –to-stdout dir/file.txt	Print file to stdout
tar -c dir/ | gzip | gpg -c | ssh user@remote 'dd of=dir.tar.gz.gpg'	Make encrypted archive of dir/ on remote machine.
find dir/ -name '*.txt' | tar -c –files-from=- | bzip2 > dir_txt.tar.bz2	Make archive of subset of dir/ and below.
find dir/ -name '*.txt' | xargs cp -a –target-directory=dir_txt/ –parents	Make copy of subset of dir/ and below.
(tar -c /dir/to/copy) | (cd /where/to/ && tar -x -p)	Copy (with permissions) copy/ dir to /where/to/ dir
(cd /dir/to/copy && tar -c .) | (cd /where/to/ && tar -x -p)	Copy (with permissions) contents of copy/ dir to /where/to/
(tar -c /dir/to/copy) | ssh -C user@remote 'cd /where/to/ && tar -x -p'	Copy (with permissions) copy/ dir to remote:/where/to/ dir
zip -r /path/to/archive.zip dir	zip a directory
unzip archive.zip	extract archive
unzip -l archive.zip	list archive content
unzip archive.zip file.txt	Extract one file from archive
dd if=/dev/vg0/vol0 of=/dev/vg1/vol1 bs=4096	Copy a partition to another one (bs must be a divider of volume blocksize)
dd bs=1M if=/dev/sda | gzip | ssh user@remote 'dd of=sda.gz'	Backup harddisk to remote machine.
dd bs=4096 if=/dev/vgsource/root_snapl ssh -c 'chacha20-poly1305@openssh.com' rootr@remote dd bs=4096 of=/dev/vgremote/root_copy	copy a partition to remote machine
dd bs=4096 if=/dev/vg0/root_snapl | ssh-c 'chacha20-poly1305@openssh.com' root	
killall -s USR1 dd	Ask dd to print the state of the current transfer.

2.9 process management

ps axww	list all processes
ps axuww	list all processes and resource used
ps axmu	list all processes and threads
ps axf -o pid,args	List processes in a hierarchy.
ps ax -o pcpu,cpu,nice,state,cputime,args –sort -pcpu | sed '/^ 0.0 /d'	List processes by decreasing cpu rate (see also top).
ps ax -opid=,rss=,args= –sort=+rss | sed '/^s*0>/d' | pr -TW$COLUMNS	List processes by mem (KB) usage (see also top).
ps -o user –sort user| uniq -c| sort -n -k1	number of processes per user.
ps -C lighttpd -o pid=	pid of *lighttpd*.
pgrep light	pid of processes having *light* in their name.
pgrep -a daemon	pid/command-line of all processes having *daemon* in their name
pidof lighttpd	pid of *lighttpd*.
ps uw -C lighttpd	user oriented list of process *lighttpd*.
ps -C firefox-bin -L -o pid,tid,pcpu,state	List all threads for a particular process.
ps -p 666 -o etime=	List elapsed wall time for process id 666
ps ew 666	show command and environment of process 666
kill -9 1234	Send SIGKILL to process 1234
killall -s USR1 dd	Send signal USR1 to the dd program
pkill -s USR1 dd	Send signal USR1 to the dd program
pmap 1234	Memory map of process 1234

2.10 monitoring, process admin

tail -f /var/log/messages	Monitor messages in a log file.
less +F /var/log/messages	Monitor messages in a log file.
lsof -p 666	List paths that process id 666 has open.
lsof /path/to/file	List processes that have specified path open.
lsof -u foo	Processes and files of user foo
lsof -u foo	Processes no of user foo
lsof -t -c pcmanfm	files open by pcmanfm
fuser -va 22/tcp	List processes using port 22
fuser -va /home	List processes accessing the /home
sudo tcpdump not port 22	Show network traffic except ssh.
sudo tcpdump -ni eth0 'dst 192.168.1.5 and tcp and port http'	all HTTP session to 192.168.1.5.
last reboot	Show system reboot history.
free -m	Show amount of (remaining) RAM (-m displays in MB)
watch -n.1 'cat /proc/interrupts'	Watch changeable data continuously.
watch -t -n1 uptime	Clock with system load.
nice *command*	Low priority *command*.
sudo renice 19 -p 666	Set process 666 to low scheduling priority (0<pr<20)
sudo renice +2 -p 666	Lower the scheduling priority.
chrt -i 0 *command*	Low priority command (more effective than nice)
sudo ionice -p 666	io class and priority of process 666. Higher priority 0
sudo ionice -c3 -p 666	Sets process 666 as an idle io process.
htop -d 5	Better top (scrollable, tree view, lsof/strace integration, . . .)
iotop	What's doing I/O.
sudo iftop	What's using the network.
vmstat 3	monitor processes, memory, paging, block IO, traps, and cpu activity.(columns are explained in the manual.)
vmstat -m	usage of kernel dynamic memory.

2.11 Users

id -a	Show the active user id with login and groups.
last	Show last logins on the system.
w	users logged on, and their processes.
groupadd admin	Add group "admin"
useradd -c "Linus Torvald" -g admin -m linus	Add new user
usermod -a -G sudo linus	add group "sudo" to linus groups.
adduser –uid 3333 linus	Add new user, with interactive prompt, create home dir.
userdel linus	Delete user linus

2.12 system information

uname -a	Show kernel version and system architecture.
cat /etc/debian_version	Get Debian version
lsb_release -a	Full release info of any LSB distribution
cat /etc/issue	Show name and version of distribution.
cat /proc/partition	Show all partitions registered on the system.
grep MemTotal /proc/meminfo	Show RAM total (see also *free*, *vmstat*)
cat /proc/cpuinfo	Show CPU(s) info
lscpu	Show CPU(s) info
lsdev	hardware info from the /proc directory
sudo lspci -tv	Show PCI info
sudo lshw	Show hardware configuration of the machine
sudo hwinfo	Show hardware configuration of the machine
lsusb -tv	Show USB info
mount | column -t	List mounted fs on the system (and align output)
grep -F capacity: /proc/acpi/battery/BAT0/info	Show state of cells in laptop battery
dmidecode -q | less	Display SMBIOS/DMI information
dumpe2fs -h /dev/part1 | grep -e '\([mM]ount\)|\([Cc]heck\)'	info about fs check
sudo e2fsck -f -v -t -C 0 /dev/part1	Check health of partition
sudo sdparm -C stop /dev/sdb	Stop scsi (also usb) disk
sudo hdparm -i /dev/sda	Show info about disk sda
dmesg	Detected hardware and boot messages

2.13 sed

See sed manual and sed1line.

`sed -n '8,12p'`	Print lines 8 to 12
`sed -n '/regexp/p'`	Print lines which match regular expression
`sed '/regexp/d'`	Print lines which don't match regular expression
`sed -n '/begregexp/,/endregexp/p'`	Print section of file between two regexp
`sed '/begregexp/,/endregexp/d'`	Print file except section between two regexp
`sed '/^#/d; /^ *$/d'`	Remove comments and blank lines
`sed -i 's/[ \t]\*$//' file.txt`	Delete trailing space at end of lines
`sed -e :a -e '/^\n*$/N;/\n$/ba'`	Delete blank lines at end of file.
`sed -i 42d ~/.ssh/known_hosts`	Delete a particular line
`sed ':a; /\\$/N; s/\\\n//; ta'`	Concatenate lines with trailing \
`sed = filename | sed 'N;s/\n/\t/'`	Put a left count number on each line of a file
`sed = filename | sed 'N; s/^/ /; s/ *\(.\{6,\}\)\n/\1 /'`	Put a right aligned count on each line
`sed 's/\x0D$//'`	Dos to unix eol
`sed 's/$/\\r/'`	Unix to dos eol

2.14 File attributes, Extended Attributes and ACL

They are three different sets of attributes that can be supported from filesytems. For more details look at the File attributes section and the ACL section.

Note: for ext 2/3/4 fs you may need to (re)mount with "acl" or "user_xattr" options. Or set the filesystem default with tune2fs. On btrfs acl and xattr are enabled by default.

getfacl foo	Show ACLs for file.
setfacl -m u:nobody:r foo.txt	Allow a specific user to read file.
setfacl -x u:nobody foo.txt	Delete a specific user's rights to file.
setfacl –default -m group:users:rw- dir/	Set umask for a for a specific dir.
getcap file	Show capabilities for a program.
setcap cap_net_raw+ep your_gtk_prog	Allow gtk program raw access to network
getfattr -m- -d	Show all extended attributes (includes selinux,acls,. . .)
setfattr -n "user.foo" -v "bar" .	Set arbitrary user attributes

2.15 Desktop management

xset q	display X user preferences.
xset -b	Turn off system beep
xset -b	Turn on system beep
xwininfo	Info of the window selected by mouse click.
xwininfo -name emacs	Emacs window info.
xprop	Xserver properties of the window selected by mouse click.
xdpyinfo	Xserver dimension and resolution.
wmctrl -lG	List managed windows with their geometry.
wmctrl -l -x	List managed windows with their `WM_CLASS`.
wmctrl -d	List desktops, current desktop has a *
wmctrl -s 3	switch to desktop 3
wmctrl -a emacs	switch to desktop containing emacs and raise it.
wmctrl -r emacs -t2	send emacs to third desktop
wmctrl -r emacs -e 0,-1,-1,756,495	resize emacs to 756x495 pixels
xdotool search –onlyvisible –class *emacs* windowsize –usehints %1 80 24	resize emacs to 80 columns x 24 lines.
xwit -columns 80 -rows 24 -names foo	resize *foo* window.
xwit -columns 80 -rows 24 -select	select and resize a window.
xwit -rows 34 -columns 80 -property WM_CLASS -names emacs	resize all emacs windows.

2.16 Images manipulation

The syntax is given for ImageMagick . If you prefer GraphicsMagick just put `gm` before the operation. The the option related to an input file comme before the file name in GraphicsMagick and **after** in ImageMagick.

identify *photo.jpg*	information about an image file
convert *photo.png* -resize 2048x1536 -quality 80 *photo.jpg*	resize an image
convert *apple.jpg* -crop 128×128+50+50 *apple_crop.jpg*	crop an image
convert *lying.jpg* -rotate 90 *standing.jpg*	rotate an image
convert *.jpg ouput.pdf	Create a single PDF from multiple images with ImageMagick
import *snapshot.jpg*	Take a snapshot of a mouse selected desktop area.

2.17 Pdf

gs -dBATCH -dNOPAUSE -sDEVICE=pdfwrite -dFirstPage=2 -dLastPage=2 -sOutputFile=page2.pdf input.pdf	Extract a page from pdf document
pdftk input.pdf burst	Burst a PDF document into pages and dump its data to doc_data.txt
pdfseparate input.pdf p-%d.pdf	separates xx.pdf into separate pages: p-1.pdf, p-2.pdf, . . .
pdfseparate -f 2 -l 3 input.pdf p-%d.pdf	separates from page 2 to page 3: p-2.pdf, p-3.pdf
pdfjam intput.pdf '2,3' –outfile output.pdf	separates pages 2 and 3
qpdf intput.pdf –pages intput.pdf 1-3 –output.pdf	separates pages 2 and 3
gs -q -sPAPERSIZE=a4 -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=all.pdf file1.pdf file2.pdf . . .	Join many pdf files into one.
pdftk in1.pdf in2.pdf cat output out1.pdf	Join two pdf files
pdfunite in1.pdf in2.pdf out1.pdf	Join two pdf files
pdfjam file1.pdf '-' file2.pdf '1,2' file3.pdf '2-' –outfile output.pdf	merge all pages of file1.pdf, page 1 and 2 of file2.pdf and all pages up from page 2 of file3.pdf
qpdf file1.pdf –pages file1.pdf –pages file2.pdf 1-2 –pages file3.pdf 2- – output.pdf	merge all pages of file1.pdf, page 1 and 2 of file2.pdf and all pages up from page 2 of file3.pdf
pdfimages input.pdf img	extracts all images as impg-000.ppm, img-001.ppm,. . .
pdfcrop –margins '5 10 20 30' input.pdf output.pdf	crop a pdf with left, top, right and bottom margins of 5, 10, 20, and 30 pt
pdfjam –trim '1cm 2cm 1cm 2cm' –clip true file1.pdf –outfile output.pdf	crop a pdf with left, top, right and bottom margins of 1cm 2cm 1cm 2cm
pdfjam –nup 2x2 input.pdf –outfile output.pdf	recombines the pdf file to contain 4 pages per page.
pdftk secured.pdf input_pw *mypass* output public.pdf	save a public copy of a password protected file
qpdf –password=*mypass* –decrypt secured.pdf public.pdf	save a public copy of a password protected file

2.18 Refs

- This page is a fork of *pixelbeat* command line reference see also the unix commands page, More Linux commands, the programming notes, the scripts
- Other system command memos: Unix Toolbox, commandlinefu, shell-fu.

CHAPTER

THREE

NETWORK COMMANDS MEMO

3.1 rsync

Use the –dry-run *option for testing and environment* `RSYNC_PARTIAL_DIR=.rsync-tmp` *to keep partial files separates.*

diff -r /path/to/dir1/ /path/to/dir2/	diff recursively two directories.
diff -rq /path/to/dir1/ /path/to/dir2/| sort	list files that differs between two directories
diff -rq /path/to/dir1/ /path/to/dir2/| diffstat	summarize differences between two directories
rsync -avn source-dir/ target-dir/	what files differs (size mod time) between two directories.
rsync -avnc source-dir/ target-dir/	what files differs (checksum) between two directories.
rsync -P rsync://rsync.server.com/path/to/file file	Use partial transfer, repeat for troublesome downloads.
rsync –bwlimit=1000 fromfile tofile	Locally copy with rate limit. It's like nice for I/O
rsync -az –delete ~/public_html/ remote.com:'~/public_html'	Mirror web site (using compression and encryption)
rsync -auz remote:/dir/ . && *rsync* -auz . remote:/dir/	Synchronize current directory with remote one.

3.2 ssh

More info in the *ssh section*.

ssh $USER@$HOST command	Run command on $HOST as $USER (default command=shell)
ssh -f -Y $USER@$HOSTNAME xterm	Run GUI command on $HOSTNAME as $USER
ssh -c 'chacha20-poly1305@openssh.com' -f -Y $USER@$LANHOST xterm	Run GUI command on $LANHOST as $USER with *faster crypto*.
tar -cf- src | ssh -q -c arcfour128 $LANHOST tar -xf- -Cdest	*quick directory transfer*.
scp -p -r -C $USER@$HOST: file dir/	Copy with permissions to $USER's home directory on $HOST, compress for slow links.
scp -c arcfour128 $USER@$LANHOST: bigfile	Use *faster crypto* for local LAN, but *tar over ssh is to be preferred*.
ssh -g -L 8080:localhost:80 root@$HOST	Forward connections to $HOSTNAME:8080 out to $HOST:80
ssh -R 1434:imap:143 root@$HOST	Forward connections from $HOST:1434 in to imap:143
ssh -D 9999 $USER@$HOST	create a SOCKS proxy on localhost and port 9999
ssh-copy-id $USER@$HOST	Install public key for $USER@$HOST for password-less log in

3.3 wget

(cd dir/ && *wget* -nd -pHEKk http://rest-sphinx-memo.readthedocs.org/)	Store local browsable version of a page to the current dir
wget -c http://www.example.com/large.file	Continue downloading a partially downloaded file
wget -r -nd -np -l1 -A '*.jpg' http://www.example.com/dir/	Download a set of files to the current directory
wget ftp://remote/file{[}1-9{]}.iso/	FTP supports globbing directly
wget -q -O- http://www.example.com/page	cat to /dev/stdout
echo '*wget* url' | at 01:00	Download url at 1AM to current dir
wget –limit-rate=20k url	Do a low priority download (limit to 20KB/s)
wget -nv –spider –force-html -i bookmarks.html	Check links in a file
wget –mirror http://www.example.com/	Efficiently update a local copy of a site (handy from cron)

3.4 networking

ethtool eth0	Show status of ethernet interface eth0
ethtool –change eth0 autoneg off speed 100 duplex full	Manually set ethernet interface speed
iwconfig eth1	Show status of wireless interface eth1
iwconfig eth1 rate 1Mb/s fixed	Manually set wireless interface speed
iwlist scan	List wireless networks in range
ip link show	List network interfaces
ip link set dev eth0 name wan	Rename interface eth0 to wan
ip link set dev eth0 address 00:80:c8:f8:be:ef	Change mac address.
ip link set dev eth0 up	Bring interface eth0 up (or down)
ip addr show	List addresses for interfaces
ip addr add 1.2.3.4/24 dev eth0	Add (or del) ip and mask (255.255.255.0)
ip route show	List routing table
ip route add default via 1.2.3.254	Set default gateway to 1.2.3.254
ip route add 192.168.16.0/28 via 192.168.31.254	route subnet
ifconfig -a	List network interfaces
ifconfig eth0 1.2.3.4 up	Bring interface eth0 up (or down)
ifconfig eth0 1.2.3.4 netmask 255.255.255.0	Add first ip and mask
ifconfig eth0:0 1.2.3.5 netmask 255.255.255.0	Add additional ip and mask
route -n	List routing table
route add default gw 1.2.3.254	Set default gateway to 1.2.3.254
route add -net 192.168.16.0 netmask 255.255.240.0 gw 192.168.31.254	route subnet
host github.com	Lookup DNS ip address for name or vice versa
hostname -i	Lookup local ip address (equivalent to host 'hostname')
whois mzlinux.org	Lookup whois info for hostname or ip address
sudo netstat -tupl	List internet services on a system
sudo netstat -tup	List active connections to/from system
sudo ss -tup	List tcp and udp active connections to/from system
iptraf	interactive ncurses colorful IP LAN monitor.
vnstat	Console hourly, daily and monthly network traffic.
lsof -i tcp:443	What tcp connection is using port 443.
lsof -i :5800	What is using port 5800.
lsof -i @192.168.1.5:22	connections to host 192.168.1.5 port 22
curl -I https://github.org	Display the server headers for a web site.

Continued on next page

Table 1 – continued from previous page

curl -s https://ftp-master.debian.org/keys/archive-key-7.0.asc | gpg –import	Import a gpg key from the web
curl ifconfig.me	get your external address through ifconfig.me
sudo apache2ctl -S	Display a list of apache virtual hosts

3.5 network manager

nmcli	state of network-manager including Wireless Access Points
nmcli dev status	status of all devices
nmcli dev show	details of all devices and connections
nmcli dev show *eth0*	details of a specific device.
nmcli radio wifi off	disable wifi
nmcli con show	list of registered connections
nmcli con show –active	list of active connections
nmcli con show *df67bf87-452f-4a03-af4e-60f31afd749a*	show connection details by uuid
nmcli con show id *myaccess_pt*	show connection details by id
nmcli con -f IP4.dns con show eth0-dhcp	dns of a connection
nmcli –show-secret –field 802-11-wireless-security.psk con show id *myaccess_pt*	show a connection password
nmcli connection del id *'my access pt'*	delete connection
nmcli con modify id *old_id* connection.id *new_id*	change connection id
nmcli con up id *MyWifi* password *mypasswd*	connect with password
nmcli con edit *conid*	interactive edit the connection
nmcli dev wifi list	List available Wi-Fi access points.
nmcli -p dev wifi list	pretty list of all availables wifi access points
nmcli dev wifi connect *FreeWifi*	setup and activate a new connection
nmcli dev wifi connect *apssid* name *conname* password *private*	new connection with name and password
nmcli -f IP4 dev show eth0	IPV4 adress, gateway, and dns
nmcli -f general.connection dev show eth0	active connection on an iface
nmcli dev disconnect *wlan0*	disconnect *wlan0* interface
curl -F login=*myid* -F password=*mypasswd* *https://wifi.provider.org/Auth*	Connect to open spot

CHAPTER

FOUR

SHELL MEMO

I only give some points which are sometime tricky in shell. There are many good shell scripting guides, to which you can refer.

The following recipes are usually in portable shell, when it is specifically for bash it is indicated.

For the numerous differences betwween bash and bourne shell see Bashism - How to make bash scripts work in dash to transform your scripts you can use Autoconf manual - Portable Shell Programming .

Most of the constructs given here for bash work also in the same way for zsh which is very similar for scripting. bash and zsh differences are mainly in the interactive. use It is explained in this linux Magazine article and in zsh wiki.

I summarise the scripting differences in the next paragraph.

4.1 Differences between zsh and bash scripts.

- zsh does not allow `${var/#old/new}` and `${var/%old/new}` for anchoring the match of old to the start or end of the parameter text, respectively. It is the only difference mentioned in the zsh FAQ.
- While bash array index begin by `0` those of zsh begin by `1`. More details in Compatibility between Zsh and Bash.
- There are some tricky differences in pattern matching. In both shell with the double `[[ ... ]]` operator the comparison for ==, !=, =~ is done whith shell patterns.

The commands:

```
$ if [[ "abc" = a*c ]]; then echo true; else echo false;fi
$ if [[ "abc" == a*c ]]; then echo true; else echo false;fi
$ [[ "abc" = a*c ]] && echo true || echo false
$ [[ "abc" = a*d ]] && echo true || echo false
```

work in the expected way both zsh and bash.

But in zsh we have:

```
$ if [ "abc" == a*c ]; then echo true; else echo false;fi
zsh: = not found
$ if [ "abc" = a*c ]; then echo true; else echo false;fi
zsh: no matches found: a*c
$ if [ a*c == a*c ]; then echo true; else echo false;fi
zsh: = not found
$ if [ a*c = a*c ]; then echo true; else echo false;fi
zsh: no matches found: a*c



While in bash:
```

```
$ if [ "abc" = a*c ]; then echo true; else echo false;fi
false
$ if [ "abc" == a*c ]; then echo true; else echo false;fi
false
$ if [ a*c = a*c ]; then echo true; else echo false;fi
true
$ if [ a*c == a*c ]; then echo true; else echo false;fi
true
```

As the single `[` does not make pattern matching in bash and the comparison is on the raw strings.

But note that all these last lines are *not corrects*, as we are using single `[ ... ]`; in bash the comparison is on *strings* not patterns; so the right way was to quote the stings and use a single =.

And the right expression

```
$ if [ "abc" = "a*c" ]; then echo true; else echo false;fi
false
$ if [ "a*c" = "a*c" ]; then echo true; else echo false;fi
true
```

Is correct in any shell: bash, zsh, or dash.

4.2 `exit` and `return`

Refs: Bash Reference Manual: Bourne Shell Builtins

4.2.1 return

`return [n]` return n or if not supplied the exit status of the last command. In a function it return with the value. When in a script being sourced with the `.` or `source` builtin terminate the script with the return value.

Any command associated with the RETURN trap is executed before execution resumes after the function or script.

A return outside of a function or sourced script, will not terminate the script nor return the value, it produces an error. If the shell is running with `-e` option, it interrupt the script with an error code.

4.2.2 exit

`exit [n]`: Exit the shell, returning a status of n to the shell's parent. If n is omitted, the exit status is that of the last command executed. Any trap on EXIT is executed before the shell terminates.

4.3 Reading a query string

To define variable `a` and `b` with respective values from the `QUERY a=1234&b=9876`

A very simple code that **should be avoided** because the `eval` here are uncontrolled is

```
IFS="&" set -- $QUERY
for l in $@; do
    eval $l
done
```

Read BASH FAQ: Eval command and security issues for an explanation of the danger, and how to avoid it.

A better solution is to use indirect variables or associative arrays, their use is explained in the BASH FAQ: How can I use variable variables or associative arrays?

In bash 4+ we can define an associative array var with each value:

```
declare -A var
IFS="&" set -- $QUERY
for i in $@; do
    IFS="=" set -- $i
    var[$1]=$2
done
```

Chris F.A Johnson gave a more complete function that does uudecode in Parsing Web Form Input in CGI Shell Scripts (also available in Dr. Dobbs)

```
parse_query() #@ USAGE: parse_query var ...
{
    local var val
    local IFS='&'
    vars="&$*&"
    [ "$REQUEST_METHOD" = "POST" ] && read QUERY_STRING
    set -f
    for item in $QUERY_STRING
    do
      var=${item%%=*}
      val=${item#*=}
      val=${val//+/ }
      case $vars in
          *"&$var&"* )
              case $val in
                  *%[0-9a-fA-F][0-9a-fA-F]*)
                       printf -v val "%b" "${val//\%/\\x}"
                       ;;
              esac
              ;;
      esac
    done
    set +f
}
```

You find a more detailled solution using associative bash arrays; and another with indirect variables in Bash FAQ: How do I write a CGI script that accepts parameters?

4.4 Links

4.4.1 Symlinks

Symlinks store in an inode any path in the system hierarchy. The symlink act as a pointer to another file name. The path can be absolute or relative; existing or dangling. Permissions are ignored in symlink inodes.

Unlike hard links, symbolic links can be made to directories or across file systems with no restrictions.

You create symlinks with:

```
$ ln -s existing-path alias-or-directory
$ cp --symbolic-link name1 name2
```

The value of a symlink is returned by:

```
$ readlink name
```

and the absolute path stripped from any symbolic link component, any `.` or `..` or repeated `/` is given by:

```
$ readlink --canonicalize name
$ readlink -f name
```

4.4.2 Hardlinks

In POSIX systems, one file can have many names at the same time. Since hardlinks reference inodes directly, they're restricted to the same file system.

Since they reference the same inode the owner and permissions of to hardlinks are always identical.

You create hardlinks with:

```
$ ln  existing-path alias-or-directory
$ cp --link name1 name2
```

4.4.3 Reflinks

Reflinks are Copy-on-write *COW* of a file; they are available on OCFS2 and Btrfs file systems. Reflinks creates a new inode that shares the same disk blocks as the original file. Reflinks works only inside the boundaries of a file system; and in contrast to hardlinks, changes to a file are not reflected to the copy.

You create manually reflinks with:

```
$ cp --reflink name1 name2
```

4.4.4 References

See also:

- ln, readlink, cp.
- libc manual: Symbolic Links, libc manual: Hardlinks

4.5 capturing input

The tee command allow to duplicate stdout.

The script command can be used to capture the input and output to/from an application.

To replay a session you want to capture only the input. I achieve it by using:

```
$ cat /dev/stdin|tee /tmp/session_input| application
```

4.6 command return status

- You can test the status of a command by executing it in a conditional block:

```
   $ if echo "foo"; then echo "ok"; fi
   foo
   ok

If you want to keep this status you get it in the ``$?`` variable.
You can later on test it against the ``0`` value that mean *true* for
the shell.

..  code-block:: shell-session

    $ echo "foo"; if [ $? -eq 0 ]; then echo "ok"; fi
    foo
    ok
```

- but this convention is the opposite of C where 0 means false, so if you want in a shell supporting numerical expressions in `((...))` to use the numerical testing, which is C compatible you have to negate it.

```
$  echo "foo"; if ! (($?)); then echo "ok"; fi
foo
ok
```

- **avoid this**: `$?` is a number not a test so you cannot put it directly in an if expression.
- **avoid this**: `[ 1 ]` is **true** so a test like `[ $? ]` fail when `$?` is not defined, but succeed with both 0 and 1 values.
- an alternative is to use a number comparison:

```
if [ $? -eq 0 ]; then echo "ok"; fi
```

4.7 Command Expansion.

The Bash Reference gives a detailled description of each part of shell expansion, wich takes place during simple command expansion.

The order of expansion is very important; it is: brace expansion; tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution from left-to-right, word splitting, filename expansion, process substitution, quote removal.

To quote the manual: *Only brace expansion, word splitting, and filename expansion can change the number of words of the expansion; other expansions expand a single word to a single word.*

4.7.1 Parameter and Command Substitution

Parameter expansion commes before command substitution, it explains why we have:

```
$ a=foo; a=bar echo $a
foo
$ a=foo; a=bar /bin/echo $a
foo
$ a=foo; a=bar; echo $a
bar
$ a=foo; a=bar; /bin/echo $a
bar
```

In the second assignement the parameter substitution is done *before the assignment*.

This is true for a builtin like `echo` or a command like `/bin/echo`

You can find more on this subject in Bruce Barnett Grymoire - sh, a subtle point.

4.7.2 Filename expansion

After word splitting comes *filename expansion* where all the patterns found in the filename are expanded.

The following are done with default shopt settings which disable `dotglob` and `nullglob`.

```
$ mkdir dir
$ cd dir
$ touch b .b a aa aaa .a .aa ..a ba ab bab
$ echo a*
a  aa  aaa  ab
$ echo *a
a aa aaa ba
$ echo a?
aa ab
$ echo ?a
aa ba
$ echo ??
aa ab ba
$ echo *
a aa aaa ab b ba bab
$ echo .*
. .. .a ..a .aa .b
$ echo * .*
a aa aaa ab b ba bab . .. .a ..a .aa .b
$ echo .?*
.. .a ..a .aa .b
$ echo .??*
..a .aa
```

As we see there is no easy way to catch all files from a directory, the default does not match dotfiles, and if we add the initial dot we catch too less or too much. If we use bash you we can set an option `dotglob` so pattern can cath the dot, let try again:

```
$ shopt -s dotglob
$ echo *a
a  .a  ..a  aa  .aa  aaa  ba
$ echo a?
aa  ab
$ echo ?a
.a  aa  ba
$ echo ??
.a  aa  ab  .b  ba
```

(continues on next page)

(continued from previous page)

```
$ echo *
a .a ..a aa .aa aaa ab b .b ba bab
$ echo .*
. .. .a ..a .aa .b
$ echo .?*
.. .a ..a .aa .b
$ echo .??*
..a .aa
```

It is now easy to catch all files from the directory, but as we see for the last three commands none of these patterns expand to the *dotfiles*, the first catch also the directory itself and the parent directory, the next one catch also the parent directory, and the last one miss some filenames. Making an operation only on dotfiles is quite common so we must find a way to match all of them, and only them.

In bash we have an environment variable `GLOBIGNORE` which is not set by default, f `GLOBIGNORE` is set, each matching file name that also matches one of the patterns in `GLOBIGNORE` is removed from the list of matches. After setting globignore by `GLOBIGNORE=.:..` we get

```
$ echo .*
.a ..a .aa .b
```

which is the proper list of dotfiles.

A special feature is that setting `GLOBIGNORE` to a non null value always disable matching `.` and `..` so we have had the same result by setting `GLOBIGNORE=qwertyuop` !

Note that if we want to change `GLOBIGNORE` only for some expansions we ca do it in a subshell.

```
$ (GLOBIGNORE=.:..; echo .*)
.a ..a .aa .b
```

but as *explained above*, the following does not work :

```
$ GLOBIGNORE=.:.. echo .*
. .. .a ..a .aa .b
```

The previous use of `shopt` is only possible in bash, these options don't exist in plain bourne shell. But we can always match all dotfiles with

```
$ echo .[!.]* ..?*
.a .aa .b ..a
$ echo .[^.] .??*
.a .b ..a .aa
```

When a range match begins with `!` or a `^` any character not enclosed is matched, so with these two patterns we get rid of the unwanted `.` and `..`.

4.7.3 Parameter Expansion, Splitting and Quote Removal.

As parameter expansion is done before word splitting, and quote removal after word splitting, we have:

```
$ IFS=: set -- a:b:c; echo $1
a b c
$ string="a:b:c"
$ IFS=: set -- $string; echo $1
a
$ IFS=: set -- "$string"; echo $1
a b c
```

But when we have:

```
$ IFS=: read x y z <<< "a:b:c"; echo $x
a
$ string="a:b:c"
$ IFS=: read x y z <<< $string; echo $x
a
$ IFS=: read x y z <<< "$string"; echo $x
a
$ echo $string | { IFS=: read x y z; echo $x; }
a
```

In these four the input file descriptor is built before the read, and the three input are the same.

4.7.4 Quoting and splitting.

The bash reference describe Quoting but sometime the combination of quoting with Command Expansion can be difficult to sort out.

If you set a variable to a string, when use it as parameter in the process of Simple Command Expansion it is subject to Shell Expansion a complex process which involves Shell Parameter Expansion, Command Substitution, *Arithmetic Expansion*, Process Substitution, and then Word Splitting followed by Filename Expansion and Quote removal.

Let's apply with a simple command, made from a simple script named `countargs`:

```
#! /bin/sh
echo nbargs: $#
i=0
for a in "$@"; do
    i=$((i+1)) # ++i in bash!
    echo $i ':' "$a"
done
```

```
$ x="one two three"
$ ./countargs $x
nbargs: 3
...
$ ./countargs "$x"
nbargs: 1
1 : one two three
```

Very simple indeed, this is the expected behavior of quotting and word splitting.

But if you use your parameter in assignment the rules are different, **assignment are not commands** but a preliminary to Simple Command Expansion and quoting this section

There is **no word splitting** there, so the last two assignments are valids and equivalents:

```
$ x="one two three"
$ y=$x
$ z="$x"
$ echo $y
one two three
$ echo $z
one two three
```

Now if you use the special characters used for glob patterns like `?` and `*` in a word assigned to a variable, and evaluate the variable afterward the Filename Expansion can occur so you get:

```
$ x=sound*
$ echo $x
sound.wav
```

(continues on next page)

(continued from previous page)

```
$ echo "$x"
sound*
$ y=$x
$ echo "$y"
sound*
$ y=$(echo $x)
$ echo "$y"
sound.wav
$ y="$(echo $x)"
$ echo "$y"
sound.wav
```

Note that in the first assignement we have not used quotes, because as *set previously* in the text after an assignement the is no Filename expansion but of course there is quote removal, so the three following assignements are identical:

```
$ x='sound*'
$ x="sound*"
$ x='sound*'
```

In the same way the line breaks in a string assigned to a variable are used for word splitting so:

```
$ x="
> one
> two
> three"
5231$ echo $x
one two three
5232$ echo "$x"

one
two
three
$ y=$x
$ echo "$y"

one
two
three
```

As usual a trailing backslach remove the following newline:

```
$ x="one \
two \
three"
$ echo "$x"
one two three
```

The other backslash escape are uninterpreted, but you can force the interpretation with the command echo with the `-e` argument or the command printf. Both commands exists as Bash builtins. As *printf* is not a dash builtin, and the dash `echo` builtin does not support the `-e` argument; in portable shell you better rely on the command printf or the command echo.

```
$ x="one\ntwo\nthree"
$ echo $x
one\ntwo\nthree
$ echo -e $x
one
two
three
$ printf "%b\n" $x
```

(continues on next page)

(continued from previous page)

```
one
two
three
```

4.8 file descriptors

4.8.1 Reference

- Wikipedia: File descriptor
- Redirections are described in the Redirections section of the Bash Reference Manual.
- Advanced bash scripting guide has also a section on redirection that has more elaborated examples, than the following recipes.
- There are also many topics on redirection in the Bash FAQ : Redirect stderr to a pipe, Redirect the output of multiple commands at once, Send all output to a log file.

4.8.2 Opening - Closing - Listing

To assign fd 3 to myfile:

```
$ exec 3>myfile
```

To close fd 3:

```
$ exec >&3-
```

For input descriptors:

```
$ exec 3<myfile
$ exec <&3-
```

To open a fd for read-write:

```
$ exec 3<>myfile
```

To list open file descriptors:

```
$ ls -l /dev/fd/*
```

or:

```
$ lsof -a -p $$ -d 0-10
```

4.8.3 Copying - Moving

To copy a file descriptor you can use:

```
$ exec 3>&1
$ exec 1>|/tmp/output1
$ ls
$ exec 1>&3
$ exec 3>&-
```

In line *(1)* the file descriptor 1 is copied to fd 3, then *(2)* 1 is redirected to the file `/tmp/output1`, *(3)* the first ls goes in this file, then *(4)* fd 3 is copied back to fd 1 which comes back to it's previous value; *(5)* the descriptor 3 is then closed.

In Bash *but not dash* the last two lines can be abbreviated in:

```
$ exec 1>&-3
```

4.8.4 Swapping stdout and stderr.

```
$ f(){ echo out; echo error >&2; }
$ f
out
error
$ x=$(f)
error
$ echo $x
out
$ x=$(f 2>&1)
$ echo $x
out error
$ x=$(f 1>&2)
out
error
$ echo $x

$ x=$(f 3>&2 2>&1 1>&3)
out
$ echo $x
error
$ exec 3>&2; x=$(f 2>&1 1>&3); 3>&-
out
$ echo $x
error
$ exec 3>&1; x=$(f 2>&1 1>&3); 3>&-
out
$ echo $x
error
```

(2) Normal call to `f`, both error and output go to standard output.

(5) The output is assigned to `x` and the error go to stdout.

(9) The output fd replace the error, so both fd go to stdout and are assigned to `x`.

(12) The error fd replace the output, so both echos are going to error and nothing on stdout, `x` is empty.

(17) We execute f in a context where error is saved as fd 3, then output replacing error, previous saved error replace output.

(21) is similar to *(17)* except that we save stderr before evaluating the expression.

(25) Is harder to understand, and can seem paradoxal first, what use of saving fd (**f**ile **d**escriptor) 1 for copying it later to the same fd 1?

But the standard output outside and inside of the Command Substitution are not the same. Before the Command Substitution the current output is untouched, if we execute this script from a pseudo terminal it is /dev/pts/0 and this value is saved to fd 3.

Inside `$(...)` which is a Command Substitution construct, the standard output is redirected to a pipe to capture the output; we copy this pipe to fd 2 with `2>&1`, then the fd 3 containing the original output is copied back to the current output that is fd 1.

We execute the function in this context and affect the content of the pipe, that is what goes to stderr inside the function, to the variable; then fd 3 is closed.

4.8.5 Reading from a bloc of text.

The shell builtin **read** first aim is reading from standard input or any file descriptor. But we can use it with Here Documents, or with Here Strings *in bash or zsh but not dash, it is not a portable construct*, in a pipe, or with process substitution.

In standard shell like dash when we read from an Here Documents the builtin *read* read one line each time. In bash or zsh we can change the delimiter and read the whole *Here Document* or *Here String* in a single shell variable.

```
$ f=foo
$ read -d'' x <<EOF
$f and bar
        go in a boat
EOF
$ echo "$x"
foo and bar
        go in a boat
$ read -d'' x << "EOF"
$f and bar
        go in a boat
EOF
$ echo "$x"
$f and bar
    go in a boat
$ read -d'' x <<- EOF
$f and bar
        go in a boat
EOF
$ echo "$x"
foo and bar
go in a boat
```

As seen in the previous example *(9)* quoting the end string disable variable expansion, and *(16)* using <<- delete initial tabulations.

4.8.6 Extracting the parts of a string.

Very often we want to extract fiels from a strings separated with a single character. It may be a space or an other character, we often choose `,`, `:`, `;`, `!`, `|`, `%`, `/` or `\` but any char can be choosen as far it is not member of the strings; if we want to be free from this limitation we have to add an escaping mechanism which is not dealed here.

A related problem is when the fields are separated by any sace sequence constituted by space characters and tab characters, but it is easily converted to the previous case with one of:

```
string=$(echo $string0 | sed 's/[[:space:]]\{1,\}/ /g')
string=$(echo $string0 | sed -r 's/[[:space:]]{1,}/ /g')
```

using either old *basic* regex or standard posix 2 *extended regex*. But when you use the shell primitives you don't need this step as the shell with the default *IFS* use sequence of spaces to delimit words.

For the example we suppose the fields are delimited by `:`.

We can of course use basic coreutils

```
string="a:123:456"
v1=$(echo $string | cut -d: -f1)
v2=$(echo $string | cut -d: -f2)
v3=$(echo $string | cut -d: -f3)
```

But we can use only the shell even basic bourne shell or dash.

```
string="a:123:456"
IFS=":" set -- $string
v1=$1
v2=$2
v3=$3
```

Or using Here Documents that is found in any shell:

```
string="a:123:456"
IFS=: read v1 v2 v3 <<EOF
$string
EOF
```

We can also use in dash, busybox ash, yash, bash, zsh and others POSIX compatibles shells Shell Parameter expansion

```
string="a:123:456"
var="${string}::"
i=0
while [ "$var" != ':' ]; do
  i=$((i+1))
  # drop part of string from first ':' to the end
  iter=${var%%:*}
  echo "v_$i=$iter"
  # drop begin of string upto first ':'
  var="${var#*:}"
done
```

Here we just echo the variables name we don't set them. To set the variables `v_1`, `v_2`, `v_3`, in bash or zsh we can replace the echo by:

```
declare v_$i="$iter"
```

or use the POSIX directive *typeset* that is synonim to *declare* in *bash* and *zsh*; but is also available in *yash*:

```
typeset v_$i="$iter"
```

in any bourne shell, *dash* or *ash* we fallback to an eval.

```
eval v_$i="$iter"
```

But quite harmless as we know the range of values of `$i`.

In bash, yash or zsh we can also use *Here String*

```
string="a:123:456"
IFS=: read v1 v2 v3 <<< "$string"
```

Also bash yash and zsh can use arrays to read the values; it is peculiarly useful when you don't know how many fields can be present.

In bash you write:

```
$ string="a:123:456"
$ IFS=: read -a v <<< "$string"
$ echo "${v[@]}"
a 123 456
$ declare -p v
declare -a v=([0]="a" [1]="123" [2]="456")
```

You can also use the previous code in yash or zsh; with a slightly different syntax:

```
$ string="a:123:456"
$ IFS=: read -A v <<< "$string"
$ echo "${v[@]}"
a 123 456
$ typeset -p v
v=('a' '123' '456')
```

An other way in bash or zsh is to assign directly the array.

```
$ string="a:123:456"
$ IFS=: v=(${string})
$ declare -p v
declare -a v=([0]="a" [1]="123" [2]="456")
```

As an example of use we parse the ouput of the `route` command to find the different fields of the default route.

```
read dest gateway mask flags metric ref use iface <<EOF
$(route -n | grep '^0\.0\.0\.0' )
EOF
```

But when we use the previous methods to parse the output of a command; the command should be executed *before* the parsing; in some case e might want to prefer, or be obliged, to process asynchronously the results. Unix use pipes for this, and everything works as long as you want to use the result of the previous process in the subsequent one. But it can be more difficult to get the result of a subprocess inthe parent process. This is the subject of next paragraph.

4.8.7 Using an asynchronous subprocess.

A pipe open a subshell. As variables in a subshell are inaccessible from the parent shell, all the variables set inside a pipe are also unavailable out of the pipe.

We often meet this problem while trying to read from the output of a pipe:

```
$ x="unset"
$ y="unset"
$ echo one two | { read x y; echo $x $y; }
one two
$ echo $x $y
unset unset
```

We can use redirection to avoid a subshell, we can either use a temporary file, a fifo or process substitution.

The use of a temporary file is allowed in bare bourne shell, dash, ash, yash and usefull for portable script.

```
$ echo "a b">|/tmp/tmpfile
$ exec 4< /tmp/tmpfile
$ read x y <&4
$ echo $x $y
a b
$ exec 4<&-
$ rm /tmp/tmpfile
```

If we need to split our string on another character than a sequence of spaces, tabs and newlines, which constitute the default IFS, we only change the IFS before the script and reset it or change it into a subprocess.

But here changing IFS for only the read works as well.

```
$ echo "a:b">|/tmp/tmpfile
$ exec 4< /tmp/tmpfile
$ IFS=":" read x y <&4
$ echo $x $y
```

(continues on next page)

(continued from previous page)

```
a b
$ exec 4<&-
$ rm /tmp/tmpfile
```

If our shell and system admit process substitution, which is the case of bash, and zsh on systems that support named pipes (FIFOs) or the `/dev/fd` files:

```
$ exec 4< <(echo a b)
$ read x y <&4
$ echo $x $y
a b
$ exec 4<&-
```

Or simply, a more condensed form:

```
$ read x y < <(echo a b)
$ echo $x $y
a b
```

This can even be used in yash with process redirection whose syntax is different from process substitution which is found in bash or zsh.

```
$ read x y <(echo a b)
$ echo $x $y
a b
```

As these custom syntaxes are not in Posix; and for portability it is better to avoid them.

A pipe create an implicit fifo, which imply to use a subshell, but we can also avoid it and keep the benefit of forking a producer by using an explicit fifo. This is also available in any shell.

```
$ mkfifo /tmp/fifo
$ echo a b >/tmp/fifo &
[1] 6934
$ read x y </tmp/fifo
$ echo $x $y
a b
[1]+  Done    echo a b > /tmp/fifo
$ rm /tmp/fifo
```

We illustrate the use of process substitution to parse the ouput of the `route` command to find the different fields of the default route; that we *did previously <parsing-route-1>* sequentially.

```
read dest gateway mask flags metric ref use iface < \
<(route -n | grep '^0\.0\.0\.0' )
```

If we don't need the asynchronous processing of the previous scripts we can store the output of the first command in a string and read from that string this is illustrated by the paragraph on splitting output of a command.

4.9 Elapsed time of a command

To get the time of a command we can use the time command

```
$ /usr/bin/time -f "%e elapsed, %U user, %S sys" locate xzuv
Command exited with non-zero status 1
1.72 elapsed, 1.61 user, 0.04 sys
```

We can also under bash use the internal time bash command, this one can be used not only with a command but before any pipe, command group, or subshell

```
$ time locate xzuv
$ time (ls >/dev/null; cat /etc/passwd >/dev/null)
$ time { ls >/dev/null; cat /etc/passwd >/dev/null; }
real    0m0.021s
user    0m0.000s
sys     0m0.012s
```

To know the elaped time of some part of a script we can also use the date command:

```
before="$(date +%s)"
..... #some shell commands
after="$(date +%s)"
echo "elapsed: $(date -u -d @$(($after - $before)) +%H:%M:%S)"
```

4.10 array indexes and globbing.

Bash can store arrays in a shell variable, but interaction between array indexes and pathname expansion is a tricky and badly documented aspect of bash.

This is summarized by the next small script

```
$ echo t[1]
t[1]
$ shopt -s nullglob
$ echo t[1]

$ touch t1
$ echo t[1]
t1
```

The same pathname expansion is done after an `unset` command, so doing `unset t[1]` may result in unsetting the array element `t[1]` or the unsetting the variable `t1` or causing an error or doing nothing, depending on the presence of a file named `t1` and of the setting of the options `nullglob`, `failglob`, `extglob`, and the environment variable `GLOBIGNORE`

So you are advised always quoting the argument of an `unset` and write: `unset 't[1]'`.

Note that within an expression like `${t[1]}` *braces disable pathname expansion*

4.11 bash and zsh regex expressions.

In bash, since version 3.0, you can match gnu regex, it allows to dispense with the call to `expr` or `sed` (the price is a lesser compatibility with older release of batch or other bourne shells, it is definitely not posix).

You use it like this:

```
$ [[ "abbbaaaaabbb" =~ a*(b*)(a*)(ab*) ]]
$ echo "${BASH_REMATCH[@]}"
abbbaaaaabbb bbb aaaa abbb
```

The array variable `BASH_REMATCH` contains substrings matched by parenthesized subexpressions.

In zsh if the option `BASH_REMATCH` is set the result is identical to the bash one. Otherwise we get the global match with `$MATCH`; and the groups with `${match[@]}`

```
$ [[ "abbbaaaaabbb" =~ a*(b*)(a*)(ab*) ]]
$ echo "$MATCH"
abbbaaaaabbb
```

(continues on next page)

(continued from previous page)

```
$necho "${match[@]}"
bbb aaaa abbb
```

The match in bash and the default on zsh is using POSIX regex functions, the same we use in *grep*.

Zsh can also test the regexp as a PCRE regular expression by setting the option `RE_MATCH_PCRE`.

4.12 Using `getopts`.

`getopts` is a POSIX function; defined in bash and zsh.

In bash the details are reviewed in abs: example 11-8, and you get a summary by typing `help getops` under the shell.

In zsh `$ getopts` followed by `Esc-h` give the interactive help.

When using it in a function it looks like that:

```
local opt OPTARG
local -i OPTIND=1
while getopts :d:D:p:F opt; do
    case $opt in
        d|D) myoptarg1=$OPTARG ;;
        p) myoptarg2=$OPTARG ;;
        F) myopt3=true ;;
        *) help $FUNCNAME
        exit 2
    esac
done
shift $(( OPTIND - 1 ))
```

If not in function replace `local` by `declare`

4.13 using `getopt`.

An example is given with getopt(1) in `/usr/share/doc/util-linux/examples/`, it is recalled here:

```
# We need TEMP as the `eval set --' would nuke the return value of getopt.
TEMP=$(getopt -o ab:c:: --long a-long,b-long:,c-long:: \
     -n 'example.bash' -- "$@")
if [ $? != 0 ] ; then echo "Terminating..." >&2 ; exit 1 ; fi
# Note the quotes around `$TEMP': they are essential!
eval set -- "$TEMP"
while true ; do
   case "$1" in
       -a|--a-long) echo "Option a" ; shift ;;
       -b|--b-long) echo "Option b, argument \`$2'" ; shift 2 ;;
       -c|--c-long)
            # c has an optional argument. As we are in quoted mode,
            # an empty parameter will be generated if its optional
            # argument is not found.
            case "$2" in
                "") echo "Option c, no argument"; shift 2 ;;
                 *)  echo "Option c, argument \`$2'" ; shift 2 ;;
            esac ;;
       --) shift ; break ;;
       *) echo "Internal error!" ; exit 1 ;;
   esac
```

(continues on next page)

(continued from previous page)

```
done
echo "Remaining arguments:"
for arg do echo '--> '"\`$arg'" ; done
```

4.13.1 `getopt` and whitespaces.

he old version of `getopt` does preserve whitespaces in arguments so you

get:

```
$ getopt a: -- -a "one two" "three four"
 -- -a one two three four
```

This stand either with the old `getopt` or the enhanced one, as the latter generate output that is compatible with that of other versions, as long as his first parameter is not an option.

This defect is the cause of `getopt` rejection in some manuals as in the SHELLdorado good coding practices

But if you use the enhanced version with it's new syntax you get:

```
$ getopt --options a: -- -a "one two" "three four"
 -a 'one two' -- 'three four'
```

So the whitespaces are preserved, with the new `getopt`;. You can check that yourgetopt; is the enhanced one by doing `getopt -V` or in a script:

```
$ getopt -T
$ if [ $? -eq 4 ]; then
# code for new getopt
```

The return value of 4 is the sign of the enhanced version.

CHAPTER

FIVE

PORTABLE SHELL CONSTRUCTS

5.1 Utilities used in a shell script.

The Gnu coding standard - Utilities in Makefiles list them:

> The configure script and the Makefile rules for building and installation should not use any utilities directly except these: awk cat cmp cp diff echo egrep expr false grep install-info ln ls mkdir mv printf pwd rm rmdir sed sleep sort tar test touch tr true

5.2 Converting bash syntax to dash

This table is extracted from Bashism look at the original for more info.

construct	bash	dash
functions	`function f { echo hello world; }`	`f() { echo hello world; }`
cases	`;;& ;& etc`	None. Duplicate the case (use a function to avoid code duplication)
C-like for loop	`for ((i=0; i<3; i++)); do echo "$i" done`	`i=0 ; while [ "$i" -lt 3 ]; do echo "$i" ; i=$(($i+1)) done`
expand sequences	`echo $'hello\tworld'`	`printf "hello\tworld\n"`
extended glob	`+( ) @( ) !( ) *( )`	sometimes you can use several globs, sometimes you can use find(1)
select	`select`	implement the menu yourself, use a command like dialog
Brace Expansion	`{a,b,c}` or `{1..10}`	
process substitutions	`foo <(bar)`	`mkfifo fifo; bar > fifo & foo < fifo`
	`${name:n:l}`	`$(expr "x$name" : "x.\{,$n\}\(.\{,$l\}\)")`
	`${name/foo/bar}`	`$(printf '%s\n' "$name" | sed 's/foo/bar/')`
	`${!name}`	use eval (dangerous)
arrays		positional parameters, use `IFS` and `set -f`, `eval` (dangerous)
simple test	`[[`	use `[` and use double quotes around the expansions `[ "$var" = "" ]`
pattern matching	`[[ foo = *glov ]]`	use case or expr or grep
equality with test	`==`	use = instead

Continued on next page

Table 1 – continued from previous page

construct	bash	dash
compare lexicographically.	<>	no change
modification times	[[file1 -nt file2]] or -ot	["$(find 'file1' -prune -newer 'file2')"] or ["file1" -nt "file2"]
files are the same hardlink	[[file1 -ef file2]]	["file1" -ef "file2"]
(( ))	var=((3+1))	var=$((3+1))
(( ))	((3 + 1 < 5))	[$((3 + 1)) -lt 5] or ["$((3 + 1 < 5))" -ne 0]
pre/post increment/decrement	++ --	i=$((i+1)) or : $((i+=1))
comma operator	, (ok in ash)	: "$((...))"; cmd "$((...))"
exponentiation	** (ok in ash)	•
let	let x=15; let x--	x=15; x=$((x - 1))
redirect both stdout and stderr	>& and &>	command > file 2>&1 or command 2>&1 | command2
duplicate and close	m>&n- m<&n-	m>&n n>&-
herestring	<<<"string"	echo "string"| command
source	source lib/xxx.sh	. lib/xxx.sh
expand ~ in PATH	PATH="~/bin:$PATH"	‘‘ PATH=”$HOME/bin:$PATH”‘‘

CHAPTER

SIX

SYSTEMD

6.1 systemctl

Reference: systemctl

$ systemctl status	Show system status
$ systemctl list-units	List running units
$ systemctl	List running units
$ systemctl –failed	List failed units
$ systemctl –all	List running and inactive units
$ systemctl list-unit-files	State of all installed units
$ systemctl status unit	unit status
# systemctl start unit	start
# systemctl stop unit	stop
# systemctl restart unit	restart the unit
# systemctl reload unit	reload configuration
$ systemctl is-enabled unit	enabled or disabled?
# systemctl enable unit	enable to be started on boot
# systemctl disable unit	disable not to be started on boot
# systemctl mask unit	forbid starting the unit
# systemctl unmask unit	unmask
# systemctl help unit	help for the unit
# systemctl daemon-reload	reload systemd
$ systemctl reboot	reboot system
$ systemctl poweroff	power-off
$ systemctl suspend	suspend to memory
$ systemctl hibernate	hibernate on disk
$ systemctl hybrid-sleep	hibernate then sleep

6.2 journalctl

Reference: journalctl

$ journalctl	system log
$ journalctl -b	journal since boot
$ journalctl -b 1	journal of previous boot
$ journalctl –since “2016-04-07 12:00:00”	journal since some date
$ journalctl –since “2h ago”	journal since some time
$ journalctl -u httpd –since=00:00 –until=9:30	
$ journalctl -u ntp	journal for unit ntp
$ journalctl -f	follow new messages
$ journalctl -e	jump at end of journal
$ journalctl -n 1000	show at most 1000 entries
# usermod -a -G adm lennart	allow user lennart to see logs

CHAPTER

SEVEN

DEBIAN PACKAGE CONFIG MEMO

7.1 dpkg Memo

See also dpkg(1), dpkg-deb(1), dpkg.cfg(5), dlocate(1), apt-file(1)

- Find out all the options:

```
dpkg --help
```

- Print out the control file (and other information) for a specified package:

```
dpkg --info foo_VVV-RRR.deb
```

- Install a package (including unpacking and configuring) onto the file system of the hard disk:

```
dpkg --install foo_VVV-RRR.deb
```

- status of all the packages installed on a system:

```
dpkg --list
```

- status of packages matching foo* installed on a system:

```
dpkg --list 'foo*'
```

- Detailed status of foo, including dependencies and configuration:

```
dpkg --status foo
```

  Note that configuration files are followed by the md5sum of the original configuration file provided by the package. It allows the package manager to know when they are changed. You can also use it in the same way.

```
$ dpkg --status mysql-common
.....
Conffiles:
/etc/mysql/conf.d/.keepme d41d8cd98f00b204e9800998ecf8427e
/etc/mysql/my.cnf 77f15d6c87f9c136c4efcda072017f71
$ md5sum /etc/mysql/my.cnf # unchanged conf
77f15d6c87f9c136c4efcda072017f71  /etc/mysql/my.cnf
```

- files provided by the installed package foo:

```
dpkg --listfiles foo
```

- dlocate cache the packages content and allow a quicker processing:
  - list status with:

```
dlocate -l 'foo*'
```

  - list files with:

```
dlocate -L foo
```

  - files long list with:

```
dlocate -ls foo
```

- what installed package produced a particular file:

```
dpkg --search filename
```

  or:

```
dlocate -S  filename
```

- what package (installed or not) produced a particular file:

```
apt-file search foo
```

- Determine what files are contained in a Debian archive file:

```
dpkg-deb --contents foo_VVV-RRR.deb
```

- Extract the files contained in a named Debian archive into a directory without installing the archive:

```
dpkg-deb --extract foo_VVV-RRR.deb tmp
```

- Unpack (but do not configure) a Debian archive, This command removes any already-installed version of the program and runs the preinst but does not necessarily leave the package in a usable state; it has to be configured:

```
dpkg --unpack foo_VVV-RRR.deb
```

- Configure a package that already has been unpacked, this action runs the postinst script and updates the files listed in the conffiles for this package:

```
dpkg --configure foo
```

- reconfigure a package:

```
dpkg-reconfigure foo
```

- Extract all files matching glob pattern “blurf*” from a Debian archive:

```
dpkg --fsys-tarfile foo_VVV-RRR.deb | tar -xf - blurf*
```

- Remove a package (but not its configuration files):

```
dpkg --remove foo
```

  or:

```
aptitude remove foo
```

- Remove a package (including its configuration files):

```
dpkg --purge foo
```

  or:

```
aptitude purge foo
```

- List the installation status of packages containing the string (or regular expression) `'foo*'`:

```
dpkg --list 'foo*'
```

- Configuration files policy, without prompt: They are listed in the `--force-things` section of the dpkg(1) manpage. Except with *confask* they apply only when the version of the package change, not when reinstalling the same version.
  - List the *force* options:

    ```
    dpkg --force-help
    ```

  - Do not modify the current configuration file touched or not:

    ```
    dpkg --install --force-confold foo
    ```

  - Do not modify the current configuration file when touched, but apply the default policy when untouched (usually update it!):

    ```
    dpkg --install --force-confold --force-confdef foo
    ```

  - Install the new version of a modified configuration file, the current version is kept in a file with the .dpkg-old:

    ```
    dpkg --install --force-confnew foo
    ```

  - If a conffile is missing and the version in the package did change, always install the missing conffile without prompting:

    ```
    dpkg --install --force-confmiss foo
    ```

  - If a conffile has been modified always offer to replace it, even if the version in the package did not change:

    ```
    dpkg --install --force-confask foo
    ```

  - Use *confask* to force install the new version of the conffile when reinstalling the same version of the package:

    ```
    dpkg --install --force-confask --force-confnew foo
    ```

  You can also use these options from *apt-get* or *aptitude as explained below*.

7.2 apt/aptitude memo

7.2.1 References

- apt(8), apt-get(8), apt.conf(5), sources.list(5).
- apt-cache(8), apt-file(1)
- apt-offline(8)
- Debian Reference: Debian package management
- aptitude User Manual, command line use and aptitude Command-Line Reference.
- Aptitude reference guide: search patterns.

- The commands that install, upgrade, and remove packages all accept the parameter `-s`, which stands for "simulate". When `-s` is passed on the command line, the program performs all the actions it would normally perform, but does not actually download or install/remove any files.

7.2.2 Install/Remove

- update the list of available packages at the repositories:

```
aptitude update
```

or:

```
apt-get update
```

- upgrade each package on the system, after installing versions of packages upon which it depends:

```
aptitude update
aptitude safe-upgrade
aptitude full-upgrade
```

- in the *safe* version installed packages are not removed unless they are unused.
- with apt: `apt-get upgrade` or `apt-get dist-upgrade` use the *safe* command.
- installs package from the unstable distribution while installing its dependencies from the current distribution:

```
aptitude install package/unstable
```

- installs package from the unstable distribution while installing its dependencies also from the unstable distribution by setting the Pin-Priority of unstable to 990:

```
aptitude install -t unstable package
```

- checks the status of packages foo bar

```
aptitude show foo bar ... | less
```

or:

```
apt-cache show foo bar ... | less
```

- installs the particular version 2.2.4-1 of the foo package:

```
aptitude install foo=2.2.4-1
```

- installs the foo package and removes the bar package:

```
aptitude install foo bar-
```

- removes the bar package but not its configuration files:

```
aptitude remove bar
```

- removes the bar package together with all its configuration files:

```
aptitude purge bar
```

- Use `--force-things` for controlling conffiles replacement when calling dpkg from apt and aptitude:

```
apt-get install --reinstall -o Dpkg::Options::="--force-confmiss" foo
aptitude reinstall -o Dpkg::Options::="--force-confmiss" foo
```

See the force options above in the dpkg Memo.

7.2.3 informations about packages

- update cache and check for broken packages

```
apt-get   check
```

- Text search in the package names and the descriptions for the POSIX regex pattern:

```
apt-cache search  pattern
```

- When using aptitude, the patterns are composed by terms introduced by the character "?" or "~", the default search is `?name()` or `~n`.To look for packages containing `foo` in their name use one of

```
aptitude search foo
aptitude search '?name(foo)'
aptitude search ~nfoo
```

- Search all manually installed packages (~i: installed, !~M not automatic)

```
aptitude search '~i!~M'
```

- Search all packages with tag hardware::input:keyboard

```
aptitude search ~Ghardware::input:keyboard
```

- Search all packages whose description contains the word "switcher"

```
aptitude search ~dswitcher
```

- Search all installed packages that contains "firewall" in description.

```
aptitude search '~dfirewall~i'
```

- Search all package installed from an other archive than debian

```
aptitude search '!~Odebian'~i
```

- Search or show all packages of priority *standard* (priority must be extra, important, optional, required, or standard.)

```
aptitude search '?priority(standard)'
aptitude search '~p standard'
aptitude show '?priority(standard)'
aptitude show '~p standard'
```

- Search patterns description is in Debian Reference: The aptitude regex formula and Aptitude reference guide: search patterns. Aptitude reference guide:Search term reference

  This table is a shorter reference only to the short form of search term .The Search term reference has also the longer form.

key	val	key	val	key	val
~A<archive>	archive	~G<tag>	tag	~s	section
~a<action>	action	~i	installed	~T	true
~B<type>	Broken-<type>	~M	automatic	~t<task>	task
~b	broken	~m<name>	maintainer	~U	upgradable
~C<pattern>	conflict	~N	new	~V<version>	version
~c	config-files	~n<name>	name	~v	virtual
~D	dependency	~O<origin>	origin	~w<pattern>	widen
~d	description	~P<pattern>	provides	!<pattern>	not
~e	essential	~p<priority>	priority	<patt1> <patt2>	and
~F	false	~R<type>:<patt>	reverse-<type>	<patt1>|<patt2>	or
~g	garbage	~S <filter> <patt>	narrow		

<type> is one of `depends`, `predepends`, `recommends`, `suggests`, `breaks`, `conflicts`, or `replaces`.

garbage means not required by any manually installed package.

- package priority/dists information:

```
apt-cache policy  package
aptitude versions package
```

- show description of candidate version of a package:

```
apt-cache show --no-all-versions package
aptitude show package
```

- show description of package in archive:

```
aptitude show package/archive
```

or:

```
aptitude show -t archive package
```

- show the installed version of a package:

```
apt-show-versions -p package
apt-show-versions -r regex
```

- show all versions in archives:

```
apt-show-versions -a package
```

- show description of all versions of a package:

```
aptitude -v show package
```

- show description of package in all dists:

```
apt-cache show package
apt-cache show -a package
```

- show description of matching source package:

```
apt-cache showsrc package
```

- package information including what repositories provide available versions and forward and reverse dependencies

```
apt-cache showpkg package
```

- Print the full package record of a package including all aptitude show output and md5, sha1, sha256 sums, and tags:

```
apt-cache show package
dpkg --print-avail package
```

- Transitive dependencies and reverse dependencies of a package:

```
apt-cache depends package
apt-cache rdepends package
```

- You can also use aptitude by replacing

```
apt-cache rdepends xdg-utils
```

by:

```
aptitude search '?dependency(xdg-utils)'
```

but the to search all dependencies of the package like `apt-cache depends` you need a complex search

```
aptitude search
'?reverse-depends(xdg-utils)\|?reverse-recommends(xdg-utils)\|reverse-
↪suggest(xdg-utils)'
```

- Detailed information about the priority selection of the named package. It helps to debug your preferences pinning.

```
apt-cache policy <package>
```

- Look for a file matching a pattern among the sources.list packages, first update the `apt-file` cache with:

```
apt-file update
```

Then search with:

```
apt-file search <pattern>
```

We can switch from the default glob pattern to a regex or a fixed string with:

```
apt-file --regexp search <pattern>
apt-file --fixed-string search <pattern>
```

- Look for a file matching a pattern among installed packages *only'*:

```
dpkg --search <pattern>
dlocate -S <string>
```

- Content of all packages (among the sources.list packages) whose name match a pattern:

```
apt-file list <pattern>
```

for installed packages *only* use:

```
dpkg {-listfiles|-L} <pattern>
dlocate -l <pattern>
```

for deb package files:

```
dpkg -c </path/to/pkg.deb>
```

- Dependencies and reverse dependencies of a package:

```
apt-cache depends pkg(s)
apt-cache rdepends pkg(s)
```

- how many packages you have from testing:

```
apt-show-versions | fgrep /testing | wc
```

- list of upgradeable packages *including upgrades not in preferences*:

```
apt-show-versions -u
```

- upgrade all unstable packages to their newest versions *(dangerous)*:

```
aptitude install `apt-show-versions -u -b | fgrep /unstable`
```

7.2.4 importing a key

Reference: apt-key(8)

- With apt-key the command *adv* allow to use gpg to receive a key, you will use either the default keyserver or give one explicitly:

```
sudo apt-key adv --recv-keys --keyserver hkp://pgp.mit.edu <missing key>
```

- You can also directly provide the key in stdin:

```
wget -q http://fr.packages.medibuntu.org/medibuntu-key.gpg -O- | \
sudo apt-key add -
```

- or put it in your keyring:

```
gpg --keyserver hkp://subkeys.pgp.net --recv-keys KEY_ID
gpg -a -export KEY_ID | sudo -H apt-key add -
```

CHAPTER

EIGHT

ACCESS CONTROL LIST

8.1 ACL References

- Acl is described in acl (5), its use is in setfacl (1) and getfacl (1).
- ArchWiki: Access Control Lists.
- POSIX Access Control Lists on Linux by Andreas Grünbacher discuss ACL and extended attributes on Linux file systems.
- Debian Wiki: Access Control Lists in Linux
- Using ACLs with Fedora Core 2 by Van Emery is an old How-To *But neither outdated nor Fedora specific*.
- eiciel is the GNOME file ACL editor.
- ACLbit is an ACL Backup and Inspect Tool
- python-pylibacl provide an acl interface to python
- NFSV4 has an ACL support The nfsv4 acls are thinner than the posix acls, even if they get translated to posix. you manage nfsv4 acl with *nfs4-acl-tools nfs4_getfacl* and *nfs4_setacl*.

8.2 Access acl

Access acl use *owner*, *group*, *other* inherited from file permission bits and a list of named user and group access. The group owner, named used and group access form the *group class*.

In a system call when the new object is created in a directory the access is further modified by the *mode* parameter (9 bits fields), in case of default acl the 'umask' is not used.

To access an object we select an acl entry by following the order: *owner*, named users, (all owning or named) groups, *others*. If the permission bit is not set in any of these acl, the access is denied.

If the permission bit is set, access is granted for an *owner* or *other* permission. For a named user, owning group, or named group entry the permission is granted if the corresponding bit of the group is 1.

When a new directory is created his mask permission is the union of all permissions in the group class. The group class contains the owning group, other, and all named user and named group permission. So the initial mask of a directory without acl is the union of group and other.

In a directory with a mask chmod change the mask (which limit the *group class* permission), nor the *owning group* permission.

8.3 Default acl.

A new directory inherit from the default acl of his parent directory both as *access acl* and *default acl*. When a directory has extra acls it is listed by `ls -l` with an extra + after the access bits.

You can reset the acls to the default value by one of:

```
$ setfacl -b /path/to/directory
$ setfacl --remove-all /path/to/directory
```

8.4 Using acl to control access.

You may have some directory that contains sensible data and you don't want to give others a read permission in any (or most) part of this directory. It is of course easy to change the permission on all objects of a directory, but it may be more complicated to ensure, that all files that you will create in the future will have the same access rights.

The mask for new files is controlled by the 'umask'. Suppose your umask is 022, you may want to have a directory with 077 mask but once *umask* set in your environment by your .profile , it is difficult to ensure that every process creating a file in this directory will get not the default *umask* but a stricter one.

But acl can help to solve this problem, if you set the default acl for your protected directory to some sensible value every process accessing this directory, will not use the mask but the *mode* field, and files or directory are created in accordance to your default acl.

8.5 My acl to protect crypt and other sensitive data.

```
setfacl -R -d user::rwx,group::---,other::---,user:root:rwx directory

setfacl -R -d user::rwx,group::---,other::---,user:root:rwx directory
```

CHAPTER

NINE

FILE ATTRIBUTES

9.1 References

- Extended attributes: the good, the not so good, the bad.
- chattr
- man pages: lsattr(1), chattr(1)

9.2 attributes memo

Only the superuser or a process possessing the `CAP_LINUX_IMMUTABLE` capability can set or clear attribute.

- file ‘a’ attribute: can only be open in append mode for writing.
- directory ‘D’ attribute write synchronously on the disk; this is equivalent to the ‘dirsync’ mount option, but only applied to the directory.
- file ‘d’ attribute means no backup with *dump*.
- directory ‘I’ unchangeable) used by the htree code to indicate that a directory is being indexed using hashed trees.
- file ‘i’ ‘’immutable” attribute cannot be modified: it cannot be deleted or renamed, no link can be created to this file and no data can be written to the file.
- file ‘j’ attribute all data is written to the ext3 journal before being written to the file itself, if the filesystem is mounted with the “data=ordered” or “data=writeback” options. No effect if mounted with the “data=journal”
- file ‘S’ attribute equivalent to the ‘sync’ mount option applied to a subset of the files.
- directory ’T’ attribute used by the Orlov block allocator to indicate the top of directory hierarchies
- file ‘u’ attribute ask for undeletion support.
- ‘c’ ‘s’ ‘u’ are auto compression flags to attribute compress the file on disk and uncompress at read, not yet implemented on ext2/ext3
- ‘E’ ’X’ ‘Z’ attribute are used by the experimental compression patches

Attribute are listed py lsattr(1) and changed by chattr(1) - ArchWiki: :archwiki:‘File attributes

<File_permissions_and_attributes#File_attributes>‘.

9.3 Extended file attributes

The ext2, ext3, ext4, JFS, Squashfs, ReiserFS, XFS, Btrfs and OCFS2 1.6 filesystems support extended attributes (abbreviated xattr) when enabled in the kernel configuration. Any regular file or directory may have extended attributes consisting of a name and associated data.

getfattr(1) and setfattr(1) utilities retrieve and set xattrs. - ArchWiki: :archwiki:‘Extended attributes

<File_permissions_and_attributes#Extended_attributes>‘.

CHAPTER

TEN

MEMORY AND SWAP MANAGEMENT

10.1 Inspecting and tuning ram

10.1.1 Free memory

```
$ free -m
```

output:

```
             total       used       free     shared    buffers     cached
Mem:          2003       1618        385          0        136        592
-/+ buffers/cache:        889       1113
Swap:         2047          0       2047
```

It means you have 2G ram 1.6G are used 889M actively by applications and 728M can be reclaimed 136M from buffers and 592M from cached data; the total amount of space that could be used is 1.1G

2G of swap are unused.

You can find more explanations in Linux ate my ram and in Tips for Optimizing Linux Memory Usage.

A very good ref is also Tuning the Memory Management Subsystem. chapter 15 of openSUSE System Analysis and Tuning Guide.

10.2 Managing swap space

10.2.1 Swap Info

```
$ swapon -s
```

The output is:

```
Filename                        Type            Size    Used    Priority
/dev/mapper/vg-swap             partition               2097148 0       -1
```

or *free -m*

10.2.2 Swap partition

```
$ mkswap /dev/mapper/vg-swap
$ swapon /dev/mapper/vg-swap
```

In *fstab*:

```
/dev/mapper/vg-swap none  swap  sw 0 0
```

10.2.3 Swap file

```
$ dd if=/dev/zero of=/swapfile bs=1M count=512
```

if the file system is ext4 or brtfs *but not ext2/3* you can also use the quicker:

```
$ fallocate -l 512M /swapfile
```

Then:

```
$ chmod 600 /swapfile
$ mkswap /swapfile
$ swapon /swapfile

/swapfile none swap defaults 0 0
```

10.2.4 Swappiness

A low value of kernel swappiness parameter will reduce swapping from RAM, default is 60 and current value is:

```
$ cat /proc/sys/vm/swappiness
```

To test swapiness:

```
$ echo 5 > /proc/sys/vm/swappiness
```

To set it at boot put in */etc/sysctl.conf*:

```
vm.swappiness = 5
```

CHAPTER

ELEVEN

UDEV DEVICE MANAGEMENT AND USB DEVICES

11.1 References

- udev(7) explains the key and variables you can use in rules.
- lsusb(8).
- udevadm(8).
- ArchWiki: udev.
- Writing udev rules by Daniel Drake *2008 - the udev command names are obsolete, but the guide is still useful*,
- Gentoo Wiki - Udev
- To detect hardware including bus information use lspci(8), lshw(1) and the *Suse* tool hwinfo (also in debian).

11.2 localrules

Some local udev rules must be placed before defaults because the first matched rule is applied, Some other rules can be placed *after* the main rules and will only modify a yet created device as when you change permissions, add a symlink or a `RUN` action.

We also need to check that each rule matches on all keys, and the rule match exactly one device.

You can put your rules in `/etc/udev/rules.d/`, and as the rules are used in lexicographical order you give a proper priority such they are used before or after a corresponding rule in `/lib/udev/rules.d/`.

If you want to *replace* a provided rule in `/lib/udev/rules.d/`, you can just write a rule with the same name in `/etc/udev/rules.d/` as they take precedence over the `/lib` rules.

11.3 finding usb devices

- To have a list of usb devices just do:

```
$ lsusb
```

- A slightly more informative list is got by:

```
$ lsusb -v \| grep -E '<(Bus\|iProduct\|bDeviceClass\|bDeviceProtocol)' 2>/dev/
↪null
```

- To get detailled information you can
  - Use the *bus:devnum* reported by lsusb as in:

```
lsusb -v -s 003:001
```

– Use the *vendor:product* that you find with `udevadm info` like:

```
$ lsusb -v -d 04b8:082b
```

Many of the keys in `lsusb` are attributes of `udevadm info` and can be used in udev configuration.

11.4 Using /sys keys

We obtain the keys by looking for dev `/sys` info, with `udevadm info`:

1. find the key by either
   - Report with `udevadm info` providing the device name

     ```
     $ udevadm info --query=property --name /dev/usb/lp0
     ```

   - use the path that you also get from device by

     ```
     $ udevadm info --query=path --name /dev/usb/lp0
     /class/usb/lp0
     ```

     and get dev `/sys` info by

     ```
     $ udevadm info --query=property --path /class/usb/lp0
     ```

2. We can then find appropriate keys to identify uniquely our device:

```
SYSFS{manufacturer}=="Hewlett-Packard"
SYSFS{product}=="DeskJet 840C"
```

3. We then add our rule in /etc/udev/rules.d/10-local.rules

```
BUS=="usb", SYSFS{product}=="DeskJet 840C", NAME="%k", KERNEL=="lp[0-9]*", NAME=
↪"usb/%k", GROUP="lp", SYMLINK="deskjet"
```

4. Reload udev conf by:

   ```
   $ udevadm control --reload
   ```

5. Test the config with:

   ```
   $ udevadm test $(udevadm info -q path -n /dev/usb/lp0)
   ```

   or:

   ```
   $ udevadm test /class/usb/lp0 usb
   main: looking at device '/class/usb/lp0' from subsystem 'usb'
   main: opened class_dev->name='lp0'
   udev_rules_get_name: reset symlink list
   udev_rules_get_name: add symlink 'deskjet'
   udev_rules_get_name: rule applied, 'lp0' becomes 'usb/lp0'
   create_node: creating device node '/dev/usb/lp0', major = '180', minor     = '0
   ↪', mode = '0660', uid = '0', gid = '7'
   create_node: creating symlink '/dev/deskjet' to 'usb/lp0'
   ```

   - note that it is only a udev simulation, not the true udev creating devices, I have experimented cases where udevtest was working, but udev did not. In some case it seems it was caused by multiple devices matching the same key.

- If the device was yet present reloading the rules or restarting udev, is not sufficient to have the new device, you have to unplug the device, it can be a hot plugging when available, otherwise you need to restart the computer.

6. We must now have:

```
$ ls -l /dev/usb/lp0
crw-rw----  1 root lp 180, 0 Mar 14 21:42 /dev/usb/lp0
$ ls -l /dev/deskjet
lrwxrwxrwx  1 root root 7 Apr  7 18:23 /dev/deskjet -> usb/lp0
```

In the same way we can link a specific mass-storage to a special `/dev` entry by looking at the keys by:

```
$ udevadm info -a -p $(udevadm info -q path -n /dev/uba1)
```

then add in /etc/udev/rules.d/10-local.rules

```
BUS="usb", SYSFS{serial}="0402170100000020EB5D00000000000", KERNEL="ub?1", NAME="%k
↪", SYMLINK="usbfoo"
```

But usually it it is better to use the provided symlinks which yet allow persistent naming.

If we want to automount some removable storage we can create a rule in `/etc/udev/rules.d/` and use systemd-mount to mount it. As the recent man page explain in the section The udev database a udev rule like the following automatically mount all USB storage plugged in:

```
ACTION=="add", SUBSYSTEMS=="usb", SUBSYSTEM=="block", ENV{ID_FS_USAGE}=="filesystem
↪", \
RUN{program}+="/usr/bin/systemd-mount --no-block --automount=yes --collect $devnode
↪"
```

In this case `systemd-mount` honors the of additional udev properties `SYSTEMD_MOUNT_OPTIONS=` to give additional mount options, `SYSTEMD_MOUNT_WHERE=` The file system path to place the mount point at, instead of `/run/media/system/`.

Here to mount a specific usb key we create a rule by using some keys to identify the file system using keys from the top sysfs entry and possibly also from a parent, like this:

```
ACTION=="add"
SUBSYSTEMS=="usb"
SUBSYSTEM=="block"
ATTRS{serial}=="0709289778d6a5"
ATTR{partition}=="1"
RUN{program}+="/usr/bin/systemd-mount --no-block --automount=yes --discover
↪$devnode"
```

Here the `serial` identify the usb key so I have to add a partition number to properly identify the partition. The mounted device in `/run/media/system` directory

For disk devices you can use any persistent naming for identifying them. The same information used by symlinks in `/dev/disk/` tree and ref:*reported by ldblk <uuid_with_lsblk>*, is also reported by:

```
$ udevadm info --query=property -n /dev/sdc
```

as `ID_SERIAL`, `ID_SERIAL_SHORT`, `ID_WWN` for a disk and also for a partition filesystem `ID_FS_UUID`, `ID_FS_LABEL`, `ID_PART_ENTRY_UUID` (instead of `PARTUUID` for lbslk). These variables are available in udev rules as `ENV{``*variable*}`‘.

If I add a label on the partition of my usb key, I can now use:

```
ACTION=="add"
SUBSYSTEMS=="usb"
SUBSYSTEM=="block"
```

(continues on next page)

(continued from previous page)

```
ENV{ID_FS_LABEL}=="VFAT_SHARE"
ENV{SYSTEMD_MOUNT_OPTIONS}="gid=john,uid=john"
ENV{SYSTEMD_MOUNT_WHERE}="/run/media/john/$env{ID_FS_LABEL}"
RUN{program}+="/usr/bin/systemd-mount --no-block --automount=yes $devnode"
```

and I find the mounted device at `/run/media/john/VFAT_SHARE` with the *john* user and group.

11.4.1 debugging udev

To debug udev we can:

1. use `udevadm test`
2. log `udevd` by issuing:

   ```
   log="yes"
   ```

   in `/etc/udev.conf` and change the level of debugging with:

   ```
   $ udevadm control log_priority=level
   ```

the priority is a numerical or symbolic level from systlog **err**, **info** and **debug**

3. `udevadm monitor` reports to the console the udevd activity

CHAPTER

TWELVE

OPERATING ON DISK DEVICES

12.1 Listing devices.

12.1.1 Using `lsblk`.

You can get the block devices on your system by using lsblk:

```
$ lsblk -o NAME,TYPE,FSTYPE,LABEL,SIZE,MODEL,MOUNTPOINT
```

Which gives a schema like this:

```
sda                    disk              465.8G ST9500325AS
├─sda1                 part                 10M
└─sda2                 part              419.2G
  ├─vg0-debian (dm-0)  lvm                13.7G                  /
  ├─vg0-home (dm-1)    lvm                23.4G                  /share/home
  ├─vg0-vguests (dm-2) lvm                  56G
  └─vg0-seafile (dm-3) lvm                   2G
sdd                    disk               3.7G STORAGE DEVICE
└─sdd1                 part vfat          3.7G
sr0                    rom                1024M DVD+-RW GSA-T11N
```

You can also use the simpler `lsblk -f` with less columns.

12.1.2 Using `blkid`.

A detailled information with UUID, but not graphical is given by blkid, it should be run as root to give a full information.

```
$ sudo blkid -c /dev/null
```

The `-c /dev/null` is to prevent `blkid` from using cache, and reporting devices which are no more available.

A more readable output is with:

```
$ sudo blkid -o list -c /dev/null /dev/sd*
device     fs_type label    mount point    UUID
--------------------------------------------------------
/dev/sda2  LVM2_member      (in use)       Fdh5Am-cNBo...
/dev/sdb1  vfat             (not mounted)  B49E-A10C

Or any device with:
```

```
$ sudo blkid -o list -c /dev/null
$ sudo blkid -o list -c /dev/null /dev/sdd1
$ sudo blkid -o list -c /dev/null /dev/mapper/*
```

12.1.3 Using *udisk disk manager*.

udisksctl is a higher level control part of the udisk disk manager If it is running on your system, you get the managed devices with their model, revision and serial number by:

```
$ udisksctl status
```

but it will not give you a partition list like *lsblk* or *blkid* which I find more informative for discovering new devices.

There are many way to know what partitions compose your storage devices like *proc filesystem*, the *dev filesystem*, or the commands: fdisk, sfdisk, gdisk, sgdisk or parted.

You may have to use a second level when the partition is not a physical partition but a logical partition. lvm volumes are grouped in *volume groups* divided in *logical volumes* that you can list using lvdisplay.

If the partition host a btrfs file system, you can list the subvolumes that compose it by using btrfs-subvolume.

When you add the device entry to *udiskctl* you obtain a more detailed info:

```
$ udisksctl info -b /dev/sdd
$ udisksctl info -b '/dev/sdd1'
$ udisksctl info -p 'block_devices/sdd'
$ udisksctl info -p 'block_devices/sdd1'
```

`-p` *is an abbrev for* `--object-path` *and* `b` *an abbrev for* `--block-device`. In any case the *block-device* the *object-path* is told in the answer.

You can also use `udisksctl monitor` to monitor devices before connecting the device and see the device entry attributed by udev.

It is also shown in your kernel messages, and can be read with dmesg but it is quite laborious to find the proper line.

References

- udisksctl
- ArchWiki: Udisks

12.1.4 Using the *udev* level for usb devices.

For an usb device you can also use lsusb:

```
$ lsusb -v | grep -E \
'\<(Bus|iProduct|bDeviceClass|bDeviceProtocol)' 2>/dev/null
```

You can also get the udev keys with udevadm, you have more details in the *udev section*.

```
$ udevadm info -a  -n /dev/usb/sdb1
```

lsusb is aimed at usb devices, it can often be replaced by the more general commands *lsblk* and *blkid*.

12.1.5 Interacting with *proc* and *sys*.

If these command are not available you can work at low level with the *proc* and *sys* virtual filesystem.

```
$ cat /proc/partitions
major minor  #blocks  name
   8        0  976762584 sda
   8        1     409600 sda1
```

(continues on next page)

(continued from previous page)

```
   8        2     307200 sda2
....
$ ls -l /sys/block/*/device
lrwxrwxrwx 1 root root 0 Jan 25 21:10 /sys/block/sda/device -> ../../../0:0:0:0
lrwxrwxrwx 1 root root 0 Jan 25 21:12 /sys/block/sr0/device -> ../../../1:0:0:0
$ cat /sys/block/sr0/device/model
DVDRAM GUA0N
```

12.1.6 Using the *dev* filesystem.

udev populate the *dev* filesystem, you can explore it with file `--special-files` abridged in `file -s`, as the */dev* entry also used symlinks to find device by *label*, *id*, or *uuid*, you may need to use also the option `--dereference` (`-L`).

file -s give dor a device the type of boot sector, for a partition the file system type or *LVM* physical volumes and the *UUID*.

```
$ sudo file -s /dev/dm-*
$ sudo file -s /dev/sd*
$ sudo file -L -s /dev/disk/by-uuid/*
$ sudo file -L -s /dev/disk/by-label/*
```

12.2 Determine the file system of an unmounted partition.

When the partition is mounted the output of mount show the file system type.

You can also use df with the command:

```
$ df --print-type --human-readable
Filesystem                  Type      Size  Used Avail Use% Mounted on
/dev/mapper/vg0-root        ext3       19G   12G  6.2G  65% /
/dev/sda2                   vfat      296M   50M  247M  17%
/boot/efi
....
```

Using short options the command is `df -Th`.

When the partition is not mounted whe have seen *above* that blkid also give the partition type, and it can even be run as a user, in this case he cannot tell if the partition is in use or not, but it still gives the fs type.

When you want to know the size and alignement of a partition you can use fdisk or sfdisk with *mbr* partition table gdisk or sgdisk with *gpt* partition table, and parted with both of them to issue one of:

```
$ sudo fdisk -l /dev/sdd
$ sudo sfdisk -l /dev/sdd
$ sudo gdisk -l /dev/sdd
$ sudo sgdisk -i -p /dev/sdd
$ sudo parted /dev/sdd print
```

All these command are to be run as root.

12.3 Mounting devices as user.

To mount the device as root you can of course use the mount command, for removable devices, usually you prefer to mount them as user. You can still use mount if the fstab has a `user` option for the device, but not for arbitrary plugged devices.

The old way is to use pmount, but if you have udisdk daemon running on your system, you should use udisksctl:

```
$ udisksctl mount -b /dev/sdd1
$ udisksctl mount -b /dev/disk/by-label/key64G001
$ udisksctl mount -b /dev/disk/by-uuid/77e19bbf-84ac-4336-a738-e6563a538f7d
$ udisksctl unmount -b /dev/sdd1
$ udisksctl power-off -b /dev/sdd1
```

The *udisks* daemon mount your block device in a directory `/media/<user>` that it creates if necessary. If there is a label it is used; so the device above *key64G001* is mounted as `/media/<user>/key64G001`.

The directory `/media/<user>` belongs to root, but has an ACL giving you the `r-x` access. The directory `/media/<user>/key64G001` and its content belongs to to you with `rwx` access.

The mount directory is shown in the output of udisksctl, but if you don't remember it, or you want to use it in a script, you can get it with the command findmnt:

```
findmnt -no TARGET /dev/disk/by-uuid/77e19bbf-84ac-4336-a738-e6563a538f7d
```

You can also use udisksctl to mount a loop device:

```
$ udisksctl loop-setup -f someimage.iso
Mapped file someimage.iso as /dev/loop0.
$ udisksctl mount -b /dev/loop0
Mounted /dev/loop0 at /media/john/someimage.
```

12.4 Mounting a partition in a FileManager

The modern file managers like *Nautilus*, *Thunar*, *Pcmanfm* use *gvfs* and *udisk2* to mount removable media. They accept that you give them a *gvfs* mountpoint.

They also list a list of partition, either yet mounted, or unmouted, and allow to mount removable partitions.

Partitions listed in `/etc/fstab` would (by default) only show up if they are mounted under `/media`, `$HOME` or `/run/media/$USER` or if there is an entry in fstab for them pointing to these directories.

If you want the partition to be mounted under a different directory (e.g. `/mnt`) and still be shown in the sidebar, you can override the default behaviour by adding `x-gvfs-show` to your mount options in fstab:

Partitions not listed in `/etc/fstab` are handled by udisks2 and will be mounted under `/run/media/$USER/VolumeName` or `/media/VolumeName` depending on the value of `UDISKS_FILESYSTEM_SHARED` (see udisks you can change it in an udev rule), hence they will be shown under Devices in the sidebar.

12.5 Front ends for mounting removable devices.

You may also want to have some frontend that allows to alleviate the burden of remembering the commands or to read the manual, *but which add the the load of remembering the frontend api, and make you depend on the presence of an added piece of software*.

- bashmount is a bash script to help mounting with *udisks2*. It is not updted since 2014 but there are more recent forks.
- lightweight device mounter (ldm) (MIT License) is a lightweight daemon that mounts removable devices automatically. Ut requires only libudev, libmount and libusb. The daemon uses 3.3M resident with 2.5M shared. There are few configuration options as it relies on fstab for mounting partitions. There is no easy way to configure what you want to be mounted by the daemon and my regular partitions yet mounted on a system path get mounted again under `/mnt`.
- triggerhappy (GPL) is a hotkey daemon developed for small and embedded systems. It attaches to the input device files and executes scripts on events. It is packaged in Debian.

- udisk-glue (BSD Licence) is a daemon that can perform user-configurable actions when a certain udisks event is detected. It can be configured to automatically mount devices. *Last commit 2013*.
- udiskie *(MIT License)* is an automounter for usb devices written in python. It uses the dbus interface through *udisks*. It comes with optional mount notifications and gtk tray icon and a command-line client `udiskie-mount`. It is in *pypi*.
- UDisksEvt (GPL) by Vladimir Matveev is a daemon written ih haskell which listens for D-Bus signals emitted by UDisks daemon and execute configured actions. *Last commit 2011*
- udevil is a command line program which mounts and unmounts removable devices. Udevil is written in C with libudev and glib without dependency on udisks or gvfs. It is part of the Spacefm project whose development stopped in April 2014.
- usbmount automatically mounts USB mass storage devices when they are plugged in, and unmounts them when they are removed. The mountpoints (`/media/usb[0-7]` by default), filesystem types to consider, and mount options are configurable. If the device provides a model name, a symlink `/var/run/usbmount/MODELNAME` pointing to the mountpoint is automatically created. *usbmount* is unmaintained since 2007 as a debian package and the last release is in *Jessie*, a git repository contains some new developpements.
- udisksvm is a small (280 loc) python GUI oriented script to automount removable medias using udisks.
- udisks_functions are bash functions to help mounting and unmounting with udisks2.

All modern file managers can automount devices for lxde desktops see PCManFM

12.6 Udisks references

- *ArchWiki: :archwiki:'Udisks*.
- Gentoo: Udisks.
- Introduction to Udisks.

12.7 Loop Devices.

12.7.1 Mounting disk images

As an example I want to mount a partition in a disk image disk.img.

The output of *file* is:

```
$ file disk.img
disk.img: DOS/MBR boot sector; partition 1 : ID=0x83, start-CHS (0x40,0,1),
end-CHS (0x3ff,3,32), startsector 8192, 2048000 sectors
```

or with fdisk:

```
$ fdisk -l disk.img
Disk disk.img: 1004 MiB, 1052770304 bytes, 2056192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x081bc7ef

Device     Boot Start     End Sectors  Size Id Type
disk.img1        8192 2056191 2048000 1000M 83 Linux
```

So first partition begin at 8192 x 512 = 4194304 bytes

I can mount it with

```
$ sudo mount -o ro,loop,offset=4194304 disk.img /tmp/mnt
```

We can check the mount with findmnt(8)

```
$ findmnt /tmp/mnt
TARGET   SOURCE     FSTYPE OPTIONS
/tmp/mnt /dev/loop0 ext4   ro,relatime
```

If you want to avoid the offset computing you can use kpartx(8), which set up device mappings for the partitions of any partitioned block device.

You can also create loop devices for each partition in your disk with:

```
$ sudo losetup -f -P disk.img
$ sudo mount -o ro,loop /dev/loop0p1
```

We can also use *udisksctl* to mount the disk partition as user.

```
$ udisksctl loop-setup --file disk.img --offset 4194304
Mapped file disk.img as /dev/loop0
$ udisksctl mount -b /dev/loop0
Mounted /dev/loop0 at /media/john/5853137f-ce83-4fa5-9845-42ff0de259b4
```

For unmountting:

```
$ udisksctl unmount -b /dev/loop0
$ udisksctl loop-delete -b /dev/loop0
```

With udisksctl you can also mount the whole disk, in contrast to losetup no extra option is needed.

```
$ udisksctl loop-setup --file disk.img
udisksctl mount -b /dev/loop0p1
```

CHAPTER

THIRTEEN

BTRFS

13.1 Filesystem

For further refs see Btrfs Wiki - Basic Filesystem Commands.

13.1.1 Resizing

To resize a filesystem, the underlying device must have room to accommodate the changes, you first may have to resize the partition or logical volume.

The following commands are self explanatory.

```
# btrfs filesystem resize +2g /path/to/filesystem
# btrfs filesystem resize -2g /path/to/filesystem
# btrfs filesystem resize 20g /path/to/filesystem
# btrfs filesystem resize max /path/to/filesystem
```

You can use as unit 'K', 'M', 'G', 'T', 'P' or 'k', 'm', 'g', 't', 'p' the units are counted with a 1024 base, i.e KiB, MiB, GiB, TiB, PiB.

13.1.2 Btrfs label

You can put a label on the filesystem at creation with:

```
# mkfs.btrfs -L mylabel /dev/sdx
```

It can be changed with

```
# btrfs filesystem label /mountpoint newlabel
```

On an unmounted volume

```
# btrfs filesystem label /dev/by-uuid/1234567890 newlabel
```

13.1.3 Changing uuid

If you copy at low level a btrfs filesystem you end up with two filesystems with the same UUID, and even if you don't use the UUID for mounting your filesystem the output of mount or findmnt is incoherent mixing the two filesystems, and you cannot mount both as the kernel see the filesystem as yet mounted. Even the command

```
# btrfs filesystem show
```

see only one of the two duplicate filesystem.

So after copying the filesystem you need to change the UUID of the new fs with btrfstune(8).

```
# btrfs check /dev/<device>
# btrfstune -u /dev/<device>
```

The new UUID is randomly choosen, you can also provide a new valid and unique UUID.

```
# btrfstune -u b532bc62-9468-4fe9-83af-9e9de6aed2d4 /dev/<device>
```

13.2 Filesystem integrity

The btrfs-scrub(8) command executes as a background process for a mounted volume. It verifies the checksums for all data and metadata. If the checksum fails it marks it as bad, and if a good copy is available on another device it replaces it. This operation runs at a default IO priority of idle to minimize the impact on other active processes.

```
# btrfs scrub start /mountpoint
```

To monitor scub progress:

```
# btrfs scrub status /mountpoint
```

Scrub should be run via a periodic system service at least each month.

There is also a command to make a read-only check of metadata and filesystem structures on an unmounted btrfs filesystem using btrfs-check(8):

```
# btrfs check -p /dev/by-partuuid/UUID
```

(you can choose any device naming!).

This command is not usually necessary and scrub should be preferred.

To know what errors have been detected on a btrfs volume you can issue:

```
# btrfs dev stats /mountpoint
```

13.3 Subvolume

Ref: btrfs-subvolume(8)

Se also *Snapshots*.

13.3.1 Creating, listing, deleting

To create a subvolume:

```
# btrfs subvolume create /path/to/subvolume
```

List subvolumes under some path:

```
# btrfs subvolume list -p /path
```

Delete it

```
# btrfs subvolume delete /path/to/subvolume
```

13.3.2 Moving data across subvolumes

To move data between subvolumes you can use reflinks to avoid a physical copy of the data:

```
$ cp -a --reflink=always volume1/dir volume2/dir
$ rm -rf volume1/dir
```

It can seem surprising that a simple move don't do the work, more quickly, but mv uses the rename syscall and when a cross subvolumes move is detected, it fall back to a plain copy.

13.3.3 Mounting

When you mount a btrfs filesystem, the default subvolume mounted. The default subvolume is initially set to be the top-level subvolume which has an id of 5, but it can be changed as shown below.

If you want to mount an other subvolume you have to give as mount option either `subvolid=<id>` or `subvol=path/from/toplevel` where id is the subvolume id that you get by

```
# btrfs subvolume list -p /path/to/filesystem
ID 267 gen 5264 top level 5 path mysubvolume
```

or by

```
# btrfs subvolume show /path/to/filesystem/subvolume
```

13.3.4 Changing default subvolume

You can change the subvolume which is mounted by default by

```
# btrfs subvolume set-default <id> /path/to/filesystem
```

The top level will become accessible by mounting it as subvolume with path `/` or id 5.

13.3.5 Changing the label

The current label is displayed with

```
# btrfs filesystem label /path/to/filesystem
```

to change it:

```
# btrfs filesystem label /path/to/filesystem a_new_label
```

You can also display or change or display the label of an unmounted filesystem

```
# btrfs filesystem label /dev/mydevice a_new_label
```

13.4 Snapshots

Snapshots are btrfs-subvolume operations described in the Btrfs wiki. They are ordinary subvolumes and files data, i.e. *extents*, are shared using the COW feature of btrfs. In this aspect they differ from lvm snaphots that are done block-level.

To create a new writable snapshot:

```
# btrfs subvolume snapshot <source> <dest>/<name>
```

If you omit <*name*> the the name of the source is used.

If you want your snapshot to be readonly add the `-r` option

```
# btrfs subvolume snapshot -r <source> <dest>/<name>
```

You often want to give the snapshot a meaningfull name like

```
# btrfs subvolume snapshot /rootfs /.snapshots/root_$(date -Iminutes)
Create a snapshot of '/rootfs' in '/.snapshots/root_2019-01-19T09:29+01:00'
# btrfs subvolume snapshot -r /rootfs /.snapshots/root_$(date -Iminutes)
Create a readonly snapshot of '/rootfs' in '/.snapshots/root_2019-01-19T15:43+01:00
↪'
```

The snapshots are in their own subvolume, and the snapshot does not cross subvolume boundaries. So the previous snapshot do not include other snapshots that may reside in `/.snapshots/`.

13.5 Btrfs send

btrfs-send and btrfs-receive allow to send a snapshot to an other btrfs filesystem. As example make a snapshot and send it *for the clarity we put the date in clear*

```
# btrfs subvolume snapshot -r /rootfs /.snapshots/root_2019-01-18
```

And send it elsewhere

```
# btrfs send .snapshots/root_2019-01-18 | btrfs receive /backup/snapshots
```

Now make a new snapshot and send the difference between the two snapshots.

```
# btrfs subvolume snapshot -r /rootfs /.snapshots/root_2019-01-19
# btrfs send -p /.snapshots/root_2019-01-18 /.snapshots/root_2019-01-19 |\
  btrfs receive /backup/snapshots
# btrfs subvolume delete  /.snapshots/root_2019-01-18
```

After this operation in the `/backup` btrfs filesystem we have two readonly snapshots in the directory `/backup/snapshots` named `root_2019-01-18` and `root_2019-01-19`.

As btrfs-send and btrfs-receive communicate with a data stream, the transfer is not necessarily synchronous, and the send and receive can be on different computers by using a data transport, like ssh, to convey the stream.

The incremental backup page of btrfs wiki give more details. The same page list also some tools to automate backups, more recent tools in the Uses Cases page of the same Wiki.

13.6 Btrfs raid

We can create raid over many partitions, or over multiple logical volumes of the same group allocated to physical volumes on distinct disks.

We use here a raid between lvm volumes of the same volume group but allocated to Physical Volumes on distinct disks:

We can check that the logical volumes are on proper devices by

```
# lvs  -o lv_name,attr,lv_size,devices myvg
  LV      Attr       LSize   Devices
  mylv0   -wi-ao---- 351.56g /dev/sdc2(0)
  mylv1   -wi-ao---- 351.56g /dev/sda4(0)
```

Here the attributes means (w) writable, (i) inherited allocation, (a) active, (o) open. More details on lvs fields in Red Hat LVM Administration - Custom Report.

13.6.1 Create a raid-1 btrfs filesystem

```
# mkfs.btrfs -m raid1 -d raid1 /dev/myvg/mylv0 /dev/myvg/mylv1
```

13.6.2 Conversion to raid

```
# mount /dev/myvg/mylv0 /mnt
# btrfs device add /dev/myvg/mylv1 /mnt
# btrfs balance start -dconvert=raid1 -mconvert=raid1 /mnt
```

The balance operation duplicate the metadata with option `-mconvert=raid1` and the data with `-dconvert=raid1`, note that these two operations are independent and could have been done in two steps as shown in btrfs-device(8).

You will see the raid devices and used space on each *which should be identical after the balance operation* with

```
# btrfs filesystem show /mnt
```

or any of

```
# btrfs filesystem show  /dev/myvg/mylv0
# btrfs filesystem show 0c0c9d36-ae9f-41d0-a4fc-fb7368ddf7b1
# btrfs filesystem show myfs_label
```

The used space for data, system, metadata is shown by

```
# btrfs filesystem df /mnt
```

Or a more detailled report, with detail for each of the underlying devices by

```
# btrfs filesystem usage /mnt
```

References: btrfs-device(8), btrfs-balance(8),:man:*btrfs-filesystem(8)*.

13.6.3 Replacing a failed device

Suppose you have a Raid-1 filesystem with two devices

```
# btrfs filesystem show /mnt
Label: 'myfs'  uuid:  0c0c9d36-ae9f-41d0-a4fc-fb7368ddf7b1
      Total devices 2 FS bytes used 501.40GiB
      devid    1 size 551.56GiB used 505.06GiB path /dev/mapper/myvg-mylv0
      devid    2 size 551.56GiB used 505.06GiB path /dev/mapper/myvg-mylv1
```

If the device 2 fail the output will be

```
# btrfs filesystem show /mnt
Label: 'myfs'  uuid:  0c0c9d36-ae9f-41d0-a4fc-fb7368ddf7b1
      Total devices 2 FS bytes used 501.40GiB
      devid    1 size 551.56GiB used 505.06GiB path /dev/mapper/myvg-mylv0
      *** Some devices missing
```

If a device fail you can only mount the filesystem in degraded mode

```
# mount -o degraded /dev/myvg/lv0 /mnt
```

You have to provide a replacement device, it implies on our scheme of using pair of logical volume, to remove the failed physical volume *in our example /dev/sda4* and the lv it includes *in our example mylv1* , and providing a new PV added to *myvg* and creating a new lv in the new PV *we name it mylva for clarity, but we could use the same previous name mylv1*.

Then there are two possibilities for replacing the failed device.

```
# btrfs replace start 2 /dev/myvg/mylva /mnt
```

You can monitor the progress with:

```
# btrfs replace status /mnt
```

Or you can add a new device as we have shown before in the *btrfs add example*, and delete the missing device with:

```
# btrfs device delete missing /mnt
```

Note that we must first add a device and only after delete the misssing one, as raid-1 needs at least two devices.

You need after any addition of device to balance the filesystem, to distribute metadata and data.

```
# btrfs balance  /mnt
```

References:

- btrfs-replace(8).
- Red Hat Storage Administration Guide - Integrated Volume Management of Multiple Devices.

13.7 Btrfs References

- The main site is the Brtfs Wiki it holds on the Home page a list of Guides and articles and the Btrfs FAQ.
- Red Hat Storage Administration Guide - Chapter 6. Btrfs
- Suse storage administration - Btrfs filesystem‘_
- Oracle Linux - The Btrfs File System
- ArchWiki - Btrfs, Btrfs Tips and Tricks

CHAPTER

FOURTEEN

VIRTUAL FILE SYSTEMS

14.1 GVFS

Gvfs is a userspace virtual filesystem where mount runs as a separate processes which you talk to via D-Bus. It also contains a gio module that seamlessly adds gvfs support to all applications using the gio API. It also supports exposing the gvfs mounts to non-gio applications using fuse.

Gvfs is used through collection of daemons which communicate with each other and the GIO module over D-Bus. Supported backends include file operations sftp, ftp, webdav, smb, smb-browse, http, google drive, obexftp; medias (burn, cdda, gphoto2, mtp), *archive mounting* support, *Gnome Online Account*, admin access for local filesystem.

gvfs-goa is for Gnome Online Account see the GNOME Online Accounts (GOA) project and Debarshi Ray posts tagged "Online Account".

Archive backend allow to read and write all formats supported by libarchive:

- read and write: tar, cpio, pax , gzip , zip, bzip2, xz, lzip, lzma, ar, mtree, iso9660, compress,
- read only: 7-Zip, mtree, xar, lha/lzh, rar, microsoft cab,

Gvfs is directly enabled in all gpio enabled applications, which include gnome applications. For other applications you have to either use *Fuse* as shown below, a special bridge as Gigolo or Tramp Gvfs backend for Emacs.

14.1.1 Gvfs references

- Wikipedia: Gvfs.
- Gnome gvfs doc is a presentation of the gvfs system architecture.
- ArchWiki: File manager functionality - Mounting

14.1.2 gvfs memo

The *gvfs* daemon by itself is not big 2.6M resident (2M shared), Each of the backend daemon take also the same size, and I have usually at least 7 of them (*gvfsd-archive*, *gvfsd-trash*, *gvfsd-sftp*, *gvfs-fuse-daemon*, *gvfs-afc-volume-monitor*, *gvfs-gphoto2-volume-monitor*, *gvfs-goa-volume-monitor*). *gvfs-gdu-volume-monitor* and *gdm-simple-slave* are bigger at 4M/2M shared. Some of this daemons are launched on demand, but some of them are launched at start by systemd, the services are in `/usr/lib/systemd/user/gvfs-<backend-name>.service`. If you never use some of them like *gphoto2* and have no *iphone* to make use of the *afc* backend; they can be disabled at session start and only launch them on demand.

There is a set of command line programs starting with "gvfs-" that lets you run commands (like cat, ls, stat, etc) on files in the gvfs mounts.

As a command line example you mount a gvfs share by

```
$ gvfs-mount sftp://192.168.1.1
$ gvfs-mount smb://192.168.1.1/share
$ gvfs-mount dav://192.168.1.1/owncloud/files/webdav.php
$ gvfs-mount davs://dav.box.com/dav
```

You will unmount it with

```
$ gvfs-mount -u sftp://192.168.1.1
```

If gvfs-fuse is enabled your mount are available in `/run/user/<id>/gvfs`.

You get info on the file system by

```
$ gvfs-info sftp://192.168.1.1
$ gvfs-info davs://dav.box.com/dav
$ gvfs-info /run/user/12345/gvfs/dav:host=dav.box.com,ssl=true
```

You can then use it from any gio enabled application or in command line with

```
$ gvfs-ls sftp://192.168.1.1/my/path
$ gvfs-ls smb://192.168.1.1/
$ gvfs-cat http://192.168.1.1/path
$ gvfs-cat dav://192.168.1.1/owncloud/files/webdav.php/pim/address.txt
$ gvfs-tree smb://192.168.1.1/share
```

And unmount it by

```
$ gvfs-mount -u sftp://192.168.1.1
```

You can use it under emacs in Tramp with a slightly different syntax. To load a file:

```
/dav:user@192.168.1.1:/owncloud/files/webdav.php/pim/address.txt
```

To browse a directory:

```
/dav:user@192.168.1.1:/owncloud/files/webdav.php:
```

Archive backend allow to read and write all formats supported by libarchive:

- read and write: tar, cpio, pax , gzip , zip xz, lzip, lzma, ar
- read only: iso9660, 7-Zip, mtree, xar, lha/lzh, microsoft CAB.

To command-line use of the archive backend is a bit harder, because the path of the archive is url-encoded. If you want to read the archive whose path is `//path/to/my/archive.tgz` you do:

```
$ gvfs-mount archive://file%3A%2F%2F%2Fpath%2Fto%2Fmy%2Farchive.tgz
```

Of course it's somewhat painful to do it by hand, you can use a script to url encode and better do:

```
$ gvfs-mount archive://file$(urlencode ':///path/to/my/archive.tgz')
```

The easier way to access your `gvfs` share with a non `gpio` enabled software is to use the `gvfs-fuse` gateway. It needs that the `gvfs-fuse-daemon` is running. Otherwise launch it with

```
$ /usr/lib/gvfs-fuse-daemon ~/.gvfs
```

Then you should see the virtual file system mounted in your directory `/run/<user_id>/gvfs`

```
$ ls /run/1234/gvfs
archive.tgz sftp on 192.168.1.1
```

You can also mount your obex enabled device, (it may be a phone) by

```
$ gvfs-mount obex://[00:0F:DE:72:22:D5]
```

Other medias (gphoto2, cdda) and network file system are also available.

Gvfs is directly enabled in all gpio enabled applications,it includes gnome applications.

In many modern file managers like *nautilus* or *pcmanfm* you can directly open gvfs file system like: *sftp://192.168.1.1* or *davs://dav.box.com/dav*.

For other applications you have to either use udiskctl, *Fuse* as shown above, a special bridge as Gigolo *this is a Debian package* or the Tramp Gvfs backend for Emacs.

The gvfs deamon can automount gvfs backends, this is an option set in `/usr/share/gvfs/mounts/<gvfs service>`. The following one is set by default:

```
$ cat /usr/share/gvfs/mounts/network.mount
[Mount]
Type=network
Exec=/usr/lib/gvfs/gvfsd-network
AutoMount=true
```

To use one's own rules, create `~/.gvfs/mounts`.

14.2 MTP

14.2.1 mtp devices

Detecting mtp devices

Mtp devices will not show with *disk devices commands*

As they are usb devices they will appear with `lsusb`

```
$ lsusb
Bus 001 Device 006: ID 0fce:01b5 Sony Ericsson Mobile Communications AB
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Then you can have a mode detailed output for the device with

```
$ lsusb -v -s 001:006
```

To list them with *libmtp* use

```
$ mtp-detect
```

It will give a long detailled list of connected devices, that include the bus and device address of the device, and all its capabilities.

With *jmtpfs* you can list them with

```
$ jmtrpfs -l
```

give a short list of bus and device address of the connected devices.

With *mtp-tools* you don't need to mount your device, you connect with `mtp-connect` and use any individual tool.

To mount the device with jmtpfs use

```
$ jmtpfs /path/to/mountpoint
$ jmtpfs --device=<busnum>,<devnum> /path/to/mountpoint.
```

The second command is used when you have many devices connected.

The mountpoint appear as any filesystem and you can use the common utilities.

You unmount with `fusermount`.

```
$ fusermount -u /path/to/mountpoint
```

To automate the process you can put in your fstab

```
jmtpfs /media/mtp fuse  noauto,rw,nosuid,nodev,user 0 0
```

Then you can mount and unmount with the ordinary

```
$ mount /media/mtp
$ umount /media/mtp
```

`/media/mtp` should be in the `fuse` group, with group permissions `rwx`.

To mount using *gvfs* with the backend *gvfs-mtp*, you first need to know the location of your device, you can use `lsusb`, then use

```
$ gvfs-mount mtp://[001,006]
```

or:

```
$ gvfs-mount --device '/dev/bus/usb/001/006'
```

The command output the location of the mount point in `/run/user/<id>/gvfs`. You can get more info on the root node of the device by one of

```
$ gvfs-info mtp://[001,006]
$ gvfs-info /run/user/<id>/gvfs/<mountpoint>
```

To list the file under any node in the fs tree:

```
$ gvfs-ls mtp://[001,006]
$ gvfs-ls /run/user/<id>/gvfs/<mountpoint>
```

Again `gvfs-info` will give a detailled listing of any node, including all the GIO GFile attributes

```
$ gvfs-info mtp://[001,006]/path/to/file
$ gvfs-info /run/user/<id>/gvfs/<mountpoint>/path/to file
```

or to any folder or file under the mountpoint with

```
$ gvfs-info /run/user/<id>/gvfs/<mountpoint>/path/to/folder_or_file
```

You can then use all the gvfs file management on this folder: `gvfs-ls`, `gvfs-copy`, `gvfs-rm`, `gvfs-trash`, `gvfs-move`, `gvfs-less`, `gvfs-cat`, `gvfs-save`.

Ordinary file management command `cat`, `cp`, ... may not work on this file system as they cannot manage the gio file attributes.

You unmount it with

```
$ gvfs-mount -u mtp://[usb:001,008]
$ gvfs-mount -u /run/user/<id>/gvfs/<mountpoint>
```

Or eject with

```
$ gvfs-mount --eject mtp://[usb:001,008]
```

14.2.2 Automounting

automounting rules

```
# Sony D2005 mount & unmount rules
SUBSYSTEM=="usb", ATTR{idVendor}=="0fce", ATTR{idProduct}=="01b5", MODE="0666",␣
↪OWNER="your-login"
ENV{ID_MODEL}=="D2005", ENV{ID_MODEL_ID}=="01b5", ACTION=="add", RUN+="/usr/bin/
↪sudo -b -u your-login /usr/bin/go-mtpfs -dev=0fce:01b5 -allow-other=true /media/
↪D2005"
ENV{ID_MODEL}=="D2005", ENV{ID_MODEL_ID}=="4ee1", ACTION=="remove", RUN+="/bin/
↪umount /media/D2005"
```

For the rule to apply, you should disconnect and reconnect the device either physically or by resetting the port with:

```
# echo -n "0000:00:1d.7" | tee /sys/bus/pci/drivers/ehci_hcd/unbind
# echo -n "0000:00:1d.7" | tee /sys/bus/pci/drivers/ehci_hcd/bind
```

CHAPTER

FIFTEEN

ENCRYPTED FILE SYSTEMS

See mzlinux: Encrypted File System

15.1 DM-Crypt

15.1.1 DM-crypt encrypted loopback file device.

An example of creation and use of an aes encrypted loopback file device.

- Fill a file with random bits:

```
dd if=/dev/urandom of=/tmp/crypt.img bs=1M count=10
```

- Setup a loopback *unencrypted* device for the file system:

```
losetup   /dev/loop0 /tmp/crypt.img
```

  As the encryption is at the device mapper level the loop device is created unencrypted.

- Setup the aes encrypted device mapper:

```
cryptsetup -v -c aes luksFormat /dev/loop0
cryptsetup luksOpen /dev/loop0 crypt
```

- Create a filesystem and use it:

```
mkfs.ext3 /dev/mapper/crypt
mkdir /tmp/mnt
mount /dev/mapper/crypt /tmp/mnt
echo "foo" >/tmp/mnt/file.txt
cat /tmp/mnt/file.txt
```

- Unmount the filesystem, close the device mapper and release the loopback device:

```
umount /tmp/mnt
cryptsetup luksClose crypt
losetup -d /dev/loop0
```

CHAPTER

SIXTEEN

GNUPG MEMO

There are two gnupg programs GnuPG 1.4 is the standalone, non-modularized series, GnuPG 2.x is the new modularized version of GnuPG supporting OpenPGP and S/MIME. GnuPG 2.1 introduce number of new characteristics including a new format for locally storing the public keys, and dropping the file `secring.gpg`. `gpg` 1.4 is the standalone version of gpg it will stay for embedded and server usage, as it brings less dependencies and smaller binaries. For desktop we use gpg2 and most distribution alias the command *gpg* to *gpg2*.

Gpg commands by category

For all commands a running gpg-agent avoid to enter again your passphrase in the same session.

16.1 list of keys.

```
gpg --list-keys
gpg --list-keys Luc
```

to add the fingerprints:

```
gpg --fingerprint
gpg --fingerprint Luc
```

List only secret keys:

```
gpg --list-secret-keys
gpg --list-secret-keys 123456789D
```

keys followed by # are not usable.

The usage field use: **e** for *encrypt*, **s** for *sign*, **c** for *certify* i.e. signing an other key, **a** for *authentication* cf doc/DETAILS installed with gnupg.

16.2 signing.

To sign with your default key:

```
gpg --sign message.txt
```

in this case the result `message.txt.gpg` is compressed and not legible.

If you want to use an other private key

```
gpg -u 1234ABC8 message.txt
gpg -u 'gnu corp. inc'  message.txt
```

Here the string *'gnu corp. inc'* is a unique substring in the private key uids.

To make a clear signature:

```
gpg --clearsign message.txt
```

for a a detached signature in `/tmp/message.txt.sig`:

```
gpg --detach-sign /tmp/message.txt
```

now a detached and ascii armored signature in `/tmp/message.txt.asc`:

```
gpg --detach-sign --armor /tmp/message.txt
```

16.3 Verifying a signature.

If you have a file `message.txt` and the detached signature `/tmp/message.txt.asc`:

```
gpg --verify message.txt.asc
```

If the signature name does not match the file name:

```
gpg --verify message.txt.asc renamedfile.txt
```

For a signed or clear-signed message use:

```
gpg --verify message.txt.gpg
```

if it is also crypted the `--decrypt` operation decrypt *and* verify the signature.

16.4 Encrypting.

```
gpg --recipient Luc  --encrypt /tmp/message.txt
```

produces `/tmp/message.txt.gpg` `Luc` is **a substring** of the Id of the recipient, if ambiguous or absent you are asked for the recipient.

Note that you don't need a full uid of a key or subkey, any non ambiguous part will do. You can also use the pub key Id itself.

To encrypt to stdout with the key associated wit pub key `A897396C`:

```
gpg --recipient A897396C --output - /tmp/message.txt
```

The same operation using fingerprints:

```
gpg --recipient 123434343434343C34343434343437343 49A3434 --output - /tmp/message.
↪txt
```

You can also use a word match:

```
gpg --recipient '+Smith Frank Junior'
```

Will match a key uid containing *Frank Smith Junior*.

If you have configured `~/.gnupg/gpg.conf` with a `default-recipient` or `default-recipient-self`:

```
gpg --encrypt /tmp/message.txt
```

encrypt to the default recipient, if it is missing it will ask for a recipient.

To encrypt and *armor* in the *ASCII-armored text* `/tmp/message.txt.asc` using the key set in configuration with `default-key` (or a single private key) use:

```
gpg --recipient Luc  --armor --encrypt /tmp/message.txt``
```

To encrypt and sign with armored text:

```
gpg --recipient Luc --sign --armor --encrypt /tmp/message.txt
```

To encrypt and sign choosing as recipient the key which have an uid with an *exact* (not *substring*) mail address of `luc.smith@gnu.org`, and a specific secret key:

```
gpg --local-user 1122C3B8 --recipient '<luc.smith@gnu.org>' --output /tmp/doc.gpg \
--encrypt --sign doc.txt
```

The recipient will use the `--decrypt` option to extract *and verify the signature* of `message.txt.asc` or `doc.gpg`.

To create an encrypted archive with your default key:

```
tar -vcz dir1 dir2 file1 | gpg --encrypt --output archive.tgz.gpg
```

And you extract the tar archive with:

```
gpg --decrypt archive.tgz.gpg | tar -zx
```

Even if gpg is most often used for public key cryptography you can use it for encoding with a symmetric key. In this case GnuPG will ask for a passphrase, and the passphrase verification. The default gnupg encryption algorithm is CAST-128 also called *CAST5*, you can change it with `--cipher-algo`. To encrypt with a symmetric key use:

```
gpg --symmetric /tmp/message.txt
```

To encrypt with a symmetric key and use the plain ASCII form of output:

```
gpg --symmetric  --armor /tmp/message.txt
```

If you have yet encrypted a file in binary format and you want to transform in ascii:

```
gpg  --output message.asc --enarmor message.gpg
```

To encrypt with a symmetric key using AES256 algorithm:

```
gpg --cipher-algo AES256 --symmetric /tmp/message.txt
```

16.5 Decrypting.

```
gpg --decrypt /tmp/message.txt.asc
gpg --decrypt --output /tmp/message.txt /tmp/message.txt.asc
```

16.6 available ciphers.

List version, available cipher algorithms and compression methods

```
gpg --version
```

16.7 receiving keys.

```
gpg --recv-keys --keyserver hkp://subkeys.pgp.net 0xC9C40C31
```

The server can be omitted to use the default one in `~/.gnupg/gpg.conf`

16.8 refreshing keys.

```
gpg --refresh-keys --keyserver hkp://subkeys.pgp.net
```

or with default server:

```
gpg --refresh-keys
```

16.9 creating a key.

```
gpg --gen-key
```

you should then create a revocation certificate with:

```
gpg --ouput revoke.asc --gen-revoke FE8512E1
```

and put it in a secure place.

16.10 Exporting a key.

To export the public keys in binary format to `/tmp/keyring`:

```
gpg --output /tmp/keyring --export
```

To export Luc public key in ascii for sending by mail:

```
gpg --export --armor Luc
```

Publish a key on a keyserver (mandatory key id):

```
gpg --keyserver keys.gnupg.net --send-key FE8512E1
```

If you need to export a secret key *for using on an other computer*:

```
gpg --output /tmp/mygpgkey_sec.gpg --armor --export-secret-key  FE8512E1
```

The secrete key is a very sensible data, exporting in cleartext should only be done on a secure computer, and the file must be shreded (shred(1)) after use.

shred does not work on some filesystem like *brtfs*, if your */tmp/* is a *tmpfs* file system, you are safe to use it but you have still the problem to protect your file during transport and on the other computer.

You can better symmetric encrypt the exported private key:

```
gpg --export-secret-key  FE8512E1 | \
gpg --symmetric --armor --output  /tmp/mygpgkey_sec.asc
```

You are then asked for a password for symmetric encryption, and you private key stay protected.

16.11 importing a key.

```
gpg --import colleague.asc
```

To import from the default keyserver when you now the key ID:

```
gpg --recv-keys FE8512E1 12345FED
```

Or choose a key by name regexp:

```
gpg --search-keys somebody
```

If there are multiple strings matching `somebody` gpg will present you a menu to choose one specific key".

To import a previously exported secret key:

```
gpg --allow-secret-key-import --import /tmp/mygpgkey_sec.gpg
```

If you follow the advice to symetric encrypt the secret key:

```
gpg --decrypt   /tmp/mygpgkey_sec.asc | gpg --allow-secret-key-import --import
```

16.12 Editing your keys

To edit a key you have to select it by a substring of one of its IDs.

```
gpg --edit-key me@example.com
gpg --edit-key FE8512E1
```

present a menu with many key management related tasks, you get a list with `help`, among which:

list	list subkeys and uid
key	select subkey N
uid	select uid N
adduid	add a user ID
deluid	delete selected user IDs
revuid	revoke selected user ID
addkey	add a subkey
delkey	delete selected subkeys
revkey	revoke key or selected subkeys
expire	change the expiration date
passwd	change the passphrase
showpref	show key preferences
setpref	change key preferences
sign	sign a key
lsign	local sign a key
trust	change owner trust level
save	save and quit
quit	ask for saving and quit

16.12.1 Deleting or Revoking UID

Sometime you either change your mail address, or drop an old one, or acquire a new one. An uid cannot be modified. You have to delete or revoke the old uid, and create a new one.

For local keys you can delete components subkeys and uid, but when your key is distributed, for instance when published on a key server, it is ineffective and your old id will still be present on the keyserver, and other people keyring, see GnuPg Manual: Adding and deleting key components for explanations.

so if your key is distributed you rather want to revoke old components, and add new ones.

```
gpg> list
pub  2048R/1234567C created ......
sub  2048R/9876543F created ....
[ultimate] (1). Frank <frank.nick@mail.org>
[ultimate] (2)  Frank <frank.oldnick@prevmail.org>
gpg> uid 2
.....
[ultimate] (2)*  Frank <frank.oldnick@prevmail.org>
gpg> revuid
```

16.12.2 Trust and validity

Trust is used to mean trust in a key's owner, and *validity* is used to mean trust that a key belongs to the human associated with the key ID. So a key is *Valid* if signed by trusted people, you can manage the *trust* in the owners of your keyring by editing key trust, and you can also *validate* a key by signing it, or by doing a private (i.e. not exported and shown to other) *local signature*.

There are four trust levels: *unknown* this is the initial trust of a newly imported key, *none* i.e. untrusted, *marginal* i.e. good trust level, *full* i.e. as secure as your own key.

A key is considered valid if it meets two conditions:

1. It has been signed by you or by one fully trusted key, or by three marginally trusted keys.
2. The path length from your key down to the considered key is less or equal to five steps.

You can see an example of a marginally trusted but nevertheless not valid key in the *next subsection*.

16.12.3 Editing trust.

You change the owner trust with the *trust* subcommand:

```
gpg> trust
pub  dsa1024/ECEC8BDAA6606D75
    created: 2004-12-09  expires: never       usage: SCA
    trust: unknown       validity: unknown
sub  elg1024/D7671AF7DE8BAA37
    created: 2004-12-09  expires: never       usage: E
    [ unknown] (1). Albert Lebrazh <albert.Lebrazh@gmail.com>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

1 = I don't know or won't say
2 = I do NOT trust
3 = I trust marginally
4 = I trust fully
5 = I trust ultimately
m = back to the main menu

Your decision?

3

pub  dsa1024/ECEC8BDAA6606D75
    created: 2004-12-09  expires: never       usage: SCA
    trust: marginal      validity: unknown
sub  elg1024/D7671AF7DE8BAA37
    created: 2004-12-09  expires: never       usage: E
    [ unknown] (1). Albert Lebrazh <albert.Lebrazh@gmail.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.
```

In this example nobody else than Albert Lebrazh has signed this key, so even when I declare marginally trusting him as explained *above*, his key is still not valid, this is confirmed when I try again to edit the key.

```
gpg --edit-key ECEC8BDAA6606D75

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
```

If I am sure than the key belongs to the user, I can sign it *or lsign*; and it will become valid.

16.12.4 Changing Preferences

Key preferences are list of preferred algorithm for ciphers, digest, and compression.

If you have some old private key, it could have been created with a set of preferrence that is no longer current.

The first versions of GnuPg used a default hash of SHA1, now considerred as weak, and sha2 is preferred.

You can inspect your preferences and change them in the following way.

```
gpg> showpref
     [ultimate] (1). Frank <frank.nick@mail.org>
     Cipher: AES256, AES192, AES, CAST5, 3DES
     Digest: SHA1, SHA256, RIPEMD160
     Compression: ZLIB, BZIP2, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify
 gpg> setpref
     Set preference list to:
     Cipher: AES256, AES192, AES, CAST5, 3DES, IDEA
     Digest: SHA256, SHA1, SHA384, SHA512, SHA224
     ......
 Really update the preferences? (y/N) y
 You need a passphrase to unlock the secret key for ...
 .....
 gpg> pref
     [ultimate] (1). Frank <frank.nick@mail.org>
     Cipher: AES256, AES192, AES, CAST5, 3DES, IDEA
     Digest: SHA256, SHA1, SHA384, SHA512, SHA224
     ......
```

Here we have used pref *without argument* to reset the preferences to the default, either the server wide default set by gnupg or if you have set personal defaults in your configuration with `default-preference-list`.

You can also set preferences only for this key, see more details in GnuPg Manual: Key Management in the *setpref* description.

16.13 Replacing old dsa key by a new rsa one.

- Ana's blog: Creating a new GPG key included also in keyring.debian.org - Creating a new GPG key.
- Weblog for dkg: HOWTO prep for migration off of SHA-1 in OpenPGP

To summarize the process:

- Create a new key, using 2048-bit RSA
- Generate revocation certificate for the new key
- Add necessary uids
- Sign your new key with your old one.
- Revoke no longer used uid from the old key.
- If all uid are to be revoked, create a new one specifying *in the comment*, that the other key is to be used now.
- Publish both keys.
- Ask trusted people that use and certificated the old key to certificate the new one.
- Issue a new certification for users keys that you certified with the old key, and are still current.

16.14 References

- Wikipedia: GNU Privacy Guard
- Using the GNU Privacy Guard
- GnuPG Home Page
  - Invoking GPG is an online version of the gpg2(1) manual.
  - The GNU Privacy Handbook, (french translation)
    - GnuPG manual This manual documents how to use the GNU Privacy Guard system as well as the administration and the architecture.
    - Faq
    - list of howtos
- GnuPG Gentoo User Guide how to install GnuPG, how to create your key pair, how to add keys to your keyring, how to submit your public key to a key server and how to sign, encrypt, verify or decode messages you send or receive, or local files.
- ArchWiki: GnuPG environment variables, configuration file, encrypt and decrypt, gpg-agent, pinentry, start gpg-agent with systemd user, unattended passphrase, keysigning Parties, smartcards, troubleshooting.
- Ubuntu Documentation: Gnu Privacy Guard Howto does not bring much on usage, but has a section on web of trust,, key signing and key backup.
  - Storing GPG Keys on an Encrypted USB Flash Drive
- OpenPGP Best Practices
- gpg quickstart by Paul Heinlein, is an up-to-date beginner how-to.
- subkeys are a quite difficult feature of gnupg and not very well documented. You can read Using multiple subkeys in GPG by Adrian von Bidder and Debian Wiki:subkeys.
- Philip Zimmermann Home page Philip Zimmermann is the creator of PGP and distributes also Zfone, soft for encrypting voip telephony.

16.14.1 GPG tools

- GnuPg Helper Tools contains *watchgnupg*, *gpgv*, *addgnupghome*, *gpgconf*, *applygnupgdefaults*, *gpgsm-gencert.sh*, *gpg-preset-passphrase*, *gpg-connect-agent*, *dirmngr-client*, *gpgparsemail*, *symcryptrun*, *gpg-zip*.
- gpgv2 a stripped-down version of gpg which is only able to check signatures.
- Gpa is a graphical user interface for GnuPG. GPA utilizes GTK+ and connects to GnuPG via the GPGME library.
- gpg-preset-passphrase Put a passphrase into the cache.

CHAPTER

SEVENTEEN

FREEDESKTOP STRUCTURE

These are notes of the structure of desktops following the Freedesktop specifications.

17.1 References

- Freedesktop.org a base platform (both software and standard) for desktop software.
- Freedesktop Software
- freedesktop specifications:
  - Menu specifications
  - Mime Actions Specification
  - Shared Mime Info Specification
  - Base-Directory Specification
  - Desktop Entry Specification: the exec key
  - Recent File Specification
  - Icon Themes Specification
  - Desktop Application Autostart Specification
- python-xdg is a python library to access freedesktop.org standards, the python-xdg documentation is on ReadTheDocs.

17.2 XDG menus

- XDG Menu in LXDE
- ArchLinux: Xdg-menu
- xdg-utils is a set of command line utilities for Free Desktop: creating menus, opening files, setting mime types it includes the command xdg-desktop-menu which is used for (un)installing freedesktop menu items i.e. `.desktop` files for freedesktop compatible environments.
- To build the full desktop menu many distribution use a command to browse the freedesktop menu hierarchy xdg-menu. It is decribed in Archlinux :archwiki‘Xdg-Menu‘.
- In Debian the freedesktop menus are generated from Debian Menus System by the command install-menu with the configuration `/etc/menu-methods/menu-xdg` from the package `menu-xdg`. To use them with default config `$XDG_MENU_PREFIX` must be set to `debian-`.
- Debian is using the alternate debian menu, with the increasing spreading of Free desktop menus through `.desktop` files, it becomes more difficult to maintain a debian menu file for each package. There was

a Proposal: using Desktop entries with the Debian menu system which compare Debian menus with `.xdesktop` files. An alternative is to switch debian packaging to freedesktop menus, it is developped in Debian and application-menu policies.

17.3 XDG Default application.

The Freedesktop way of opening applications is through xdg-open and the Mime type of the file.

Each application provide a `.desktop` file that conform to Desktop Entry Specification, amon many key, value pairs this file contain a key `MimeType` that indicates the MIME Types that an application knows how to handle. An application is expected to be able to reasonably open files of these types using the command listed in the `Exec` key. It is specified in Mime Actions Specification.

Example:

```
$ cat /usr/share/applications/feh.desktop
[Desktop Entry]
Name=Feh
.......
Exec=feh %F
Type=Application
.......
MimeType=image/jpeg;image/png;image/gif;image/tiff;image/bmp;
         image/x-icon;image/x-xpixmap;image/x-xbitmap;
```

ArchLinux wiki <https://wiki.archlinux.org/> has also many related documentation : Default applications *describe mime database, xdg-open, xdg-mime, ect.* Xdg user directories, :archwiki:'Desktop Entries.

It list also some alternatives, that try to provide more flexibility than the official Freedesktop mechanism.

17.4 Managing default applications with xdg-utils.

To know the mime file type of a file we use xdg-mime:

```
$ xdg-mime query filetype example.png
image/png
```

An URI is associated with a special mime type `x-scheme-handler/scheme` where `scheme` is the URI scheme like `http`, `https`, `ftp`, `mms`, `rtsp` .... (see URI scheme handlers in freedesktop specification)

What application open this file type:

```
$ xdg-mime query default image/png
feh.desktop
```

Change the default application:

```
$ xdg-mime default geeqie.desktop image/png
```

Open a file with the default application with xdg-open:

```
$ xdg-open example.png
```

The command xdg-settings allow to change at once all defaults for web scheme handlers or other url scheme handlers:

```
$ xdg-mime query default x-scheme-handler/http
org.kde.falkon.desktop
$ xdg-mime query default x-scheme-handler/https
```

(continues on next page)

(continued from previous page)

```
org.kde.falkon.desktop
$ xdg-settings get default-web-browser
org.kde.falkon.desktop
$ xdg-settings set default-web-browser firefox.desktop
$ xdg-settings get default-web-browser
firefox.desktop
$ xdg-mime query default x-scheme-handler/http
firefox.desktop
$ xdg-settings get default-url-scheme-handler http
firefox.desktop
$ xdg-settings get default-url-scheme-handler mms
smplayer.desktop
$ xdg-mime query default x-scheme-handler/mms
smplayer.desktop
```

As seen above xdg-settings is a convenience tool that replace one or many operations that can also be done with xdg-mime, but usually we want to use the same browser for all web url

If you have the gio command from the `libglib2.x-bin` package (glib is used in GTK+ and Gnome applications) you can use it to see all packages that declare the mime type by issuing a query like:

```
$ gio mime x-scheme-handler/http
```

you can use any mime type. With gio you don't have to dig manually in the mime databases, in the file system, that we now describe.

17.5 The *mimeapps* file.

Each *mimeapps* file is composed of many sections:

1. *Default Applications*: are made of lines like:

   ```
   mimetype=application1.desktop;application2.desktop...
   ```

   They give the default to open this mime type, many defaults can be specified in the same or different *mimeapps* file they are used in their order in the file or in the *mimeapps* browse order. The first application installed is used *some may be missing*.

2. *Added Associations*: The association between an application and the mime types elle can open is usually defined in the desktop file, but this section define some added associations.

3. *Removed Associations*: removes associations of applications with mimetypes, this mean that this asociation is not used, even if present in a desktop.

Added associations should be in preference order, if a valid default application is not used the higher preferrence association will be used.

The adding and removal of associations only applies to desktop files in the current directory, or a later one, this mean that if a desktop file is defined say in `/etc/xdg` directory you cannot add or remove an association related to it in `/usr/local/share/applications/mimeapps.list` that has a lower priority.

You can see all applications are defined for each mime type in `/usr/share/applications/mimeinfo.cache`, they are not prioritized in this file which only summarize the `MimeType=` field of each desktop file.

The priority for each type is shown in the first file with this type in this list: `~/.config/$desktop-mimeapps.list`, `~/.config/mimeapps.list`, `/etc/xdg/$desktop-mimeapps.list`, `/etc/xdg/mimeapps.list`, `/usr/local/share/applications/$desktop-mimeapps.list`, `/usr/local/share/applications/mimeapps.list`, `/usr/share/applications/$desktop-mimeapps.list`, `/usr/share/applications/mimeapps.list`.

Each of these files can be absent, the `$desktop-mimeapps.list` are seldom used, in these entry *$desktop* is stand for `$XDG_CURRENT_DESKTOP` environment variable, which contain the desktop name in lowercase.

Here we have used the defaults for the examined directories, of course you should use instead the *Freedesktop Directory environment variables* instead if they are not let at their default values.

For a detailed description of the *mimeapps* traversal algorithm look at Adding/removing associations in Mime Actions Specification.

The previous utilities changes the entries in `~/.config/mimeapps.list`.

Follow the Gnome System Administration Guide instructions, if you want to add a custom MIME type for all users or add a custom MIME type for individual users.

17.6 Freedesktop Directories

The Base Directories are used when looking for for user configuration.

- XDG Base Directories are specified in Freedesktop Base-Directory Specification.
- ArchWiki: XDG Base Directory support. catalog software using the XDG Base Directory Specification.
- GNOME Goal: XDG Base Directory Specification Usage explains why and how Gnome software should implement XDG base directories, and list the present support in Gnome programs.
- ArchWiki: XDG user directories.

The freedesktop base directories that follow is used by all freedesktop compatible application. They have a default that can be overrided by exporting in your environment the variables.

- `$XDG_DATA_HOME` default `$HOME/.local/share` contains user-specific data files.
- `$XDG_CONFIG_HOME` default `$HOME/.config` contains user specific configuration files.
- `$XDG_DATA_DIRS` default `/usr/local/share/:/usr/share/` are directories seperated with a colon ':' to search for data in addition of `$XDG_DATA_HOME`
- `$XDG_CONFIG_DIRS` default `/etc/xdg` are directories seperated with a colon ':' to search for configuration files in addition of `$XDG_CONFIG_HOME`. Configurations are searched in directory order, using the first match.
- `$XDG_CACHE_HOME` default `$HOME/.cache` for temporary data.
- `$XDG_RUNTIME_DIR` temporary runtime, his life must be the session, and it must be owned by the user with access mode 0700 *see full requirement in the 'specification <http://standards.freedesktop.org/basedir-spec/latest/>'__* Usually it is set by `pam_systemd` at login and there is no need to change it. You can get its value from the environment variable `$XDG_RUNTIME_DIR`.

The user directories are the directories under `$HOME` used by your desktop to store your data their default set in `$XDG_CONFIG_DIRS/user-dirs.defaults` usually `/etc/xdg/user-dirs.defaults` it default to:

```
DESKTOP=Desktop
DOWNLOAD=Downloads
TEMPLATES=Templates
PUBLICSHARE=Public
DOCUMENTS=Documents
MUSIC=Music
PICTURES=Pictures
VIDEOS=Videos
```

These system defaults can be changed in `user-dirs.defaults`.

The program xdg-user-dirs-update is run very early in the login phase. This program reads a configuration file, and a set of default directories. It then creates localized versions of these directories in the users home directory and

sets up a config file in `$(XDG_CONFIG_HOME)/user-dirs.dirs` *defaults to* `~/.config` that applications read to find these directories.

You can customize the values in your `~/.config/user-dirs.dirs`; as an example if you have a non english locale and wish to force these directories to keep their default english names run:

```
$ LC_ALL=C xdg-user-dirs-update
```

That will create the `~/.config/user-dirs.dirs`. It also creates an `~/.config/user-dirs.locale` used to remember the locale used and allowing to translate names if it changes.

An other popular alternative to avoid to create too many directories under `$HOME` is:

```
MUSIC=Documents/Music
PICTURES=Documents/Pictures
VIDEOS=Documents/Videos
```

The Debian Wiki list the dotfiles we can find in a Debian system, their role and the programs that use them. Most of them are not yet following the XDG standard, many programs may be launched with a specific environment on command line option to make them comply with xdg satndard as explained in ArchWiki: XDG Base Directory support. You can also symlink many of these files or directories inside the corresponding XDG Base directory.

17.7 Menu specification.

The reference is Freedesktop Menu Specification see also the Gnome: Desktop Menu Specification.

- `$XDG_CONFIG_DIRS/menus/${XDG_MENU_PREFIX}applications.menu` is a file containning the XML definition of the main application menu layout, with the first match strategy you can overide the system wide menu with `$XDG_CONFIG_HOME/${XDG_MENU_PREFIX}applications.menu`.
- `$XDG_CONFIG_DIRS/menus/applications-merged/` is the default merge directory included in the `<DefaultMergeDirs>` element of the previous file.
- `$XDG_DATA_DIRS/applications/` contains a `.desktop` file for each menu item. Desktop entries are collected from all of them, but in case of name conflict the first one is used.
- `$XDG_DATA_DIRS/desktop-directories/` contains `.directory` files giving directory entries in the menu layout.

17.8 Autostart applications

- ArchWiki Autostarting, Desktop entries

Applications referenced by a `.desktop` file in `$XDG_CONFIG_DIRS/autostart` and `$XDG_CONFIG_HOME` may be autostarted by xdg compliant window managers.

In additions to generic keys, autostart `.desktop` files may contain additional keys:

- `Hidden` when true, the application is ignored
- `OnlyShowIn` and `NotShowIn` can list desktop environments in which the application is only (not?) started. These two keys are exclusives each other.
- `TryExec`: Tha application is started only when the named exec exist. It can be an absolute path or a name to be looked for in `$PATH`.

CHAPTER

EIGHTEEN

NETWORK COMMANDS

18.1 nmap

References: nmap home page: nmap.org, nmap tutorial, Nmap Reference Guide.

18.1.1 Target Specification

IPv4 address	192.168.1.1
IPv6 address	AABB:CCDD::FF%eth0
Host name	www.target.tgt
IP address range	192.168.0-255.0-255
CIDR block	192.168.0.0/16

18.1.2 Target Ports

default	1,000 most popular ports
-F	Scan 100 most popular ports
-p<port1>-<port2>	Port range
-p<port1>,<port2>,...	Port List
-pU:53,U:110,T20-445	Mix TCP and UDP
-r	Scan linearly (do not randomize)
–top-ports <n>	Scan n most popular ports
-p-65535	1 to 65535
-p40-	40 to 65535
-p-	scan ports 1-65535

18.1.3 Scan Types

-sP	Probe only (host discovery)
-sS	SYN Scan
-sT	TCP Connect Scan
-sU	UDP Scan
-sV	Version Scan
-O	OS Detection

18.1.4 Probing Options

-Pn	Don't probe
-PB	Default probe (TCP 80, 445 & ICMP)
-PS<portlist>	probe TCP ports
-PE	Use ICMP Echo Request
-PP	Use ICMP Timestamp Request
-PM	Use ICMP Netmask Request

18.1.5 Timing

Time between packets.

-T0	-T paranoid
-T1	-T sneaky
-T2	-T polite
-T3	-T normal
-T4	-T aggressive
-T5	-T insane

18.1.6 Examples

More details in nmap.org examples

nmap -v scanme.nmap.org	*verbose* scan of reserved TCP ports
nmap -sS -O scanme.nmap.org/24	stealth SYN, try to determine os
nmap -sV -p 22,53,110,143,4564 198.116.0-255.1-127	host and scan on SSH, DNS, POP3, IMAP and 4564 TCP ports
nmap -sP -PS 198.116.1.0/24	Discover hosts with TCP SYN ping scans
nmap -T4 -n -Pn -p- 198.116.0.0/16	quick scan: aggressive, no dns, no ping

CHAPTER

NINETEEN

SSH

19.1 ssh memo.

Fo ssh commands examples see *ssh commands* in the *Network Commands Memo*.

19.1.1 ssh escapes.

From escape characters in the ssh manual

`~.`	Disconnect.
`~^Z`	Background ssh.
`~#`	List forwarded connections.
`~&`	Background ssh at logout.[1]
`~?`	list escape characters.
`~B`	Send a BREAK to the remote system
`~C`	Open command line.[2].
`~R`	Request rekeying of the connection.
`~V`	Decrease the log verbosity.
`~v`	Increase the log verbosity.

19.1.2 sshfs

sshfs is the *fuse* access to ssh file systems.

You can get more information in

- sshf README
- Ubuntu: sshfs explains the usage, including mounting from fstab.
- *ArchWiki: :archwiki:'Sshfs*
- Gentoo Wiki: SSHFS

To mount sshfs

- The user must be in the group *fuse*

```
$ sudo adduser foo fuse
```

- mounting

[1] when waiting for forwarded connection / X11 sessions to terminate.
[2] currently this allows the addition or cancelation of port forwardings using the `-L`, `-R` and `-D`; or `-KL` `-KR`, `-KD`; `-h` for help.
`!command` execute a local command if the `PermitLocalCommand` option is enabled in ssh_config.

```
$ sshfs hostname: mountpoint
```

- unmounting

```
$ fusermount -u moutpoint
```

You can find below *how to use autossh to mount sshfs*.

19.2 ssh keys.

19.2.1 Key encryption

SSH protocol 2 supports DSA that ssh report as `ssh-dss`, RSA ECDSA, Ed25519 keys, the two last being instance of Curve algorithm; protocol 1 only supports RSA keys.

DSA has vulnerabilities and is deprecated in openssh 7.0, there are concerns about the security of ECDSA and it is supposed that NSA could have put backdoors in this algorithm, as Ed25519 is also technically superior we can always prefer it.

If we look at the SSH implementation comparison - hostkey format we see that the widder support is for *dss* and RSA, then ECDSA and Ed25519. Ed25519 will give you the best security,and performance but requires recent versions of client & server.

Ed25519 and ECDSA are not supported by gnome keyring until v 3.27.2 released in november 2017, so in Debian your safe with at least *buster*.

You can ever disable gnome keyring, and keep as key provider ssh-agent or gpg-agen optionally enhanced by keychain or gpg-exec this is also what suggest Kyle Manna, Ryan Daniels among others.

SSH implementation comparison: hostkey <http://ssh-comparison.quendi.de/comparison/hostkey.html> give the support of key algorithm for most of ssh software. ssh-RSA is required to be supported by ssh RFC, so is always present ECDSA is widely present; but SSH-Ed25519 is only supported by OpenSSH, and few other software like the windows clients PuTTY (since 2016) and smartFTP, the iOS and Android client TinyTerm, and the linux tiny client TinySSH.

You can also find a list of Things that use Ed25519 including a list of ssh software.

Even if Ed25519 is both secure and fast, most often for ssh what matter is the ref:*cipher performance* not the authentication speed.

19.2.2 Generating a key pair

To generate a RSA key with default keysize of 2048:

```
$ ssh-keygen
```

The `-b` option allow to choose an other key size but as state the Gnupg FAQ *Once you move past RSA-2048, you're really not gaining very much* and you loose the portability.

If you use Ed25519 all keys are 256 bits.

You can consult a list of Summary of keylength recommendations of well-known security organizations

If you wantto explore the keylength topic you have first to understand why symmetric cryptography have smaller key than asymmetric cryptography. You can also look in the Référentiel Général de Sécurité version 2.0.

If you really want a stronger key you can use Ed25519 with:

```
$ ssh-keygen -t ed25519
```

But it is a good choice only to communicate with recent OpenSSH as dicussed *above*.

The ed2519 are stored in a new format that implement a Key derivation function using many bcrypt rounds to make more difficult rainbow table attacks. This new format is the default for ed2519 and can be requested for other keys by adding the option `-o`:

```
$ ssh-keygen -o -f ~/.ssh/myspecialid_rsa
```

See *below* for details on this new format.

To know what keys and ciphers are supported by your ssh software issue:

```
$ ssh -Q key
$ ssh -Q cipher
```

It is not advisable to have a key without password since any one that get access to your private key can will be able to assume your identity on any SSH server. Nevertheless if I never use as main key a key without password, it can be acceptable to have a secondary key that allow unattended connections if you make sure that only the appropriate daemon can use it, by using *a proper authorized-keys entry like shown below*.

19.2.3 Modifying a key

To change the passphrase of an existing key:

```
$ ssh-keygen -f ~/.ssh/id_rsa -p
```

To get the public key from the private one:

```
$ ssh-keygen -f ~/.ssh/id_rsa -y
```

19.2.4 Key formats

To convert a public key to PEM format:

```
$ ssh-keygen -e -m PEM -f ~/.ssh/id_rsa.pub >id_rsa_PEM.pub
```

It works also with the private key as input, but the output is only the public key:

```
$ ssh-keygen -e -m PEM -f ~/.ssh/id_rsa >id_rsa_PEM.pub
```

You can also give to `-m` the format `RFC4716` to have a SSH2 public key or `PKCS8` to have an openssl compatible PKCS8 key.

Refs: ssh-keygen, openssl

You can convert your old key to *new key format* by:

```
$ ssh-keygen -opa 64 -f .ssh/id_rsa
```

The `-a` give the number of bcrypt rounds, and default to 16, the bigger they are the longer is the password verification time, and the stronger the protection to brute-force password cracking. As example adding to the agent with `ssh-add` a private RSA 256 bytes on my laptop gives a time of 0.004s (too small to be truly significative) but with a default of 16 rounds encryption 0.292s i.e 73 time longer, a 100 rounds encryption 1.616s 404 times longer, a 1000 rounds encryption it is 16.172 seconds 4176 longer, it means that a rainbow table attack will try one table entry for the encrypted format in the same time than 4000 entries with the unencrypted format.

Of course a slower decrypting could be annoying if you wait for each ssh-connection, but if you use the agent, and still more if you have *keychain* or *gpg-exec*. You have to wait only once.

To recognize the formats of your key you can look at the head comment of the key block.

For an RSA password less key

```
-----BEGIN RSA PRIVATE KEY-----
(base64 blurb)
```

For a RSA encrypted ssh old format

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,227...
(base64 blurb)
```

For the new format

```
-----BEGIN OPENSSH PRIVATE KEY-----
(base64 blurb)
```

You can list the type and sha256 fingerprint on any of your key with

```
$ ssh-keygen -lf ~/.ssh/myid
```

or

```
$ ssh-keygen -lf ~/.ssh/myid.pub
```

This command look in the public key, and report information there, if you provide the private key it will simply look for the public one in the same directory.

If combined with -v, a visual ASCII art representation of the key is supplied with the fingerprint.

```
$ ssh-keygen -lfv ~/.ssh/myid
```

19.2.5 authorized-keys.

- The file `authorized-keys` protocol 2 public key consist of: options, keytype, base64-encoded key, comment. Where options are separated by a comma
- You can secure ssh when using a key without passphrase by putting **options** in your authorized_keys file. Options allow you to restrict to some clients, limit port forwarding, or force the use of a predefined command. The options are listed in the SSHRC section of sshd man page that also gives some examples like

```
# Comments allowed at start of line
ssh-rsa AAAAB3Nza...LiPk== user@example.net
from="*.sales.example.net,!pc.sales.example.net" ssh-rsa AAAAB2...19Q==␣
↪john@example.net
command="dump /home",no-pty,no-port-forwarding ssh-dss   AAAAC3...51R==␣
↪example.net
permitopen="192.0.2.1:80",permitopen="192.0.2.2:25" ssh-dss  AAAAB5...21S==
tunnel="0",command="sh /etc/netstart tun0" ssh-rsa AAAA...==  jane@example.net
```

19.2.6 copying the key to a remote server

You can use ssh-copy-id to copy the file to the remote server:

```
$ ssh-copy-id -i ~/.ssh/mykeyid_rsa.pub username@remote-server.org
```

If you omit the id it will add all your keys to the remote server, either the keys returned by ssh-add -L, if nothing is in your agent it will use the most recent file that matches: `~/.ssh/id*.pub`.

When using the ssh-agent keys, ssh-copy-id will loose your comment. When you have multiple keys the comment is very usefull to remember the key role, so it is better to always give the key file with the `-i` option.

It is allowed but not recommended to specify the port or other options with ssh-copy-id like this:

```
$ ssh-copy-id -i ~/.ssh/mykeyid_rsa.pub -p 27654 -o 'X11Forward=Yes'␣
↪username@remote-server.org
```

But is is always better to put these option in ssh_config.

We can also manually copy the key, if we can ssh to the server by:

```
$ cat ~/.ssh/mykeyid_rsa.pub | ssh username@remote-server.org \
'sh -c "cat >> ~/.ssh/authorized_key; chmod 0600  ~/.ssh/authorized_key"'
```

which is similar to the previous `ssh-copy`.

In some case you have not yet an ssh access to the server, which is the case when the remote sshd has an option of `PasswordAuthentication no` or the default for root of `PermitRootLogin prohibit-password`.

You can still copy the key to the server by any mean like ssh to an allowed account, ftp, webdav, shared cloud ... If the transport media is not protected, to avoid the key being tampered, you can consider that is more secure to encrypt it during the transport with gpg or symetric encryption, but the simpler may be to compare visually the keys or some checksum.

Manually install the key on the server:

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
$ cat /path/of/mykeyid_rsa.pub >> ~/.ssh/authorized_keys
$ rm /path/of/mykeyid_rsa.pub
$ chmod 600 ~/.ssh/authorized_keys
```

19.2.7 Gnome Keyring

Gnome Keyring is a daemon that keeps user's security credentials, such as user names and passwords encrypted in a keyring file in the user's home folder. The default keyring uses the login password for encryption.

- ArchLinux: GNOME Keyring describe also how to use it without gnome.
- mozilla-gnome-keyring is a mozilla extension to replace the default password manager in Firefox and Thunderbird and store passwords and form logins in gnome-keyring. The Debian package is named *xul-ext-gnome-keyring*.

19.3 ssh agent.

An SSH agent is a program which caches your decrypted private keys and provides them to SSH client programs on your behalf.

19.3.1 Launching ssh-agent.

On Debian the `ssh-agent` is launched in the ancestors of your X session by `/etc/X11/Xsession` so it should run in your X session.

`ssh-agent` export two environments variables `SSH_AUTH_SOCK` the socket path, and `SSH_AGENT_PID` the pid of the process, so you can check a running instance with:

```
$ [ $SSH_AUTH_SOCK ] && echo "socket $SSH_AUTH_SOCK" && ps u $SSH_AGENT_PID
```

If it is not running you can launch it by:

```
$ eval $(ssh-agent)
```

In Debian default you have no ssh-agent session when in a console session, or connected from a remote site.

You can launch it from your profile, if it is not yet present.

You may use a more elaborate script to ensure you are launching an unique agent session for your user on the computer.

In the way used by default by Debian, if it is not yet done you can launch it as a parent process of a daemon with:

```
$ ssh-agent startx
```

or adding to your .xinitrc:

```
eval $(ssh-agent)
```

It is also possible to start it as a systemd user service, and you will have a global ssh-agent for your global user session, whatever it run X or not.

`ssh-agent` can be replaced by `gpg-agent` that can act as an agent both for gpg keys and ssh keys if it is run with the argument `--enable-ssh-support` you can then launch it like set in the manual

```
unset SSH_AGENT_PID
if [ "${gnupg_SSH_AUTH_SOCK_by:-0}" -ne $$ ]; then
  export SSH_AUTH_SOCK="/run/user/$UID/gnupg/S.gpg-agent.ssh"
fi
```

in the same way used for ssh you can prefer to start gpg-agent with systemd user.

Refs: ssh-agent, gpg-agent

19.3.2 Using ssh-agent.

You can list the cached keys:

```
$ ssh-add -l
2048 SHA256:4135dff81d9eff01f2319078995c06ab05feccc0S28 /home/user/.ssh/id_rsa␣
↪(RSA)
```

Add a key with:

```
$ ssh-add /path/of/key
```

Remove all keys from cache by:

```
$ ssh-add -D
```

Refs: ssh-add

19.3.3 ssh agent forwarding.

To get agent forwarding we must have the option `ForwardAgent` set, it is not recommended to set it globally because users with the ability to bypass file permissions on the remote host socket `$SSH_AUTH_SOCK` can access the local agent through the forwarded connection.

You can either do it when required by:

```
$ ssh -oForwardAgent=true user@example.com
```

or use the short option `-A`:

```
$ ssh -A user@example.com
```

or if you want to always forward agent to a specific server you trust, you can put in `~/.ssh/config`:

```
Host example.com
  ForwardAgent yes
```

in any case you can check your have forwarder your agent by looking at the value of `$SSH_AUTH_SOCK` which should be defined:

```
$ ssh -oForwardAgent=true user@example.com
Linux server 3.2.62-1  ...
....
$ echo "$SSH_AUTH_SOCK"
/tmp/ssh-4TjiNKqsGf/agent.3737
```

Refs: ssh

19.3.4 Forwarding to a sudo session.

If you are logged in a machine A with a ssh-agent running and holding your key, and you ssh to a machine B with agent forwarding in your B session you can still use your key to log in to a server C.

Now suppose you do a sudo you loose the agent because SSH_AUTH_SOCK is not exported, so you can no longer ssh to C even if your user key is authorized.

You can preserve your agent by using:

```
$ sudo -i SSH_AUTH_SOCK=$SSH_AUTH_SOCK
```

or if you want to use su:

```
$sudo SSH_AUTH_SOCK=$SSH_AUTH_SOCK su -p -l
```

Note than when using `su` the option `-p` preserve the environment that as yet be reset by `sudo` except SSH_AUTH_SOCK=$SSH_AUTH_SOCK.

If you want to do it for all your sudo sessions you could add to your `/etc/sudoers`:

```
Defaults    env_keep+=SSH_AUTH_SOCK
```

This method may not work for an other user than root because it does not have the rights to read `$SSH_AUTH_SOCK`, you have to add it either by adding it to your group and ensuring thet the group has read-write access, or using acl like:

```
$ setfacl -m otheruser:x   $(dirname "$SSH_AUTH_SOCK")
$ setfacl -m otheruser:rwx "$SSH_AUTH_SOCK"
$ sudo su - otheruser
```

Refs: su, sudo, man:*setfacl*

19.3.5 Connection sharing

You can enable connection sharing over a single network connection by setting `ControlMaster` to `yes`. **ssh** will listen for connections on a control socket specified using the `ControlPath` argument.

These feature are described in the ssh_config(5) manual page under the `ControlMaster`, `ControlPath` and `ControlPersist` options.

You can fix the control path of your connections by putting in `~/.ssh/config`

```
Host *
ControlPath ~/.ssh/sshsocket-%r@%h:%p
```

then you can set first a master connection by adding the option `-M` to your ssh command. The following connections will use the same control socket. and will not ask for any authentication If you don't want to use `-M` you can put in your ssh config

```
Host *
ControlMaster auto
```

you can also use `ask` to be asked if you want to reuse an existing connection and `autoask` to combine both options

If you use `ControlMaster` you need to specify `-o ControlMaster=no` when using ssh to do ssh tunneling.

    $ ssh -Y example.com

when your goal is to open an X11 application on the server you can use:

```
$ ssh -X -f example.com xprog
```

ssh will open the remote session, letting you enter your credentials, then background before command execution.before command execution.

You may want to allow automatic X11 forwarding to trusted servers, you can do it by putting in your `~/.ssh/config`:

```
Host example.com
  ForwardX11 yes
  ForwardX11Trusted yes
```

Note that to be able to forward connection you the server should have in its sshd_config `X11Forwarding yes` and the default is `no`, and `AllowTcpForwarding`, `X11UseLocalhost` set to `yes` which is the default. In some case you may want to change also `X11DisplayOffset`. A basic Xorg configuration including `xauth` should also be present on the remote server, but it does not imply that the remote server has a display.

Refs: ssh manual - X11 forwarding section, sshd_config(5), ssh_config(5).

19.3.6 Keychain

While ssh-agent is a daemon that cache your decrypted private keys during your session Keychain is a front-end to ssh-agent, allowing you to have one long-running ssh-agent process per system, rather than one per login session. Keychain was introduced by Daniel Robins in 2001 for Gentoo *Keychain has evolved since this article*, It is now available in most distributions.

- Gentoo Guide: Keychain.
- *ArchWiki: Keychain <SSH_keys#Keychain>*.
- man: keychain(1)

Simon Gomizelj who has previously written Envoy (GPL), a c language ssh/gpg-agent wrapper leveraging cgroups and systemd/socket activation with functionalities similar to keychain. Now advise to replace ssh-agent by gpg-agent wrapped in a systemd service.

It has set up a small new project gpg-exec to support this policy.

19.4 Ssh port forwarding

- ssh port forwarding and tunneling is explained in the Tcp forwarding section and X11 forwarding section of the man page, SSH Port Forwarding by Brian Hatch see also Compressed-TCP HOWTO by Sebastian

Schreiber.

- The general syntax for port forwarding is: -L port:host:hostport – redirect a local port to a remote host:hostport -R port:host:hostport – redirect a remote port to a local host:hostport
- An example of redirecting a local port to a remote one is a tunnel to a remote smtp server by forwarding request to local port 25 to a remote machine port 25

```
$ ssh -fN -L 25:127.0.0.1:25 remoteuser@remote.mach.in
```

Here the `-f` tel *ssh* to go to background after the session is established, so you can still enter a password before it backgrounds. `-N` tel not to execute any remote command, your ssh session will **only** be used for port forwarding.

You may want to use *autossh* to keep your forwarding alive; so you will use the options explained in the *keep alive section*

```
$ autossh -fN -M 0 -o "ServerAliveInternal 60" -o "ServerAliveCountMax 3" \
> -L 25:127.0.0.1:25 remoteuser@remote.mach.in
```

Here `-M 0` disable the *autossh* keepalive mechanism as the internal keepalive of *ssh* is preferred, to activate it we need to provide the two options *ServerAliveInternal* and *ServerAliveCountMax*.

There are many use of forward port proxy, if there is a remote hhtp server, serving *localhost:8384* (this is what provide syncthing) you can access it by forwarding from client port 8385 with:

$ ssh -fNL 8385:127.0.0.1:8384 remoteuser@remote.mach.in

and you can acces the site at *localhost:8385*.

The *sock proxy* below would also allow you to browse *localhost:8384*, but your browser would send any request through the remote host, which may go beyond what you need.

- An example of redirecting a remote port, is the *reverse ssh connection* below.
- You can also use *ssh* as *socks proxy* by:

```
$ ssh -fND 4321 user@example.com
```

and you get a socks proxy on port 4321 forwarding all traffic to example.com, you can browse the web as if you originate from example.com either to access a hidden lan or go thru a firewall. Of course you need a socks proxy enabled browser like firefox. You can use this socks with any socks-able client, but there are not many of them. So you can use a proxy relay a list of them is on the Wikipedia SOCKS page

- Beginning with version 4.3, ssh has an option to do tunneling a tun device see:
  - tun-based VPN section of the Openssh wikipedia page
  - The manual of ssh, sshd, ssh-config (references above)
  - HOWTO VPN over SSH and tun
  - Tunnels ethernet avec openssh
- If you change user over ssh via su or sudo, you will no more find your X credentials. You can take as `XAUTHORITY` environment your original `~/.Xauthority`, but it works only if the new user has access to this file. As it it not even true for root if your home is on a nfs file system, a better solution is to forward your credentials to the new user. A complete wrapper by François Gouget, sux is available on many distribution. But when we don't have it at hand we can simply do:

```
$ sudo -u <user> $SHELL -c "xauth add $(xauth list :${DISPLAY##*:}); <xprogram>
↪"
```

19.5 Keeping a ssh session alive

You can work either on the server side or the client side.

For the client you can set the configuration option ServerAliveInterval which is an intervall after wich a ssh *keepalive* message is sent to the server, *keep alive* is not enabled by default and the default `ServerAliveInterval` is 0. Note that these messages are sent through the encrypted channels and are not the same than the TCPKeepAlive messages which are TCP layer messages enabled by default, they are *spoofable* and may be blocked by firewalls; if you use `ServerAliveInterval` you can disable `TCPKeepAlive`.

`ServerAliveInterval` works in combination with ServerAliveCountMax which is the max number of such message sent, the default value is 3. If you have only set `ServerAliveInterval` to 30 you send every 30s a message, and no reponse is received after 3 messages the session is closed.

If in a script you set BatchMode to `yes` to disable password/passphrase querying, then `ServerAliveInterval` will be set to a 300 seconds default.

On the Server side you can send keep alive mesage to the client. By default ClientAliveInterval is 0 which means that the server does not send keep alive message to the client.

If you set `ClientAliveInterval 300` and ClientAliveCountMax `12` (default is 3) you send to the inactive client a keep alive message each 5mn, but drop an inactive connection after 2 hours.

All these option may be set in the ssh_config file.

19.5.1 autossh

autossh (modified BSD) is a program to start a copy of ssh and monitor it, restarting it as necessary should it die or stop passing traffic. A small included script `rscreen` or `rtmux` allow a *perpetual* ssh session. It is in Debian. To use autossh a monitoring port should be choosen using the `-M` option, but the debian version of autossh uses a wrapper to automatically select a free monitoring port.

As OpenSSH supports *keepalive* message since v 3.8 (2004), it is better to use it rather than the monitoring port so you will disable the monitoring port with `-M 0` and have ssh do itself the monitoring by setting `ServerAliveInterval` and `ServerAliveCountMax` as explained in in the above *keep alive section*.

If the *keepalive* is not set in the ssh_config file your command line looks like:

```
$ autossh -M 0 -o "ServerAliveInterval 45" -o "ServerAliveCountMax 2"␣
↪username@example.com
```

To use sshfs with autossh you can use:

```
$ sshfs -o reconnect,compression=yes,transform_symlinks,\
    ServerAliveInterval=45,ServerAliveCountMax=2,\
    ssh_command='autossh -M 0' username@example.com:/\
/mnt/remote
```

Even without using autossh you can restart automaticaly restart a ssh tunnel started from systemd by using the `Restart` option in your unit file as shown in this *ArchWiki example <Secure_Shell#Automatically_restart_SSH_tunnels_with_systemd>*.

- *ArchWiki: autossh <Secure_Shell#Autossh_-_automatically_restarts_SSH_sessions_and_tunnels>*.

19.5.2 mosh

mosh (GPL with OpenSSL exceptions) is a replacement for SSH that allows roaming, supports intermittent connectivity, and provides intelligent local echo and line editing of user keystrokes. Mosh improve ssh usability for mobile users. It is in Debian. Mosh does not use the ssh tcp protocol, but runs a terminal emulator at the server and transmits this screen to the client through udp. This udp protocol may conflict with firewall rules. Mosh cannot forward ssh-agent nor X11.

- mosh
- Mosh usage, info and FAQ.
- GitHub: keithw/mosh source repository.
- Mosh has a chrome plugin and an android client JuiceSSH.

19.6 Reverse ssh connection

This is a case study of *ssh port forward*. The tackled problem is you are on a server *serverA* which has a ssh server open on internet, either because there are no firewall, or there is a firewall but you can set a redirect for ssh connections to *serverA* we here suppose it listen on standard port 22, but it apply whatever port is used. You want to ssh outside of the lan on a machine *serverB*, which has a ssh server, but which is behind a firewall.

The solution is to go through the firewall with a tunnel. We can use any type of tunnel, a vpn connection is appropriate, but if it is only an occasional connection to set a vpn for it would be overkill. So we will use two ssh, one to establish the tunnel, the other one to connect through the tunnel.

We will redirect the remote port of ssh i.e. 22 by default, to a local port in order to bypassing the firewall on the remote lan.

On the remote *serverB* you forward the port 5022 of your *serverA* to the localhost port 22.

```
$ ssh -fN -R 5022:localhost:22 usera@serverA-ipaddress
```

Here this command should be done as *root* because only root can forward privileged port. If the ssh server on *serverB* use an unpriviliged port, you can do the tunnel even without being root.

Optionally you may want also to use the *keep alive options* to harden your tunnel.

Then on your *serverA* you connect to port 5022 on localhost:

```
$ ssh userB@localhost -p 5022
```

and don't forget when asked for a password that that you will be in fact connecting on *serverB* as *userB*.

This command don't need to be done as *root* and *userB* can also be an ordinary user.

19.7 Cipher Performances

The list of supported symmetric **cipher**, supported message integrity codes (**MAC**), key exchange algorithms (**KEX**), and **key** types are displayed by using the `-Q` option:

```
$ ssh -Q cipher
```

the result may contain aes, triple DES *superseded by aes*, blowfish, cast128, arcfour also spelled RC4, chacha20, ...

Arcfour is now known to be vulnerable to some complex attacks, so it should not be used in exposed situations; but the speed of arcfour let him stand as a good candidate on firewalled local area networks *when chacha20 is still unavailable*.

Note that chacha20 is a fast and secure algorithm, see the *speed tests* below.

Note that you can only use it if the server allow this cipher otherwise you will get an answer like:

```
$ ssh -c arcfour128 server.example.com
no matching cipher found: client arcfour128 \
server aes25.
```

SSH Implementation Comparison: Ciphers shows what cipher is supported by each ssh software, Arcfour is still suported by many server and clients, while chacha20 is only available in OpenSSH, PuTTY and TinySSH.

For *chacha20-poly1305* there are a CloudFare page showing the improvement on https when opting for *chacha20-poly1305* encryption.

We find some tests in the articles ssh speed tests that test ssh between two pentiums and OpenSSH ciphers performance benchmark that ssh from a pentium to an arm computer.

As you will see below *aes256* is very fast on Pentium, but may be quite slow on arm computers, it is why it is more important to choose your cipher for speed when transferring from or to an arm computer, when it does not involve security risks.

This article compare *scp*, *tar over ssh*, *rsync*, *sshfs* when transferring compressible or incompressible data. He shows *tar over ssh* without compression at 100MB/S while scp at 10MB/s and sshfs at 4MB/s.

In this test with a gigabit connection, compression of the tar or scp decrease the speed; of course it would be no longer true with slow links, but even then we must care that bzip2 is too slow to be used for on-the-fly compression.

The main conclusion is that to transfer a big directory on a fast lan the better is:

```
tar -cf- src | ssh -q -c chacha20-poly1305@openssh.com lanhost tar -xf- -Cdest
```

As set *above* we should replace `chacha20-poly1305@openssh.com` with `arcfour128` whenever it is unavailable.

19.8 sshd config

19.8.1 AllowUsers

To restrict to some users and hosts the ssh access, we can use the directives *Allowusers*, *AllowGroups*, *DenyUsers*, *DenyGroups*.

Allowusers can use patterns that takes the form *USER@HOST* to restrict to some user on specific hosts.

Example:

```
AllowUsers john root@119.20.143.62 root@119.20.143.116
      maint@119.20.143.*
```

19.8.2 Match directive examples

Match deirectives are more powerfull than the *Allowusers*, *AllowGroups*, *DenyUsers*, *DenyGroups* directive but need more care to setup properly.

An example of overriding settings on a per-user basis from the sshd configuration example in the *openssh* package:

```
Match User anoncvs
       X11Forwarding no
       AllowTcpForwarding no
       PermitTTY no
       ForceCommand cvs server
```

and older examples previously posted by Darren Tucker:

```
# allow anyone to authenticate normally from the local net
Match Address 192.168.0.0/24
        RequiredAuthentications default

# allow admins from the dmz with pubkey and password
```

(continues on next page)

(continued from previous page)

```
Match Group admins Address 1.2.3.0/24
        RequiredAuthentications publickey,password

# deny untrusted and local users from any other net
Match Group untrusted,lusers
        RequiredAuthentications deny

# anyone else gets normal behaviour
Match all
        RequiredAuthentications default

There's also some potential for other things too:

Match User anoncvs
        PermitTcpForwarding no

Match Group nosftp
        Subsystem sftp /bin/false
```

19.8.3 Testing new configuration

If we administer a server where the only access is through ssh we should be very careful when changing sshd configuration, or we can be locked out with no way to get in.

I use to test my configuration on the server with:

```
$ /usr/sbin/sshd -p 10000 -f /etc/ssh/sshd_config.new -d
```

which I test on a client with:

```
$ ssh -p 10000 -vvv server.example.com
```

19.9 ssh config

19.9.1 Match directive

The match directive is available also for the client since 6.4.

I use it to detect local subnets like:

```
# faster ciphers for lan
Match exec "local_ip %h"
     Ciphers chacha20-poly1305@openssh.com,arcfour128,blowfish-cbc,aes128-ctr
Match exec "local_ip --local '^119\.20\.143' %h"
     Ciphers chacha20-poly1305@openssh.com,arcfour128,blowfish-cbc,aes128-ctr
```

here local ip is a python function that match the ip associated with an hostname:

```
import socket
import re
import sys
private_re = r'^192\.168\.\d\d?\d?\.\d\d?\d?$'
private_re += '|' + r'10\.\d\d?\d?\.\d\d?\d?\.\d\d?\d?$'
private_re += '|'  + r'172\.(?:1[0-6]|2\d|3[0-1])\.\d\d?\d?.\d\d?\d?$'

def check_local(local_re, hostname):
    local = re.compile(local_re)
```

(continues on next page)

(continued from previous page)

```
    hostip = socket.gethostbyname(hostname)
    return local.match(hostip)

def main():
    import argparse
    parser = argparse.ArgumentParser(description='Match local ips.')
    parser.add_argument('hostname', help='hostname or ip')
    parser.add_argument('--local', dest='local_re', default=private_re)
    args = parser.parse_args()
    raise SystemExit(0 if check_local(args.local_re, args.hostname) else 1)

if __name__ == '__main__':
    main()
```

With these settings when I target a local subnet my settings are used, we can check it with the `-v` *verbose* option:

```
OpenSSH_6.5, OpenSSL 1.0.1f 6 Jan 2014
debug1: Reading configuration data /home/marc/.ssh/config
debug1: Executing command: 'local_ip 119.20.143.62'
debug1: permanently_drop_suid: 1206
debug1: Executing command: 'local_ip --local '^119\\.20\\.143' 119.20.143.62'
debug1: permanently_drop_suid: 1206
debug1: /home/marc/.ssh/config line 11: matched 'exec "local_ip --local '^119\\.
↪20\\.143' 119.20.143.62"'
.....
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client arcfour128 hmac-md5 none
debug1: kex: client->server arcfour128 hmac-md5 none
```

Note that if you use some special cipher for a client, you should make sure that your list include one *server compatible* cipher, it is why the well known *aes128-ctr* is included above, as a server may want to disable less secure cipher, the defaults of openssh 6.7 do not allow arcfour or blowfish, it does allow *chacha20* but it is unknown by older releases and most alternate servers.

If you administer an openssh server you can tune your ciphers, in accordance with your security and speed needs.

When connecting to a small server like Dropbear the choice of ciphers, MACs and key exchange algorithms is limited.

Dropbear can only support AES128, AES256, 3DES, TWOFISH256, TWOFISH128, BLOWFISH *disabled ny default*; look at options.h in source tree for details.

When dropbear is built for a small server some of these ciphers may be disabled.

19.10 ssh debugging

- A usual and easy problem are the permissions on your home directory, .ssh directory, and the authorized_keys file. Your home directory should be writable only by you, `~/.ssh` should be 700, all the keys and `authorized_keys` should be 600. On the client this is the easier problem, because your client clearly signal this error, it is less obvious for `authorized_keys` on the server side.
- On ssh client side you can add a `-v` option to your ssh command add more `-v` for more detailed debug
- To see authentification problems on the server tail the authentication log: `less +H /var/log/auth.log`, and the sshd.service: `journalctl -f -u ssh.service`.
- On the server run sshd in debug mode on a distinct port ex: `/usr/sbin/sshd -d -p 2222`

19.11 Fish

Fish is the acronym for Files transferred over shell protocol, it is a protocol to use SSH or RSH and Unix utilities like ls, cat or dd to transfer files. The protocol was designed for Midnight Commander and can also be used by lftp and by KDE KIO kioslave.

The fish protocol reference is midnight commander: README.fish it is also explained in Wikipedia: Files transferred over shell protocol.

You can use fish when the remote host does not provide a sftp service, as it is often the case with with dropbear *(because an openssl sftp is needed to run sftp with dropbear)* and on servers where sftp is not enabled. You need only a full ssh access to the remote host as fish requires a full rsh or ssh shell on the remote side.

19.12 SSH References

- Introduction: Wikipedia: Secure Shell, OpenSSH, SSh tunnel.

  Openssh susefaq how-to, OpenSSH FAQ

- The man pages are

ssh	Basic rlogin/rsh-like client program.
sshd	Daemon that permits you to login.
ssh_config	Client configuration file.
sshd_config	Daemon configuration file.
ssh-agent	Authentication agent that can store private keys.
gpg-agent	Authentication agent for both gpg and ssh.
ssh-add	Tool which adds keys to in the above agent.
ssh-copy-id	copy your pub key to a remote server
sftp	FTP-like program over SSH protocol.
scp	File copy program.
ssh-keygen	Key generation tool, include use of certificates
sftp-server	SFTP server subsystem (started automatically by sshd).
ssh-keyscan	Utility for gathering public host keys from a number of hosts.
ssh-keysign	Helper program for host based authentication.

- ArchWiki: ssh, Sshfs, SSH Keys, Sshguard *daemon that protects SSH and other services against brute-force attacts*.
- Red Hat Entreprise System Administrator's Guide - Chapter 9 OpenSSH
- Matt Taggart: Good practices for using ssh explains basic security rule to use ssh **client**.
- The 101 Uses of OpenSSH: Part II by Mick Bauer explain the public key crypto aspect of ssh.
- Ibm Developer Work: OpenSSH key management by Daniel Robbins introduces RSA/DSA key authentication, the second article shows you how to use ssh-agent, ssh-add and keychain. The third article explains ssh-agent authentication forwarding mechanism.
- Van Emery: Useful OpenSSL Tricks, X over SSH
- The eecs departement of berkeley has some quick text help files among with ssh.help and ssh-agent.help.
- OpenSSH certificates are not so well known, the reference is the CERTICATES section of ssh-keygen(1) '. they are distinct and simpler than X.509 certificates used in ssl and allow client and servers to authenticate in a simpler and more reliable wy than user/host keys.

  There are some tutorials on this subject: DigitalOcean: How To Create an SSH CA to Validate Hosts and Clients, Blargh: OpenSSH certificates tutorial, Using a CA with SSH.

CHAPTER

TWENTY

WPA_SUPPLICANT

See also MzLinux Wifi Page.

- Linux WPA/WPA2/IEEE 802.1X Supplicant, has support for WPA and WPA2 Supported wireless cards/drivers it includes Host AP driver for Prism2/2.5/3, Linuxant DriverLoader, Agere Systems Inc. Linux Driver (Hermes-I/Hermes-II chipset), madwifi (Atheros ar521x) ATMEL AT76C5XXx, Linux ndiswrapper, Broadcom wl.o driver, Intel ipw2100.
- wpa supplicant README.
- Example wpa_supplicant configuration file
- Debian: WiFi How To Use
- WPA support in Debian
- Quick Start Guide to Debian and WPA Wireless Security by Mike Shuey - 2006.
- ArchWiki: WPA supplicant.
- Ubuntu WiFi Doc: WPA HowTo, WiFi HowTo

20.1 Using wpa_supplicant

If you want dispense from using an heavy network manager; or if you don't have the graphic desktop; or the resources to do it you can connect to a roaming wifi ap directly with wpa_supplicant and wpa_cli. If you have a graphic interface wpa_gui will make the work a lot simpler.

You have first to set in your `/etc/network/interface` a configuration entry like:

```
iface roam inet manual
wpa-driver wext wpa-roam
```

In `/etc/wpa_supplicant_roam.conf`:

```
ctrl_interface=DIR=/var/run/wpa_supplicant
ctrl_interface_group=netdev
update_config=1

network={
    key_mgmt=NONE
    disabled=1
}
```

It must be writable by the group from which you will control the interface:

```
$ sudo chown root:netdev /etc/wpa_supplicant/roam.conf
$ sudo chmod 660 /etc/wpa_supplicant/roam.conf
```

To test your configuration you can test with debug as:

```
$ wpa_supplicant  -Dwext -iwlan0 -d -C/var/run/wpa_sup
```

Then to background the daemon:

```
$ wpa_supplicant  -Dwext -iwlan0 -B -C/var/run/wpa_sup
```

In production you usually launch wpa_supplicant with:

```
$ sudo ifup -v wlan0=roam
```

You can inspect the scanned networks and change the configuration either with wpa_gui or with the command line wpa_cli.

wpa_gui is quite simple just do scan, then edit the network following what you found, and connect.

wpa_cli is less easy, but you can do the following:

```
> scan
SCANNING
> scan_results
bssid / frequency / signal level / flags / ssid
00:0f:66:56:f1:b3       2432    196     [WPA-PSK-TKIP][ESS]     myap
> set_network 1 key_mgmt WPA-PSK
> set_network 1 psk "mysecretpassword"
> set_network 1  pairwise TKIP
```

You may prefer to use `iwlist` to scan your network as it gives more information.

```
$ iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 00:0F:66:56:F1:B3
                    Channel:5
                    Frequency:2.432 GHz (Channel 5)
                    Quality=54/70  Signal level=-56 dBm
                    Encryption key:on
                    ESSID:"myap"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                              24 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                    ....
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (1) : TKIP
                        Authentication Suites (1) : PSK
```

An example of full session, adding a new AP is

```
# wpa_cli -g/var/run/wpa_sup
> status
wpa_state=INACTIVE
address=00:c9:33:34:4f:ed
> scan
OK
> scan_result
bssid / frequency / signal level / flags / ssid
de:ad:be:ef:70:e2    2422    -48     [WPA2-EAP-TKIP+CCMP][ESS]       FreeAP_secure
de:ad:be:ef:70:e0    2422    -48     [WPA-PSK-CCMP][ESS]     AP-123456
de:ad:be:ef:70:e1    2422    -60     [ESS]   FreeAP
> add_network
0
> set_network 0 ssid "AP-123456"
OK
```

(continues on next page)

(continued from previous page)

```
> set_network 0 psk "mysecretpassword"
OK
> enable_network 0
OK
> status
bssid=de:ad:be:ef:70:e0
ssid=AP-123456
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA-PSK
wpa_state=COMPLETED
address=00:c9:33:34:4f:ed
> set update_config 1
OK
> set_network 0 proto WPA
OK
> set_network 0 key_mgmt WPA-PSK
OK
> set_network 0 pairwise CCMP
OK
> set_network 0 group CCMP
OK
> save_config
OK
```

CHAPTER

TWENTYONE

WORKING WITH PDF/DJVU

21.1 Selecting and merging.

21.1.1 Selecting pages with pdftk

Select pages 1 to 4 and 6 from a pdf document:

```
$ pdftk document.pdf cat  1-4 7  output mymix.pdf
```

Merge pages from two documents:

```
$ pdftk A=doc1.pdf B=doc2.pdf cat A1-5 B3 A6-8 output mymix.pdf
```

To create a blank page:

```
$ echo "" | ps2pdf -sPAPERSIZE=a4  - /tmp/blank.pdf
$ convert -size 1024x1448 xc:white /tmp/blank.pdf
```

Insert the blank page at some position:

```
$ pdftk A=document.pdf B=/tmp/blank.pdf cat A1-3 B1 A4-230 B1 A231-250 output␣
↪combined.pdf
```

Reference: pdftk manual.

21.1.2 Selecting pages with qpdf

Select pages 1 to 4 and 6 from a pdf document:

```
$ qpdf document.pdf --pages document.pdf  1-4,7 -- mymix.pdf
```

Merge pages from two documents:

```
$ qpdf doc1.pdf --pages doc1.pdf 1-5 doc2.pdf 3 doc1.pdf 6-8 -- mymix.pdf
```

Insert a blank page at some position:

```
$ qpdf  document.pdf  --pages document.pdf 1-3 blank.pdf 1 document.pdf 4-230 \
  blank.pdf 1 -- combined.pdf
```

Reference:QPDF Manual <http://qpdf.sourceforge.net/files/qpdf-manual.html>_ : ‘Page Selection Options.

21.2 PDF compression

21.2.1 Compressing pdf

Using Ghostscript we can do:

```
$ gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/screen\
-dNOPAUSE -dQUIET -dBATCH -sOutputFile=out.pdf in.pdf
```

To get a better resolution `-dPDFSETTINGS` can be `/screen`, `/ebook`, `/printer` or `/prepress`. They imply an image resolution of 72dpi, 150dpi or 300dpi see the `ColorImageResolution` parameter in ps2pdf options Look also at the `DownSamplexxxImage` options.

If you want to do color conversion use `ColorConversionStrategy` switch with one of `/LeaveColorUnchanged`, `/Gray`, `/RGB`, `/CMYK` or `/UseDeviceIndependentColor`. (`/RGB` is not allowed with PDF/X but correct in PDF/A)

We may try also to use `DownsampleType` with the the value `/Bicubic` to gain a better compression than the default method.

In any case it is a *lossy* compression.

We can use `qpdf` to do a content-preserving changes.

```
$ qpdf --linearize input.pdf output.pdf
```

Or `pdftk`:

```
$ pdftk file1.pdf output file2.pdf compress
```

If you have a pdf with only scanned images, you can use ImageMagick `convert` to create a pdf with jpeg compression (you will loose all text informations).

```
$ convert input.pdf -density 200x200 -quality 90 -compress jpeg \
output.pdf
```

Imagemagik allow to choose the density quality and compress type

21.2.2 Choosing the pixel density for images.

Usually we choose the pixel pixel density in dependence with the device capabilities.

Computer screen use between 72dpi and 100 dpi, but if we keep this low density for images we will not get a good result when zooming in. The better is to use 72*max zoom rate.

For printing we use 300dpi, 300dpi or 400dpi is also a good density for OCR.

For ebook reader we have a usual density close ranging from 200dpi to 300dpi, E-Ink Pearl, E Ink Carta HD, are 1430 x 1080, 265 dpi, 6.8'.

But usualy the book is zoomed out from a larger format.

6.8'=17.27cm diagonal is 4.1'=10.41cm wide and and 5.42'=13.78cm heigh.

If you have an A5 book, 148x210 mm you have to zoom out by 104/148 = 0.70 to make it full width, or 138 / 210 = 0.65 to get it full height.

But there is usually a margin on printed matters that you don't want to keep on the ebook reader. So you can often keep a 100% zoom factor, and crop the page.

If you have an A4 book, you should at least reduce by 0.70 so your 300 dpi on the tablet need only 210 dpi on the original.

21.2.3 PDF compression of Images.

There are many compression algorithm the can be applied to an image before including it in pdf. Images are stored as binary strings.

You can list the compression methods by:

```
$strings document.pdf| \
  sed -n '/\/Filter \/[a-zA-Z]*Decode/s/^.*\/Filter \(\/[a-zA-Z]*Decode\).*/\1/'p|␣
↪\
  sort -u
```

Imagemagick allow many options for decoding images listed by:

```
$ convert -list compress
```

but only some are usable, for pdf output, they are listed in the ImageMagick manual: pdf options :

Compression	image '/Filter [...]' setting
"-compress none"	'/ASCII85Decode'
"-compress zip"	'/FlateDecode'
"-compress jpeg"	'/DCTDecode'
"-compress lzw"	'/LZWDecode'
"-alpha off -monochrome -compress fax"	'/CCITTFaxDecode'
"+compress" "-compress rle" any thing else	'/RunLengthDecode'

When you use *ps2pdf* Ghostscript by default examine the image to decide between JPEG and LZW or Flate compression:

Scanning Lecture Notes – Compression compare the size of some compressions method for a 38 pages handwritten document. It shows the following resut.

compression | size MB |
Deflate | 10 |
LWZ | 9.7 |
Group 4 | 2.7 |
JBig2 (lossy) | 2.1 |
djvu (lossless) | 2.2 |
djvu (lossy) | 1.8 |

djvu files are generated by minidjvu.

An other test in PDFs vs PNG vs JBIG2 show the following results:

compression | size MB |
jpeg | 43.8 |
png | 6.9 |
jbig2 | 0.9 |
jbig2 upscaled | 1.4 |

21.3 Extracting objects from pdf

21.3.1 using mutool

mutool extract objects in the current directory, so you better act in a dedicated subdirectory:

```
$ mkdir extracted; cd extracted
```

To see the list of objects

```
$ mutool info ../document.pdf
Info object (897 0 R):
<</CreationDate(D:20110929010358Z)/ModDate(D:20111031125854+11'00')/Producer(ABBYY␣
↪FineReader 8.0 Professional Edition; modified using iTextSharp 5.0.6 \(c\) 1T3XT␣
↪BVBA)>>

Pages: 274

Retrieving info from pages 1-274...
Mediaboxes (28):
    1   (901 0 R):  [ 0 0 485.28 702.36 ]
    2   (1 0 R):    [ 0 0 485.16 723.84 ]
....
Fonts (10):
    3   (4 0 R):    TrueType 'TimesNewRomanPSMT' (855 0 R)
    ....
Images (274):
    1   (901 0 R):  [ DCT ] 1348x1951 8bpc DevRGB (905 0 R)
    2   (1 0 R):    [ JBIG2 ] 4043x6032 1bpc DevGray (3 0 R)
...
    274 (820 0 R):  [ DCT ] 1348x1939 8bpc DevRGB (823 0 R)
```

The second image is a jbig2 compressed png 4043x6032 monochrome and is the object number is found in the last column which read here `(3 0 R)` so the object number is `3`.

To extract an image:

```
$ mutool extract ../document.pdf 3
extracting image img-003.png
```

The same command can also extract fonts, but here the given object nomber is to the one of the font. To extract the `TimesNewRomanPSMT` you do

```
$ mutool extract ../document.pdf 857
extracting font TimesNewRomanPSMT-0857.ttf
```

21.3.2 Using pdfimages

To list images:

```
$ pdfimages -list document.pdf
page   num  type   width height color comp bpc  enc interp  object ID x-ppi y-ppi␣
↪size ratio
--------------------------------------------------------------------------------
↪---------
   1     0 image    1348  1951  rgb     3   8  jpeg   no       905  0   201   200 ␣
↪963K  12%
   2     1 image    4043  6032  gray    1   1  jbig2  no         3  0   600   601 ␣
↪ 30B 0.0%
```

(continues on next page)

(continued from previous page)

```
   3     2 image    4046  6034  gray    1   1  jbig2  no         6  0   600   600␣
↪3742B 0.1%
   4     3 image    4043  6032  gray    1   1  jbig2  no         9  0   600   601 ␣
↪ 30B 0.0%
```

The object number is important for extracting the images. The listing is more detailled than the one you get with `mutool info` or `qpdf --show-pages`.

Then you can extract the images either in the native format with:

```
$ pdfimages  -all -f 3 -l 3 document.pdf document-images
```

That generate a `document-images-000.jb2e` in the original jbig2 format.

The jbig2 format is patent protected from IBM and Mitsubishi. JBIG2 is designed for lossy or lossless encoding of 'bilevel' (1-bit monochrome) images at moderately high resolution, and in particular scanned paper documents. In this domain it can be very efficient, offering compression ratios on the order of 100:1. JBIG2 images can be included in PDF from version 1.4. It is very similar to the JB2 compression scheme used in the DjVu file format, but JB2 is open source.

To manipulate jbig2 file you can use the open source encoder jbig2enc or decoder jbig2dec from ghostscript (man [man:jbig2dec]), which can decode jbig2 to png or pbm.

If you need to use the image out of pdf, you may prefer a more usual format than `jbig2` and do:

```
$ pdfimages  -png -f 3 -l 3 document.pdf document-images
```

to get a `document-images-000.png`. Note that you get images in `jbig2`, `jpeg`, `jpeg2000` if they are yet in this format in the pdf stream, the only available conversions are to `pbm`, `tiff` and `png`.

21.3.3 Using pdf-parser.py

To look at the description of object containing images in a document:

```
$ ./pdf-parser.py -s '/Subtype /Image' document.pdf
obj 6 0
 Type: /XObject
 Referencing:
 Contains stream

  <<
    /Type /XObject
    /Subtype /Image
    /BitsPerComponent 8
    /Width 773
    /Height 279
    /ColorSpace /DeviceRGB
    /Filter /DCTDecode
    /Length 18841
  >>

obj 7 0
 Type: /XObject
 Referencing:
 Contains stream

  <<
    /Type /XObject
    /Subtype /Image
    /BitsPerComponent 8
    /Width 587
```

(continues on next page)

(continued from previous page)

```
    /Height 480
    /ColorSpace /DeviceRGB
    /Filter /DCTDecode
    /Length 40962
  >>
  ...
```

21.3.4 Encoding pdf with jbig2.

The jbig2 options are:

- `-d | --duplicate-line-removal`: When encoding generic regions each scan line can be tagged to indicate that it's the same as the last scanline This is an option because some versions of `jbig2dec` cannot handle this.
- `-p | --pdf`: Encode with PDF format for JBIG2 streams. In symbol mode the output is to a series of files: `symboltable` and `page-`*n* (numbered from 0)
- `-s | --symbol-mode`: use symbol encoding. Turn on for scanned text pages it implies symbol recognition and encode each recognized symbol only once.
- `-t <threshold>`: sets the fraction of pixels which have to match in order for two symbols to be classed the same increasing this will increase the number of symbol classes.
- `-T <threshold>`: sets the black threshold (0-255). Any gray value darker than this is considered black. Anything lighter is considered white.
- `-r | --refine <tolerance>`: (requires `-s`) turn on refinement for symbols with more than `tolerance` incorrect pixels. (10 is a good value for 300dpi, try 40 for 600dpi). Note: this is known to crash Adobe products.
- `-O <outfile>`: dump a PNG of the 1 bpp image before encoding. Can be used to test loss.
- `-2` or `-4`: upscale either two or four times before converting to black and white.
- `-S` Segment an image into text and non-text regions. This isn't perfect, but running text through the symbol compressor is terrible so it's worth doing if your input has images in it (like a magazine page). You can also give the `--image-output` option to set a filename to which the parts which were removed are written (PNG format).

```
$ .jbig2 -s --pdf *.pbm
$ python pdf.py output > jbig2_pbm.pdf
$ rm output.*
```

```
$ jbig2 -d -p -s *.jpg; pdf.py J > JBIG2.pdf

$ jbig2 -s -p -d -v *.jpg; pdf.py > jbig2_doc.pdf
```

Jbig2enc manual

21.4 Creating djvu document

21.4.1 Bitonal document

You can create a bitonal djvu page with:

```
$ cjb2 -clean image.pbm page.djvu
```

`-clean` removes small marks caused by noise and dust during the scanning process.

To get a smaller file you can try `-lossy` and check that the visual quality of the page is unchanged.

The default resolution is 300 for *pbm* files and unchanged fot *tiff* file, but you can choose a specific resolution with `-dpi`.

You can also use *minidjvu* to encode multiple bitonal pages:

```
$ minidjvu --clean image_*.pbm document.djvu
```

minidjvu has options similar to the previous *cjb2* ones: `-clean`, `--lossy`, `--dpi`, you can also fine tune the lossy encoding with `--aggression`, `--erosion`, `--smooth`, the `--lossy` option is a shortcut for `--clean --erosion --aggression 100 --smooth`

If there i a loss of quality with `--lossy` you can try to drop `--clean` and `--erosion` with `--aggression 100 --smooth` wich can also be written `--match --smooth`.

Refs: cjb2(1), minidjvu(1)

21.4.2 Colour document.

Refs: didjvu(1), c44(1), Créer un fichier DjVu/Linux

21.5 Working with djvu.

21.5.1 Bundling djvu pages.

To bundle individual pages in a document:

```
$ djvm -c page_*.djvu document.djvu
```

To list the components of a document:

```
$ djvm -l document.djvu
```

To insert a new page as 18th page in a document:

```
$ djvm -i document.djvu page.djvu 18
```

To remove the 18th page:

```
$ djvm -d document.djvu 18
```

To insert an empty blank page first create a new blank page with the same size, and the same resolution than your book pages:

```
$ convert -size 2583x3354 xc:white /tmp/blank.tiff
$ cbj2 -dpi 600 /tmp/blank.tiff blank.djvu
```

then insert it as previously:

```
$ djvm -i document.djvu blank.djvu 2
```

Refs: djvm(1)

21.5.2 Managing djvu outline.

The outline is represented by S-expressions in a textual file.

To print the current outline:

```
$ djvused -e "print-outline" document.djvu
```

To replace the outline by a new one create a text file with the textual representation of the outline:

Your file looks like:

```
$ cat outline.el
(bookmarks
 ("Contents"
  "#5" )
 ("Preface"
  "#20" )
 ("chapter 1"
  "#31"
  ("Section 1.1"
   "#31"
   ("The first subject"
    "#32" )
   ("The second subject"
    "#36" )
   ("The third subject"
    "#41" )
   ("The fourth subject"
    "#46" ) ) ) )
```

Then insert your outline with:

```
$ djvused -s -e "set-outline "outline.el" document.djvu
```

Refs: djvused(1)

21.5.3 Setting the page numbers.

To renumber some page:

```
$ djvused -s -e "select 1; set-page-title C0;" document.djvu
```

If you have many pages to renumber, create a script to generate your *djvused* program.

```
$ cat repaginate.sh
#!/bin/sh
echo <<EOF
select 1; set-page-title C0;
select 262; set-page-title C4;
EOF
for i in $(seq 2 261)
do
  echo "select $i; set-page-title $(($i+1));"
done

$ sh repaginate.sh > /tmp/repaginate.djvused
$ djvused -s -f /tmp/repaginate.djvused document.djvu
```

Refs: djvused(1)

21.6 PDF and DJVU bookmarks

21.6.1 Bookmark conversion

bmconverter.py converts between the bookmark description formats used by different pdf and djvu bookmarking tools such as pdftk, the iText toolbox, pdfLaTeX pdfWriteBookmarks, jpdftweak, djvused, and the DJVU Bookmark Tool.

bmconverter.py is available in GitHub.

21.6.2 pdf bookmarks with pdftk.

pdftk - The PDF Toolkit (GPL) is a java *compiled (with gcj)* application which uses the iText library (LGPL).

It can merge, split, rotate, encryt, decrypt, attach files, unpack, repair pdf documents. It allows also to fill PDF Forms with FDF data or XFDF data and flatten Forms.

pdftk allows also to edit bookmarks, by dumping bookmarks to a text file, and importing bookmarks from a text file. The bookmark format is specific to *pdftk*; but the *bmconverter* program allow to convert it from and to other formats.

To dump the bookmarks do

```
$ pdftk document.pdf dump_data output bookmarks.txt
```

You can the work with the text file *bookmark.txt* then

```
$ pdftk document.pdf update_info bookmarks.txt output document_new.pdf
```

The python3 script booky is a *pdftk* wrapper that uses a simpler bookmark format.

CHAPTER

TWENTYTWO

DVD AND CD

22.1 Knowing about your drive.

To know what are your devices

Or with *xorisso* or *cdrskins*:

```
$ xorriso -devices
$ xorriso -device_links
$ cdrskin --devices
$ cdrskin --device_links
```

with wodim:

```
$ wodim --devices
```

You may have to use it with sudo.

while *xorriso* and *cdrskin* find any device, *wodim* works only when you have the *obsolete* symlinks `/dev/scd*`. The following *-prcap* and *-checkdrive* are free from these limitations and usually find your device even if `--devices` fail.

You may also find the drive by looking at `/dev/sr*` or the symlinks `/dev/cdrom`, `/dev/cdrw`, `/dev/dvd`, `/dev/dvdrw`, and check with the following *-checkdrive dev=/dev/dvd* command.

The most basic and secure way, is to use directly the proc filesystem with:

```
$ cat /proc/sys/dev/cdrom/info
CD-ROM information, Id: cdrom.c 3.20 2003/12/17

drive name:          sr0
drive speed:         62
....
Can write CD-R:      1
Can write CD-RW:     1
Can read DVD:        1
Can write DVD-R:     1
Can write DVD-RAM:   1
.....
```

To have a summary of the model and capacities of your drive:

```
$ wodim dev=/dev/sr0 -checkdrive
$ cdrskin dev=/dev/sr0 -checkdrive
$ xorrecord dev=/dev/sr0 -checkdrive
```

xorrecord is a shortcut for *xorriso -as cdrecord*.

Without *dev* parameter *wodim* or *cdrskin* will check all cd devices, *xorriso* needs a device.

To get all the capacities of your cd/dvd driver and of the media inserted:

```
$ wodim dev=/dev/sr0 -prcap
```

This `-prcap` command is not implemented by *xorriso* or *cdrskin*, it works only with the original *cdrecord* or *wodim*, and as *-checkdrive* explore all drives when *dev* is not given.

22.2 Working with iso images.

To get the label of a dvd:

```
$ dd if=/dev/sr0 bs=1 skip=32808 count=32
```

This work also with an image file:

```
$ dd if=/boot/grml/grml64-full_2017.05.iso  bs=1 skip=32808 count=32 2>/dev/null
grml64-full 2017.05
```

To mount *read only* an iso image as root:

```
# mount -t iso9660 -o ro,loop /path/to/image.iso /mountpoint
```

The system usually can guess the options, so you can use:

```
# mount  /path/to/image.iso /mountpoint
```

Unmount as usual:

```
# umount /mountpoint
```

This work also for a disk inserted in a drive, but usually you have yet put in your fstab a line like:

```
/dev/sr0   /media/cdrom0   udf,iso9660 user,noauto   0     0
```

so a simple user can mount the device with:

```
$ mount /media/cdrom0
```

To mount an iso image as user:

```
$ udisksctl loop-setup -r -f /path/to/image.iso
```

The option `-r` means read-only and `-f` gives the path of the file.

and mount it under `/media/$USER/` with:

```
$ udisksctl mount -b /dev/loop0p1
```

Where you replace *loop0* with the loop device given by `loop-setup`, don't forget to add the partition *p1*.

As usual unmount it with:

```
$ udisksctl unmount -b /dev/loop0p1
```

and detach the loop device with:

```
$ udisksctl loop-delete -b /dev/loop0
```

You can also mount true cd/dvd disk devices with udisksctl.

```
$ udisksctl mount -b /dev/sr0
$ udisksctl unmount -b /dev/sr0
```

udisksctl allow to dispense with the *user mount in fstab*

22.3 Make iso image from a directory.

Make an iso image of a directory with Joliet `-J`, and Rock Ridge `-R` extensions. Use a label (max 32 chars) `-V`, be verbose `-v`. *Joliet is only useful to use it on windows*.

```
$ genisoimage -v -o cd.iso -V DISK_LABEL -R -J /path/to/cd_dir
```

If you want to use this iso on an other system, you don't want to keep owner and acces bits, so you will replace `-R` with `-r` to get ownership cleared to uid and gid 0; read access to each file and execute for everybody if the file was executable.

xorriso -as mkisofs aliased as *xorrisofs* use exactly the same options:

```
$ xorrisofs -v -o cd.iso -V DISK_LABEL -r -J /path/to/cd_dir
```

22.4 Extract an iso image from a CD/DVD.

Some media types will possibly return more bytes than those found in the ISO image, because cd writers are allowed to add "run out" sectors at the end of an iso9660 image. This trailing garbage MAY HAPPEN with CD written in TAO mode, incrementally recorded DVD-R[W], formatted DVD-RW, DVD+RW, BD-RE, and also with USB keys.

Nevertheless if you copy the full disk content, may be constitued by an iso file and garbage trailing sectors, it will still be mountable. It should still fit onto a medium of the same type as the medium from which the image was copied.

So if you want to copy the full content:

```
$ dd bs=2048 if=/dev/sr0 of=isoimage.iso status=progress
```

If you want to extract only the iso9660 image, first determine the size of the image with:

```
$ isosize -x /dev/sr0
sector count: 2309214, sector size: 2048
```

Then extract with:

```
$ dd if=/dev/sr0 of=isoimage.iso bs=2048 count=2309214 status=progress
```

22.5 Verifying the burnt image.

First you have to know the hash of the iso image, either you have a sha that you have used to check a download was correct or you compute it.

Any hash sum will do the job, distributions usually use sha256 or sha512, and even if md5 is now to be avoided, it is still much used.

```
$ sha256sum isoimage.iso
```

Then you can either extract the size in blocks of the isoimage on disk *like shown previously* or use the size of of the isoimage file. Both numbers should be the same, *or your write surely failed*, but reading from hard disk is quicker.

And you compare this number of sectors ignoring *garbage trailing sectors*.

```
$ isosize -x isoimage.iso
sector count: 2309214, sector size: 2048
$ dd if=/dev/sr0 bs=2048 count=2309214 | sha256sum
```

22.6 Media Type and Capacity

For a dvd dvd+rw-mediainfo gyve the type, available speeds, status, number of sessions, capacity and free blocks of the media.

```
dvd+rw-mediainfo /dev/sr0
```

wodim -prcap gives the type of inserted media, and read and write speed, but not the capacity.

The type of media and supported modes are also given by:

```
$ wodim dev=/dev/sr0 -atip
```

To know the capacity of CD or DVD use *cdrskin* or xorriso :

```
$ cdrskin dev=/dev/sr0 --tell_media_space
2298496
$ xorriso -dev /dev/sr0 -tell_media_space
Drive current: -dev '/dev/sr0'
Media current: DVD-RW restricted overwrite
Media status : is blank
Media summary: 0 sessions, 0 data blocks, 0 data, 4488m free
Media space  : 2297856s
```

Here `2298496` is the number of 2kiB sectors, so the capacity is `2298496/512 = 4489.25MiB` or `2298496/(512*1024) = 4.3840GiB`.

22.7 Burn an iso image

To burn a CD with wodim:

```
$ wodim -v dev=/dev/sr0 -dao /path/to/file.iso
```

`-dao` *disk at once* is used for a single session, a multi session would require `-tao` *track at once*.

Used CD-RW media need to be erased before you can rewrite them, a *fast* blank is sufficient, you may also want to and eject the drive at the end of write:

```
$ wodim -v dev=/dev/sr0 blank=fast -dao -eject /path/to/file.iso
```

It is not recommended to use *wodim* with DVD or Blu-ray.

To burn a CD, DVD or Blu-ray with *xorriso* or *cdrkit*:

```
$ cdrskin -v dev=/dev/sr0 -dao /path/to/file.iso
$ xorriso -as cdrecord -v dev=/dev/sr0 -dao /path/to/file.iso
```

`-dao` has only meaning for CD and DVD-R, DVD-RW, let *xorriso* or *cdrkit* choose the write mode for other medium or multi-session.

As we have seen used CD-RW need to be blanked before rewrite, this is also true for DVD-RW but DVD-RAM, DVD+RW, BD-RE are overwritable without blanking.

```
$ cdrskin -v dev=/dev/sr0 -dao blank=fast -eject  /path/to/file.iso
$ xorriso -as cdrecord -v dev=/dev/sr0 -dao blank=fast -eject /path/to/file.iso
```

In addition to *fast*, *xorriso* and *cdrskin* have a value *as_needed* which apply the proper blanking to the media, and resolve to *fast* for used CD-RW or DVD-RW.

To write a DVD or Blu-ray with *growisofs*:

```
$ growisofs -dvd-compat -Z /dev/sr0=/home/user/file.iso
```

CHAPTER

TWENTYTHREE

TMUX

In this page I suppose that the tmux prefix is the default one that is `C-B` *Control B*, you can change the prefix, then replace `B` in the following page by the appropriate symbol.

In this memo the list of keys, commands, options of a command are not exhaustive. For a full list look at tmux(1).

The tmux server manage multiple **sessions** wich can be attached to zero, one or many clients. Each session is composed of **windows** which are made up of one or more **panes**. Each pane run a pseudo terminal.

You lanch a new tmux session by the command:

```
$ tmux
```

Which open a new window in a new session. You can interact with tmux with commands, the more usuals are bound to key shortcuts.

When you're lost in a tmux session, you will get all the defined the keys by typing `C-B ?` inside the current window.

23.1 Table of Tmux Keys

23.1.1 Sessions

`s`	list and switch to sessions
`$`	name session
`d`	detach current client
`(`	switch to the previous session
`)`	switch to the next session

23.1.2 Windows

c	create window
w	choose windows from list
n	next window
p	previous window
<digit>	go to window n° <digit>
f	find window
,	name window
&	kill window
[	enter *copy mode*
Page Up	enter *copy mode* and scroll one page up
]	paste last coppied text
#	list all paste buffers
=	choose paste buffer and paste
?	show keys inside a *copy mode* window
:	prompt for entering commands
t	show a big clock

23.1.3 Panes

%	vertical split
"	horizontal split
o	next pane
q	show panes numbers[1],
x	kill pane
+	break pane into window
-	restore pane from window
	space - toggle between layouts
{	swap current pane with previous one
}	swap current pane with next one
z	toggle pane zoom
Up	go to upper pane
Down	go to lower pane
Left	go to left pane
Right	go to right pane
C-Up	resize pane up by 1 lines
M-Up	resize pane up by 5 lines
C-Down	resize pane down by 1 line
M-Down	resize pane down by 5 lines
C-Left	resize pane left by 1 lines
M-Left	resize pane left by 5 lines
C-Right	resize pane right by 1 line
M-Right	resize pane right by 5 lines

23.1.4 Copy mode

C-b [switch to *Copy mode*, then q comes back to default mode.

In *copy mode* we can move with the arrow keys, and *Page Up/Down*

[1] if you type the number you go to this pane.

There are two modes for key bindings: *emacs* is the default, you can switch to vi mode by the command `setw mode-keys vi`, to make it permanent put in your configuration:

```
setw -g mode-keys vi
```

In *vi mode* we use h, j, k, and l to move around our buffer.

The following keys are bound in copy mode *(for an exhaustive list see* tmux(1) *)* :

	Function	vi	emacs
Move by characters	up	k	Up
	down	j	Down
	left	h	Left
	right	l	Right
Move in the line	Start of line	0	C-a
	End of line	$	C-e
	Back to indentation	^	M-m
	Next word	w	M-f
	Previous word	b	M-b
Jump in line	Jump forward <char>	f<char>	f<char>
	Jump backward <char>	F<char>	F<char>
	Jump next occurrence	;	;
	Jump previous occurence	,	,
Search	Search forward	/	C-s
	Search backward	?	C-r
	Search again	n	n
	Search again in reverse	N	N
Move to a line	Goto line	:	g
	Bottom line	L	
	Middle line	M	M-r
	Top line	H	M-R
Move by pages	Half page up	C-u	M-Up
	Half page down	C-d	M-Down
	Next page	C-f	Page down
	Previous page	C-b	Page up
Scroll	Scroll up	C-Up or C-y	C-Up
	Scroll down	C-Down or C-e	C-Down
Selection	Start selection	Space	C-Space
	Clear selection	Escape	C-g
	Copy selection	Enter	M-w
	Paste buffer	p	C-y
Delete	Delete entire line	d	C-u
	Delete to end of line	D	C-k
Misc	Quit mode	q	Escape
	Transpose chars		C-t

23.2 Tmux commands

All tmux commands can be either issued by a command line

```
$ tmux <command> <flags>
```

or in a tmux session by opening the command prompt with `Ctrl_B :`.

The default command is `new-session` and can be omitted, so the command

```
$ tmux
```

start a new session.

23.2.1 Common sessions commands

`new -s sessname`	start new *sessname* session
`attach`	attach to any open session
`a`	attach to any open session
`attach -t sessname`	attach to *sessname* session
`new -As sessname`	attach to a session *sessname*, create it if necessary.
`list-sessions`	list sessions
`ls`	list sessions
`switch-client -t sessname`	switch to session *sessname*
`kill-session -a`	kill all sessions
`kill-session -t sessname`	kill *sessname* session
`kill-session -at sessname`	kill all sessions but *sessname*

23.2.2 Execute a shell command in a new window

You can create a new window with the command

```
:new-window [-d] [-n window-name] <shell-command>
```

- With `-d` the current window in unchanged.
- `neww` is an alias to `new window`.

Examples:

```
$ tmux neww 'vi ~/.tmux.conf'
$ tmux neww -d 'rsync -avz ~/documents ssh:example.org'
```

23.2.3 Synchronize panes

You can synchronize panes, i.e. send the keyboard to multiple panes with the command:

```
:setw synchronize-panes
```

Toggle it off again by repeating the command.

23.2.4 Execute a shell command in a pane

You can create a new pane with the command:

```
:split-window [-dhv] [p percentage] <shell-command>
```

The flag `-h` means horizontal split, and `-v` stand for vertical split; if you add `d` the new pane will not get the focus. `splitw` is an alias for `split-window`. *The name vertical split is somewhat confusing, it means that your panes will be separated by an horizontal line, and so are stacked vertically.*

It is usefull to launch a long running command in forground. Example:

```
$ tmux splitw -dh htop
$ tmux splitw -v -p 90 man tmux
```

These two command can also be entered at tmux command prompt with:

```
C-b:splitw -dh htop
C-b:splitw -v -p 90 man tmux
```

If you use often such commands, an alias makes it easier:

```
alias thspl "tmux splitw -dh'
alias tvspl "tmux splitw -v"
alias tmman "tmux splitw -v -p 90 man"
```

with the `command-alias` command you can also create the aliases inside the session.

```
$ tmux set-option -s command-alias[10] vspl='split-window -v'
```

and you can do either in command line

```
$ tmux vspl
```

or at command prompt

```
vspl
```

To know which cells of the array `command-alias` are used, and their content do:

```
$ tmux show-options -s command-alias
```

23.2.5 Move a window to a pane

When you want to bring an other window as pane in the current window you can use the command:

```
:joinp -s :2
```

Or you can prefer to send your window inside another one as new pane:

```
:joinp -t :1
```

The post join window to pane propose to add to tmux.conf

```
# pane movement
bind-key J command-prompt -p "join pane from:"  "join-pane -s '%%'"
bind-key S command-prompt -p "send pane to:"  "join-pane -t '%%'"
```

or

```
bind-key J choose-tree -w 'join-pane -h -s "%%"'
bind-key S choose-tree -w 'join-pane -t "%%"'
```

23.3 Mouse support

If you set the mouse option, mouse events can be bound to keys. The default is to use the mouse to select and resize panes, to copy text and to change window using the status line.

You turn on the mouse with the command *for tmux 2.1 and above*:

```
setw -g mouse on
```

23.4 Configurations Options:

```
# Mouse support - set to on if you want to use the mouse
setw -g mouse on

# split panes using | and -
bind | split-window -h
bind - split-window -v
unbind '"'
unbind %

# reload config file
# will hide the default refresh-client binding
bind r source-file /path/to/tmux.conf

# Set the default terminal mode to 256color mode
set -g default-terminalq "screen-256color"

# enable activity alerts
setw -g monitor-activity on
set -g visual-activity on

# Center the window list
set -g status-justify centre
```

23.5 References

- tmux manual: tmux(1)
- GitHub - tmux
- tmux FAQ
- ArchWiki: Tmux is a good introduction with references to complementary articles.
- The Tao of Tmux an online book
- Awesome tmux a list of helpful tmux links for various tutorials, plugins, and configuration settings.

CHAPTER

TWENTYFOUR

VI(M) REFERENCE CARD

24.1 Basic movement

`h l k j`	character left, right; line up, down
`b w`	word/token left, right
`ge e`	end of word/token left, right
`{ }`	beginning of previous, next paragraph
`( )`	beginning of previous, next sentence
`0 gm`	beginning, middle of line
`^ $`	first, last character of line
`nG ngg`	line *n*, default the last, first
`n%`	percentage *n* of the file *(*n must be provided)*
`n|`	column *n* of current line
`%`	match of next brace, bracket, comment, `#define`
`nH nL`	line *n* from start, bottom of window
`M`	middle line of window

24.2 Insertion & replace → insert mode

`i a`	insert before, after cursor
`I A`	insert at beginning, end of line
`gI`	insert text in first column
`o O`	open a new line below, above the current line
`rc`	replace character under cursor with *c*
`grc`	like ‘‘ r‘‘, but without affecting layout
`R`	replace characters starting at the cursor
`gR`	like ‘‘ R‘‘, but without affecting layout
`cm`	change text of movement command *m*
`cc` *or* `S`	change current line
`C`	change to the end of line
`s`	change one character and insert
`~`	switch case and advance cursor
`g~m`	switch case of movement command *m*
`gum gUm`	lowercase, uppercase text of movement *m*
`<m >m`	shift left, right text of movement *m*
`n<< n>>`	shift *n* lines left, right

24.3 Deletion

`x X`	delete character under, before cursor
`dm`	delete text of movement command *m*
`dd D`	delete current line, to the end of line
`J gJ`	join current line with next, without space
`:rd`	delete range *r* lines
`:rdx`	delete range *r* lines into register *x*

24.4 Insert mode

`^Vc ^Vn`	insert char *c* literally, decimal value *n*
`^A`	insert previously inserted text
`^@`	same as `^A` and stop insert → command mode
`^Rx ^R^Rx`	insert content of register *x*, literally
`^N ^P`	text completion before, after cursor
`^W`	delete word before cursor
`^U`	delete all inserted character in current line
`^D ^T`	shift left, right one shift width
`^Kc1c2` *or* `c1←c2`	enter digraph *\c1,c2*
`^Oc`	execute *c* in temporary command mode
`^X^E ^X^Y`	scroll up, down
`<esc>` *or* `^[`	abandon edition → command mode

24.5 Copying

`"x`	use register *x* for next delete, yank, put
`:reg`	show the content of all registers
`:reg x`	show the content of registers *x*
`ym`	yank the text of movement command *m*
`yy` *or* `Y`	yank current line into register
`p P`	put register after, before cursor position
`]p [p`	like ‘‘ p‘‘, ‘‘ P‘‘ with indent adjusted
`gp gP`	like ‘‘ p‘‘, ‘‘ P‘‘ leaving cursor after new text

24.6 Advanced insertion

`g?m`	perform rot13 encoding on movement *m*
`n^A n^X`	*+n*, *-n* to number under cursor
`gqm`	format lines of movement *m* to fixed width
`:rce w`	center lines in range *r* to width *w*
`:rle i`	left align lines in range *r* with indent *i*
`:rri w`	right align lines in range *r* to width *w*
`!mc`	filter lines of movement *m* through command *c*
`n!!c`	filter *n* lines through command *c*
`:r!c`	filter range *r* lines through command *c*

24.7 Visual mode

`v V ^V`	start/stop highlighting characters, lines, block
`o`	exchange cursor position with start of highlighting
`gv`	start highlighting on previous visual area
`aw as ap`	select a word, a sentence, a paragraph
`ab aB`	select a block ( ), a block `{ }`

24.8 Undoing, repeating & registers

`u U`	undo last command, restore last changed line
`. ^R`	repeat last changes, redo last undo
`n.`	repeat last changes with count replaced by *n*
`qc qC`	record, append typed characters in register *c*
`q`	stop recording
`@c`	execute the content of register *c*
`@@`	repeat previous ‘‘ @‘‘ command
`:@c`	execute register *c* as an *Ex* command
`:rg/p/c`	execute *Ex* command *c* on range *r*

where pattern *p* matches

24.9 Complex movement

`- +`	line up, down on first non-blank character
`B W`	space-separated word left, right
`gE E`	end of space-separated word left, right
`n_`	down *n-1* line on first non-blank character
`g0`	beginning of *screen* line
`g^ g$`	first, last character of *screen* line
`gk gj`	*screen* line up, down
`fc Fc`	next, previous occurence of character *c*
`tc Tc`	before next, previous occurence of *c*
`; ,`	repeat last ‘‘ fFtT‘‘, in opposite direction
`[[ ]]`	start of section backward, forward
`[] ][`	end of section backward, forward
`[( ])`	unclosed (,) backward, forward
`[{ ]}`	unclosed `{`, `}` backward, forward
`[m ]m`	start of backward, forward *Java* method
`[# ]#`	unclosed `#if`, `#else`, `#endif` backward, forward
`[* ]*`	start, end of `/* */` backward, forward

24.10 Search & substitution

`/s ?s`	search forward, backward for *s*
`/s/o ?s?o`	search fwd, bwd for *s* with offset *o*
`n` *or* `/`	repeat forward last search
`N` *or* `?`	repeat backward last search
`# *`	search backward, forward for word under cursor
`g# g*`	same, but also find partial matches
`gd gD`	local, global definition of symbol under cursor
`:rs/f/t/x`	substitute *f* by *t* in range *r*
	x: ‘‘ g‘‘-all occurrences, ‘‘ c‘‘-confirm changes
`:rs x`	repeat substitution with new *r* & *x*

24.11 Special characters in search patterns

`.^ $`	any single character, start, end of line
`\< \>`	start, end of word
`[c1-c2]`	a single character in range *c1..c2*
`[^c1-c2]`	a single character not in range
`\i \k \I \K`	an identifier, keyword; excl. digits
`\f \p \F \P`	a file name, printable char.; excl. digits
`\s \S`	a white space, a non-white space
`\e \t \r \b`	<esc>, <tab>, <>, <←>
`\= * \+`	match *0..1*, *0..∞*, *1..∞* of preceding atoms
`|`	separate two branches (* * *or*)
`\( \)`	group patterns into an atom
`\& \n`	the whole matched pattern, *nth* `()` group
`\u \l`	next character made upper, lowercase
`\c \C`	ignore, match case on next pattern

24.12 Offsets in search commands

`n` *or* `+n`	*n* line downward in column 1
`-n`	*n* line upward in column 1
`e+n e-n`	*n* characters right, left to end of match
`s+n s-n`	*n* characters right, left to start of match
`;sc`	execute search command *sc* next

24.13 Marks and motions

`mc`	mark current position with mark *c [a..Z]*
`` `c `C ``	go to mark *c* in current, *C* in any file
`` `0..9 ``	go to last exit position
`` ``` `" ```	go to position before jump, at last edit
`` `[ `] ``	go to start, end of previously operated text
`:marks`	print the active marks list
`:jumps`	print the jump list
`n^O`	go to *nth* older position in jump list
`n^I`	go to *nth* newer position in jump list

24.14 Key mapping & abbreviations

`:map c e`	map *c e* in normal & visual mode
`:map! c e`	map *c e* in insert & cmd-line mode
`:unmap c :unmap! c`	remove mapping *c*
`:mk f`	write current mappings, settings... to file *f*
`:ab c e`	add abbreviation for *c e*
`:ab c`	show abbreviations starting with *c*
`:una c`	remove abbreviation *c*

24.15 Tags

`:ta t`	jump to tag *t*
`:nta`	jump to *nth* newer tag in list
`^] ^T`	jump to the tag under cursor, return from tag
`:ts t`	list matching tags and select one for jump
`:tj t`	jump to tag or select one if multiple matches
`:tags`	print tag list
`:npo :n^T`	jump back from, to *nth* older tag
`:tl`	jump to last matching tag
`^W} :pt t`	preview tag under cursor, tag *t*
`^W]`	split window and show tag under cursor
`^Wz` *or* `:pc`	close tag preview window

24.16 Scrolling & multi-windowing

^E ^Y	scroll line up, down
^D ^U	scroll half a page up, down
^F ^B	scroll page up, down
zt *or* z	set current line at top of window
zz *or* z.	set current line at center of window
zb *or* z-	set current line at bottom of window
zh zl	scroll one character to the right, left
zH zL	scroll half a screen to the right, left
^Ws *or* :split	split window in two
^Wn *or* :new	create new empty window
^Wo *or* :on	make current window one on screen
^Wj ^Wk	move to window below, above
^Ww ^W^W	move to window below, above (wrap)

24.17 Ex commands ()

:e f	edit file *f*, unless changes have been made
:e! f	edit file *f* always (by default reload current)
:wn :wN	write file and edit next, previous one
:n :N	edit next, previous file in list
:rw	write range *r* to current file
:rw f	write range *r* to file *f*
:rw>>f	append range *r* to file *f*
:q :q!	quit and confirm, quit and discard changes
:wq *or* :x *or* ZZ	write to current file and exit
<up> <down>	recall commands starting with current
:r f	insert content of file *f* below cursor
:r! c	insert output of command *c* below cursor
:args	display the argument list
:rco a :rm a	copy, move range *r* below line *a*

24.18 Ex ranges

zfm	create fold of movement *m*
:rfo	create fold for range *r*
zd zE	delete fold at cursor, all in window
zo zc zO zC	open, close one fold; recursively
[z]z	move to start, end of current open fold
zj zk	move down, up to start, end of next fold

24.19 Miscellaneous

`:sh :!c`	start shell, execute command *c* in shell
`K`	lookup keyword under cursor with ‘‘ man‘‘
`:make`	start ‘‘ make‘‘, read errors and jump to first
`:cn :cp`	display the next, previous error
`:cl :cf`	list all errors, read errors from file
`^L ^G`	redraw screen, show filename and position
`g^G`	show cursor column, line, and character position
`ga`	show ASCII value of character under cursor
`gf`	open file which filename is under cursor
`:redir>f`	redirect output to file *f*
`:mkview [f]`	save view configuration [to file *f*]
`:loadview [f]`	load view configuration [from file *f*]
`^@ ^K ^_ \ Fn ^Fn`	unmapped keys

This reference card is taken from Laurent Grégoire VIM Quick Reference Card *(the original is better formatted than the present derivative)* it exists also in pdf and translated in many format and languages.

There is also a fork (pdf) from Michael Goerz.

CHAPTER

TWENTYFIVE

NANO SHORTCUTS REFERENCE

25.1 Movements

M-	M-x	Go to the first line of the file
M-/	M-?	Go to the last line of the file
^_	F13 M-G	Go to line and column number
^F		Go forward one character
^B		Go back one character
^Space		Go forward one word
M-Space		Go back one word
^P		Go to previous line
^N		Go to next line
^A		Go to beginning of current line
^E		Go to end of current line
M-(	M-9	Go to beginning of paragraph; then of previous paragraph
M-)	M-0	Go just beyond end of paragraph; then of next paragraph
M-]		Go to the matching bracket
^Y	F7	Go to previous screen
^V	F8	Go to next screen
M–	M-_	Scroll up one line without scrolling the cursor
M-+	M-=	Scroll down one line without scrolling the cursor
^C	F11	Display the position of the cursor
M-D		Count the number of words, lines, and characters
^G	F1	Display help screen
^L		Refresh (redraw) the current screen

25.2 Insertion/deletion

M-V		Insert the next keystroke verbatim
^I		Insert a tab at the cursor position
^M		Insert a newline at the cursor position
^D		Delete the character under the cursor
^H		Delete the character to the left of the cursor
M-T		Cut from the cursor position to the end of the file
^K	F9	Cut the current line and store it in the cutbuffer
^U	F10	Uncut from the cutbuffer into the current line
M-^	M-6	Copy the current line and store it in the cutbuffer
^J	F4	Justify the current paragraph
M-J		Justify the entire file
M-}		Indent the current line
M-{		Unindent the current line

25.3 Files/Buffers

^X	F2	Close the current file buffer / Exit from nano
^O	F3	Write the current file to disk
^0 ^T		choose a file to write the buffer with the file browser.
^R	F5	Insert another file into the current buffer
^R ^T		insert with file browser
M-<	M-,	Switch to the previous file buffer
M->	M-.	Switch to the next file buffer
^Z		Suspend the editor

25.4 Search

^W	F6	Search for a string or a regular expression
^T	F12	Invoke the spell checker, if available
^	F14 M-R	Replace a string or a regular expression
M-W	F16	Repeat last search
^^	F15 M-A	Mark text at the cursor position

25.5 Toggles

M-X	Help mode enable/disable
M-C	Constant cursor position display enable/disable
M-O	Use of one more line for editing enable/disable
M-C	Constant cursor position display enable/disable
M-O	Use of one more line for editing enable/disable
M-S	Smooth scrolling enable/disable
M-P	Whitespace display enable/disable
M-Y	Color syntax highlighting enable/disable
M-H	Smart home key enable/disable
M-I	Auto indent enable/disable
M-K	Cut to end enable/disable
M-L	Long line wrapping enable/disable
M-Q	Conversion of typed tabs to spaces enable/disable
M-B	Backup files enable/disable
M-F	Multiple file buffers enable/disable
M-M	Mouse support enable/disable
M-N	No conversion from DOS/Mac format enable/disable
M-Z	Suspension enable/disable
M-$	Soft line wrapping enable/disable

Nano reference is also available as an *info* manual.

CHAPTER

TWENTYSIX

PRINTING WITH CUPS

The reference is cups command-line printing and options on the cups.org site or on your cups:*local server <options.html>*.

Refs: lpoptions(1) (local), lpinfo(1) (local), lpstat(1) (local), lp(1) (local), lpadmin(8) (local), cancel(1) (local), lpmove(8) (local), cupsenable(8) (local)

26.1 Knowing about your printers

- List the devices

```
$ lpinfo -v
```

- List the drivers:

```
$ lpinfo -m
```

- To know the available options:

```
$ lpoptions -d foo -l
```

26.2 Printing

- Get the lists of available sizes and use paper size of A3 and fit to page, and print A3 page

```
$ lpoptions -p foo -l| grep 'PageSize'
$ lp -d foo -o fitplot:PageSize=A3 /etc/motd
```

- get the lists of slots and use a separate Input slot for first page:

```
$ lpoptions -p foo -l|grep -i slot
$ lp -d foo -o 1:InputSlot=UpperCassette -o InputSlot=LowerCassette
```

- watermarks ("Draft" . . .) on even pages, gray color mode on odd:

```
$ lp -d foo -o even:Watermark=on -o odd:ColorMode=Gray file
```

- Print some pages, and page ranges:

```
$ lp -d foo1 -o 1,6-10,15,20- file
```

- Print multiple copies from a pipe output:

```
$ program | lp -d foo -n 8
```

- idem with collated copies

```
$ program | lp -d foo -n 8 -o Collate=true
```

- Print landscape, duplex with short side tumble:

```
$ lp -d foo  -o sides=two-sided-short-edge:landscape file
```

- Print with a custom media size (example 624pts width, 312pts length)

```
$ lp -d foo  -o media=Custom.624x312 file
```

  You can also use a predefined media size: Letter Legal A4 A5 A6 A7 A8 B5 B6 B7 B8 C5 C6 DL C7 C8 Custom.WIDTHxHEIGHT . WIDTHxHEIGHT is in point or is suffixed with in, cm, mm

- print one side (even with default duplex) and a banner (*standard*: without label, *classified,unclassified, secret, topsecret*: with corresponding label)

```
$ lp -o sides=one-sided:job-sheets=standard
```

- Print 4-up (or 1,2,4,16) with double border (or single, single-thick, double, double-thick), Bottom to top, left to right (btlr or btrl or lrbt or rlbt or rltb or tblr or tbrl):

```
$ lp -o number-up=4:page-border=double:number-up-layout=btlr file
```

- **pretty**printing 2 columns (or 3, 4, . . .) with 12 char/inch (default 10) and 8 lines/inch (default 6):

```
$ lp -o prettyprint:columns=2:cpi=12:lpi=8 file
```

- The borders can be forced by `page-{left,right,bottom,top}`

26.3 Managing printers

Choosing a printer:

```
$ lpstat -p -d
```

choosing a printer on another server

```
$ lpstat -h server -p -d
```

setting the default printer:

```
$ lpoptions -d printer
```

The user default printer and options are written in `~/.cups/lpoptions`, which override system wide options in `/etc/cups/lpoptions`

Printer queue state:

```
$ lpstat -p foo
```

All printer states:

```
$ lpstat -p
```

Printer devices summary

```
$ lpstat -s
```

Physical devices connected to all printers

```
lpstat -v
```

All the status and defaults of printer *foo*

```
$ lpstat -l -p foo
```

List all jobs on printer *foo*

```
$ lpstat -o foo
$ lpq -P foo
```

Canceling job 12345 on printer *foo*:

```
$ cancel 12345 foo
```

Canceling all jobs from printer *foo*:

```
$ cancel -a foo
```

move job 123 to printer *bar*:

```
$ lpmove 123 bar
```

Define the *device-uri* and *ppd file* for the windows printer *foo* shared thru samba

```
$ lpadmin -p foo -v \
smb://workgroup/user:mypass@windows-server/inkjet \
-P /root/inkjet.ppd
```

Enable and accepts jobs on foo:

```
$ lpadmin -p foo -E
```

or:

```
$ cupsenable foo
```

Disable, i.e stop the *foo* printer:

```
$ cupsdiable foo
```

CHAPTER

TWENTYSEVEN

INDICES AND TABLES

- genindex
- modindex
- search