LINUX TOOLBOX

This document is a collection of Linux commands and tasks which are useful for IT work or for advanced users. This also serves as a essential reference for Devops Engineers. This is a practical guide with concise explanations, however the reader is supposed to know what s/he is doing. Visit www.schoolofdevops.com to get your devops journey started.

1 SYSTEM

Running kernel and system information

```
# uname -a                           # Get the kernel version (and BSD version)
# lsb_release -a                     # Full release info of any LSB distribution
# cat /etc/issue                     # Get OS distribution and  version
# cat /etc/debian_version            # Get Debian version
```

Use /etc/DISTR-release with DISTR= lsb (Ubuntu), redhat, gentoo, mandrake, sun (Solaris), and so on. See also `/etc/issue`.

```
# uptime                             # Show how long the system has been running + load
# hostname                           # system's host name
# hostname -i                        # Display the IP address of the host. (Linux only)
# man hier                           # Description of the file system hierarchy
# last reboot                        # Show system reboot history
```

1.1 Hardware Informations

Kernel detected hardware

```
# dmesg                              # Detected hardware and boot messages
# lsdev                              # information about installed hardware
# dd if=/dev/mem bs=1k skip=768 count=256 2>/dev/null | strings -n 8 # Read BIOS
```

Linux

```
# cat /proc/cpuinfo                  # CPU model
# cat /proc/meminfo                  # Hardware memory
# grep MemTotal /proc/meminfo        # Display the physical memory
# watch -n1 'cat /proc/interrupts'   # Watch changeable interrupts continuously
# free -m                            # Used and free memory (-m for MB)
# cat /proc/devices                  # Configured devices
# lspci -tv                          # Show PCI devices
# lsusb -tv                          # Show USB devices
# dmidecode                          # Show DMI/SMBIOS: hw info from the BIOS
```

1.2 Load, statistics and messages

The following commands are useful to find out what is going on on the system.

```
# top                                # display and update the top cpu processes
# mpstat 1                           # display processors related statistics
# vmstat 2                           # display virtual memory statistics
# iostat 2                           # display I/O statistics (2 s intervals)
# ipcs -a                            # information on System V interprocess
# tail -n 500 /var/log/syslog      # Last 500 kernel/syslog messages
```

1.3 Users

```
# id                                 # Show the active user id with login and group
# last                               # Show last logins on the system
# who                                # Show who is logged on the system
# groupadd devops                     # Add group "devops" and user abeer
# useradd -c "Abeer Shah" -g devops -m abeer
# useradd -c "Kiaan Shah"  -m kiaan
# usermod -a -G <group> <user>       # Add existing user to group (Debian)
# usermod -a -G devops kiaan
# groupmod -A <user> <group>         # Add existing user to group (SuSE)
# userdel abeer kiaan                      # Delete users abeer and kiaan (Linux/Solaris)
```

Following files contain user information

```
# /etc/passwd                  # User information
# /etc/shadow                       # User Passwords
# /etc/group                        # User Groups on the system
```

Encrypted passwords are stored in /etc/shadow for Linux .

To temporarily prevent logins system wide (for all users but root) use nologin. The message in nologin will be displayed (might not work with ssh pre-shared keys).

```
# echo "Sorry no login now" > /etc/nologin       # (Linux)
```

1.4 Limits

Some application require higher limits on open files and sockets (like a proxy web server, database). The default limits are usually too low.

Linux

Per shell/script

The shell limits are governed by `ulimit`. The status is checked with `ulimit  -a`. For example to change the open files limit from 1024 to 10240 do:

```
# ulimit -n 10240                    # This is only valid within the shell
```

The `ulimit` command can be used in a script to change the limits for the script only.

Per user/process

Login users and applications can be configured in `/etc/security/limits.conf`. For example:

```
# cat /etc/security/limits.conf
*   hard    nproc   250              # Limit user processes
asterisk hard nofile 409600          # Limit application open files
```

System wide

Kernel limits are set with sysctl. Permanent limits are set in `/etc/sysctl.conf`.

```
# sysctl -a                          # View all system limits
# sysctl fs.file-max                 # View max open files limit
# sysctl fs.file-max=102400          # Change max open files limit
# echo "1024 50000" > /proc/sys/net/ipv4/ip_local_port_range  # port range
# cat /etc/sysctl.conf
fs.file-max=102400                   # Permanent entry in sysctl.conf
# cat /proc/sys/fs/file-nr           # How many file descriptors are in use
```

1.5 Runlevels (systemd)

Linux

Once booted, the kernel starts `systemd` which then starts `rc` which starts all scripts belonging to a runlevel. The scripts are stored in /etc/init.d and are linked into /etc/rc.d/rcN.d with N the runlevel number.
The default runlevel is configured in /etc/inittab. It is usually 3 or 5:

```
# systemctl get-default
graphical.target
```

The actual runlevel can be changed with `systemd`. For example to go from 3 to 5:

```
# systemctl set-default multi-user.target       # Configured default runlevel to 3
```

```
# systemctl isolate multi-user.target           # Changes to  runlevel 3
```

```
# telinit 3                                     # Changes to  runlevel 3 (alternate command)
```

```
# systemctl isolate graphical.target            # Changes to  runlevel 5
```

0	Shutdown and halt	poweroff.target
1	Single-User mode (also S)	rescue.target
2	Multi-user without network	multi-user.target
3	Multi-user with network	graphical.target
5	Multi-user with X	reboot.target
6	Reboot	emergency.target

Use `chkconfig` to configure the programs that will be started at boot in a runlevel.

```
# systemctl list-unit-files                   # List all services
# systemctl disable ssh                # Disable sshd
# systemctl enable ssh      # Enable sshd
# systemctl status ssh      # Check ths status of sshd service
# systemctl daemon-reload     # Reload systemctl after any changes
```

1.6 Reset root password

Linux method 1

At the boot loader (lilo or grub), enter the following boot option:

```
init=/bin/sh
```

The kernel will mount the root partition and `init` will start the bourne shell instead of `rc` and then a runlevel. Use the command `passwd` at the prompt to change the password and then reboot. Forget the single user mode as you need the password for that.
If, after booting, the root partition is mounted read only, remount it rw:

```
# mount -o remount,rw /
# passwd                            # or delete the root password (/etc/shadow)
# sync; mount -o remount,ro /       # sync before to remount read only
# reboot
```

Linux method 2

The solution is to mount the root partition from an other OS (like a rescue CD/ another cloud server), mount the volume/root disk and change the password on it.

- Boot a live CD or installation CD into a rescue mode which will give you a shell if you have access to physical host
- Find the root partition with fdisk e.g. fdisk /dev/sda
- Mount it and use chroot:
- On cloud, launch a new instance, detach root parition from earlier instance, attach it to new one

```
# mount -o rw /dev/ad4s3a /mnt
# chroot /mnt                       # chroot into /mnt
# passwd
# reboot
```

1.7 Kernel modules

Linux

```
# lsmod                             # List all modules loaded in the kernel
# modprobe isdn                     # To load a module (here isdn)
```

2 PROCESSES

Listing (p4) | Priority (p5) | Background/Foreground (p5) | Top (p6) | Kill (p6)

2.1 Listing and PIDs

Each process has a unique number, the PID. A list of all running process is retrieved with ps.

```
# ps -aux                          # Extensive list of all running process
```

```
# ps -ef --forest                  # Extensive list of all running process in tree format
```

However more typical usage is with a pipe or with pgrep (for OS X install proctools from MacPorts (page 0)):

```
# ps axww | grep cron
  586  ??  Is     0:01.48 /usr/sbin/cron -s
# ps axjf                            # All processes in a tree format (Linux)
# ps aux | grep 'ss[h]'              # Find all ssh pids without the grep pid
# pgrep -l sshd                      # Find the PIDs of processes by (part of) name
# echo $$                            # The PID of your shell
# fuser -va 22/tcp                   # List processes using port 22 (Linux)
# pmap PID                           # Memory map of process (hunt memory leaks) (Linux)
# fuser -va /home                    # List processes accessing the /home partition
# strace df                          # Trace system calls and signals
```

2.2 Priority

Change the priority of a running process with renice. **Negative numbers have a higher priority**, the lowest is -20 and "nice" have a positive value.

```
# renice -5 586                      # Stronger priority
586: old priority 0, new priority -5
```

Start the process with a defined priority with nice. Positive is "nice" or weak, negative is strong scheduling priority. Make sure you know if /usr/bin/nice or the shell built-in is used (check with # which nice).

```
# nice -n -5 top                     # Stronger priority (/usr/bin/nice)
# nice -n 5 top                      # Weaker priority (/usr/bin/nice)
# nice +5 top                        # tcsh builtin nice (same as above!)
```

While nice changes the CPU scheduler, an other useful command ionice will schedule the disk IO. This is very useful for intensive IO application (e.g. compiling). You can select a class (idle - best effort - real time), the man page is short and well explained.

```
# ionice c3 -p123                    # set idle class for pid 123 (Linux only)
# ionice -c2 -n0 firefox             # Run firefox with best effort and high priority
# ionice -c3 -p$$                    # Set the actual shell to idle priority
```

The last command is very useful to compile (or debug) a large project. Every command launched from this shell will have a lover priority. $$ is your shell pid (try echo $$).

2.3 Background/Foreground

When started from a shell, processes can be brought in the background and back to the foreground with [Ctrl]-[Z] (^Z), bg and fg. List the processes with jobs. When needed detach from the terminal with disown.

```
# ping google.com > ping.log
^Z                                   # ping is suspended (stopped) with [Ctrl]-[Z]
# bg                                 # put in background and continues running
# jobs -l                            # List processes in background
[1]  - 36232 Running                       ping cb.vu > ping.log
[2]  + 36233 Suspended (tty output)        top
# fg %2                              # Bring process 2 back in foreground
```

```
# make                               # start a long compile job but need to leave the terminal
^Z                                   # suspended (stopped) with [Ctrl]-[Z]
# bg                                 # put in background and continues running
# disown -h %1                       # detatch process from terminal, won't be killed at logout
```

No straight forward way to re-attach the process to a new terminal, try reptyr (Linux).
Use nohup to start a process which has to keep running when the shell is closed (immune to hangups).

```
# nohup ping -i 60 > ping.log &
```

2.4 Top

The program top displays running information of processes. See also the program htop from htop.sourceforge.net (a more powerful version of top) which runs on Linux and FreeBSD (ports/sysutils/htop/). While top is running press the key h for a help overview. Useful keys are:

- **u [user name]** To display only the processes belonging to the user. Use + or blank to see all users
- **k [pid]** Kill the process with pid.
- **1** To display all processors statistics (Linux only)
- **R** Toggle normal/reverse sort.

Important process states from TOP output:

R	running
S	sleeping
D	disk Sleep (uninturrupted)
T	stopped
t	tracing stop
X	dead
Z	zombie
P	parked
I	idle (kernel threads)

Some useful alternates to top.

```
# htop                                 # enhanced version of top
# glances                             # another alternative with useful system info
```

2.5 Signals/Kill

Terminate or send a signal with kill or killall.

```
# ping -i 60 google.com > ping.log &
[1] 4712
# kill -s TERM 4712                   # same as kill -15 4712
# killall -1 httpd                    # Kill HUP processes by exact name
# pkill -9 http                       # Kill KILL processes by (part of) name
# pkill -TERM -u www                  # Kill TERM processes owned by www
# fuser -k -TERM -m /home             # Kill every process accessing /home (to umount)
```

Important signals are:

1	HUP (hang up)
2	INT (interrupt)
3	QUIT (quit)
9	KILL (non-catchable, non-ignorable kill)
15	TERM (software termination signal)

3 FILE SYSTEM

3.1 Permissions

Change permission and ownership with chmod and chown. The default umask can be changed for all users in /etc/profile for Linux. The default umask is usually 022. The umask is subtracted from 777, thus umask 022 results in a permission 0f 755.

```
1 --x execute                         # Mode 764 = exec/read/write | read/write | read
2 -w- write                           # For:       |--  Owner  --|   |- Group-|   |Oth|
4 r-- read
```

```
  ugo=a                              u=user, g=group, o=others, a=everyone
```

```
# chmod [OPTION] MODE[,MODE] FILE    # MODE is of the form [ugoa]*([-+=]([rwxXst]))
# touch /tmp/file                    # Create a empty file
# ls -l /tmp/file                    # check the permissions of the file
# chmod 640 /tmp/file                # Restrict the log -rw-r-----
# chmod u=rw,g=rx,o= /tmp/file        # Same as above
# chmod -R o-r /home/*               # Recursive remove other readable for all users
# chmod u+s /path/to/prog            # Set SUID bit on executable (know what you do!)
# find / -perm -u+s -print           # Find all programs with the SUID bit
# chown user:group /path/to/file     # Change the user and group ownership of a file
# chgrp group /path/to/file          # Change the group ownership of a file
# chmod 640 `find ./ -type f -print` # Change permissions to 640 for all files
# chmod 751 `find ./ -type d -print` # Change permissions to 751 for all directories
```

3.2 Disk information

```
# hdparm -I /dev/sda                 # information about the IDE/ATA disk (Linux)
# fdisk /dev/ad2                     # Display and manipulate the partition table
# smartctl -a /dev/ad2               # Display the disk SMART info
```

3.3 System mount points/Disk usage

```
# mount | column -t                  # Show mounted file-systems on the system
# df                                 # display free disk space and mounted devices
# cat /proc/partitions               # Show all registered partitions (Linux)
```

Disk usage

```
# du -sh *                           # Directory sizes as listing
# du -csh                            # Total directory size of the current directory
# du -ks * | sort -n -r              # Sort everything by size in kilobytes
# ls -lSr                            # Show files, biggest last
```

3.4 Who has which files opened

This is useful to find out which file is blocking a partition which has to be unmounted and gives a typical error of:

```
# umount /home/
umount: unmount of /home             # umount impossible because a file is locking home
   failed: Device busy
```

Linux

Find opened files on a mount point with fuser or lsof:

```
# fuser -m /home                     # List processes accessing /home
# lsof /home
COMMAND   PID    USER    FD   TYPE DEVICE    SIZE     NODE NAME
tcsh    29029 eedcoba   cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
lsof    29140 eedcoba   cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
```

About an application:

```
ps ax | grep Xorg | awk '{print $1}'
3324
# lsof -p 3324
COMMAND   PID    USER    FD   TYPE DEVICE    SIZE    NODE NAME
Xorg     3324 root     0w   REG        8,6   56296      12492 /var/log/Xorg.0.log
```

About a single file:

```
# lsof /var/log/Xorg.0.log
COMMAND  PID USER   FD   TYPE DEVICE  SIZE  NODE NAME
Xorg    3324 root    0w   REG    8,6 56296 12492 /var/log/Xorg.0.log
```

3.5 Mount/remount a file system

For example the cdrom. If listed in /etc/fstab:

```
# mount /cdrom
```

Or find the device in /dev/ or with dmesg

Linux

```
# mount -t auto /dev/cdrom /mnt/cdrom   # typical cdrom mount command
# mount /dev/hdc -t iso9660 -r /cdrom   # typical IDE
# mount /dev/scd0 -t iso9660 -r /cdrom  # typical SCSI cdrom
# mount /dev/sdc0 -t ntfs-3g /windows   # typical SCSI
```

Entry in /etc/fstab:

```
/dev/cdrom   /media/cdrom  subfs noauto,fs=cdfss,ro,procuid,nosuid,nodev,exec 0 0
```

Mount a FreeBSD partition with Linux

Find the partition number containing with fdisk, this is usually the root partition, but it could be an other BSD slice too. If the FreeBSD has many slices, they are the one not listed in the fdisk table, but visible in /dev/sda* or /dev/hda*.

```
# fdisk /dev/sda                    # Find the FreeBSD partition
/dev/sda3   *        5357        7905    20474842+  a5  FreeBSD
# mount -t ufs -o ufstype=ufs2,ro /dev/sda3 /mnt
/dev/sda10 = /tmp; /dev/sda11 /usr   # The other slices
```

Remount

Remount a device without unmounting it. Necessary for fsck for example

```
# mount -o remount,ro /             # Linux
# mount -o ro -u /                  # FreeBSD
```

Copy the raw data from a cdrom into an iso image (default 512 blocksize might cause problems):

```
# dd if=/dev/cd0c of=file.iso bs=2048
```

Virtualbox

Allow a share on the host:

```
# VBoxManage sharedfolder add "GuestName" --name "share" --hostpath "C:\hostshare"
```

Mount share on guest (linux, FreeBSD)

```
# sudo mount -t vboxsf share /home/vboxshare # -o uid=1000,gid=1000 (as appropriate)
share /home/colin/share vboxsf defaults,uid=colin 0 0 # fstab entry
```

3.6 Add swap on-the-fly

Suppose you need more swap (right now), say a 2GB file /swap2gb (Linux only).

```
# dd if=/dev/zero of=/swap2gb bs=1024k count=2000
# mkswap /swap2gb                   # create the swap area
# swapon /swap2gb                   # activate the swap. It now in use
# swapoff /swap2gb                  # when done deactivate the swap
# rm /swap2gb
```

3.7 Create a file based image

For example a partition of 1GB using the file /usr/vdisk.img. Here we use the vnode 0, but it could also be 1.

Linux

```
# dd if=/dev/zero of=/usr/vdisk.img bs=1024k count=1024
# mkfs.ext3 /usr/vdisk.img
# mount -o loop /usr/vdisk.img /mnt
# umount /mnt; rm /usr/vdisk.img                      # Cleanup
```

Linux with losetup

/dev/zero is much faster than urandom, but less secure for encryption.

```
# dd if=/dev/urandom of=/usr/vdisk.img bs=1024k count=1024
# losetup /dev/loop0 /usr/vdisk.img                    # Creates and associates /dev/loop0
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /mnt
# losetup -a                                           # Check used loops
# umount /mnt
# losetup -d /dev/loop0                                # Detach
# rm /usr/vdisk.img
```

3.8 Create a memory file system

A memory based file system is very fast for heavy IO application. How to create a 64 MB partition mounted on /memdisk:

Linux

```
# mount -t tmpfs -osize=64m tmpfs /memdisk
```

3.9 Disk performance

Read and write a 1 GB file on partition ad4s3c (/home)

```
# time dd if=/dev/ad4s3c of=/dev/null bs=1024k count=1000
# time dd if=/dev/zero bs=1024k count=1000 of=/home/1Gb.file
# hdparm -tT /dev/hda      # Linux only
```

4 NETWORK

Routing (p9) | Additional IP (p10) | Change MAC (p10) | Ports (p10) | Firewall (p10) | IP Forward (p10) | NAT (p10) | DNS (p11) | DHCP (p0) | Traffic (p12) | QoS (p0) | NIS (p0) | Netcat (p12)

4.1 Debugging (See also Traffic analysis) (page 12)

Linux

```
# ethtool eth0              # Show the ethernet status (replaces mii-diag)
# ethtool -s eth0 speed 100 duplex full # Force 100Mbit Full duplex
# ethtool -s eth0 autoneg off # Disable auto negotiation
# ethtool -p eth1           # Blink the ethernet led - very useful when supported
# ip link show              # Display all interfaces on Linux (similar to ifconfig)
# ip link set eth0 up       # Bring device up (or down). Same as "ifconfig eth0 up"
# ip addr show              # Display all IP addresses on Linux (similar to ifconfig)
# ip neigh show             # Similar to arp -a
```

4.2 Routing

Print routing table

```
# route -n                  # Linux or use "ip route"
# netstat -rn               # Linux, BSD and UNIX
# route print               # Windows
```

Add and delete a route

Linux

```
# route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.16.254
# ip route add 192.168.20.0/24 via 192.168.16.254       # same as above with ip route
# route add -net 192.168.20.0 netmask 255.255.255.0 dev eth0
# route add default gw 192.168.51.254
# ip route add default via 192.168.51.254 dev eth0      # same as above with ip route
# route delete -net 192.168.20.0 netmask 255.255.255.0
```

4.3 Configure additional IP addresses

Linux

```
# ifconfig eth0 192.168.50.254 netmask 255.255.255.0       # First IP
# ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0     # Second IP
# ip addr add 192.168.50.254/24 dev eth0                   # Equivalent ip commands
# ip link set dev eth0 up                                  # Activate eth0 network interface
# ip addr add 192.168.51.254/24 dev eth0 label eth0:1
# ip link ls dev eth0                                      # Get info on eth0
# ip addr del 1.2.3.4/32 dev eth0                          # Remove an IP
# ip addr flush dev eth0                                   # Remove all addresses
```

4.4 Change MAC address

Normally you have to bring the interface down before the change. Don't tell me why you want to change the MAC address...

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05      # Linux
```

4.5 Ports in use

Listening open ports:

```
# netstat -pan | grep LISTEN
# lsof -i                  # Linux list all Internet connections
# socklist                 # Linux display list of open sockets
# netstat -pan --udp --tcp | grep LISTEN        # Linux
# netstat -tup             # List active connections to/from system (Linux)
# netstat -tupl            # List listening ports from system (Linux)
# netstat -ano             # Windows
```

4.6 Firewall

Check if a firewall is running (typical configuration only):

Linux

```
# iptables  -nvL                  # For status
Open the iptables firewall
# iptables -P INPUT       ACCEPT     # Open everything
# iptables -P FORWARD     ACCEPT
# iptables -P OUTPUT      ACCEPT
# iptables -Z                        # Zero the packet and byte counters in all chains
# iptables -F                        # Flush all chains
# iptables -X                        # Delete all chains
```

4.7 IP Forward for routing

Linux

Check and then enable IP forward with:

```
# cat /proc/sys/net/ipv4/ip_forward  # Check IP forward 0=off, 1=on
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

or edit /etc/sysctl.conf with:

```
net.ipv4.ip_forward = 1
```

4.8 NAT Network Address Translation

Linux

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE        # to activate NAT
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT \
--to 192.168.16.44:22           # Port forward 20022 to internal IP port ssh
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 993:995 -j DNAT \
--to 192.168.16.254:993-995     # Port forward of range 993-995
# ip route flush cache
# iptables -L -t nat            # Check NAT status
```

Delete the port forward with -D instead of -A. The program netstat-nat[1] is very useful to track connections (it uses `/proc/net/ip_conntrack` or `/proc/net/nf_conntrack`).

```
# netstat-nat -n                # show all connections with IPs
```

4.9 DNS

On Unix the DNS entries are valid for all interfaces and are stored in /etc/resolv.conf. The domain to which the host belongs is also stored in this file. A minimal configuration is:

```
nameserver 78.31.70.238
search sleepyowl.net intern.lab
domain sleepyowl.net
```

Check the system domain name with:

```
# hostname -d                        # Same as dnsdomainname
```

Windows

On Windows the DNS are configured per interface. To display the configured DNS and to flush the DNS cache use:

```
# ipconfig /?                        # Display help
# ipconfig /all                      # See all information including DNS
```

Flush DNS

Flush the OS DNS cache, some application using their own cache (e.g. Firefox) and will be unaffected.

```
# /etc/init.d/nscd restart           # Restart nscd if used - Linux/BSD/Solaris
# lookupd -flushcache                # OS X Tiger
# dscacheutil -flushcache            # OS X Leopard and newer
# ipconfig /flushdns                 # Windows
```

Forward queries

Dig is you friend to test the DNS settings. For example the public DNS server `8.8.8.8` from google can be used for testing. See from which server the client receives the answer (simplified answer).

```
# dig sleepyowl.net
sleepyowl.net.          600     IN      A       78.31.70.238
;; SERVER: 192.168.51.254#53(192.168.51.254)
```

The router 192.168.51.254 answered and the response is the A entry. Any entry can be queried and the DNS server can be selected with @:

```
# dig MX google.com
# dig @127.0.0.1 NS docker.com          # To test the local server
# dig @8.8.8.8 NS MX kubernetes.io  # Query an external server
# dig -t CNAME hub.docker.com       # Query a CNAME entry
# dig AXFR @ns1.xname.org cb.vu       # Get the full zone (zone transfer)
```

The program host is also powerful.

```
# host -t MX cb.vu                   # Get the mail MX entry
# host -t NS -T sun.com              # Get the NS record over a TCP connection
# host -a sleepyowl.net              # Get everything
```

Reverse queries

Find the name belonging to an IP address (in-addr.arpa.). This can be done with `dig`, `host` and `nslookup`:

```
# dig -x 78.31.70.238
```

1. http://tweegy.nl/projects/netstat-nat

```
# host 78.31.70.238
# nslookup 78.31.70.238
```

/etc/hosts

Single hosts can be configured in the file /etc/hosts instead of running named locally to resolve the hostname queries. The format is simple, for example:

```
78.31.70.238   sleepyowl.net   sleepyowl
```

The priority between hosts and a dns query, that is the name resolution order, can be configured in `/etc/nsswitch.conf` AND /etc/host.conf. The file also exists on Windows, it is usually in:

```
C:\WINDOWS\SYSTEM32\DRIVERS\ETC
```

4.10 Traffic analysis

Bmon[2] is a small console bandwidth monitor and can display the flow on different interfaces.

Sniff with tcpdump

```
# tcpdump -nl -i bge0 not port ssh and src \(192.168.16.121 or 192.168.16.54\)
# tcpdump -n -i eth1 net 192.168.16.121           # select to/from a single IP
# tcpdump -n -i eth1 net 192.168.16.0/24          # select traffic to/from a network
# tcpdump -l > dump && tail -f dump               # Buffered output
# tcpdump -i rl0 -w traffic.rl0                   # Write traffic headers in binary file
# tcpdump -i rl0 -s 0 -w traffic.rl0              # Write traffic + payload in binary file
# tcpdump -r traffic.rl0                          # Read from file (also for ethereal
# tcpdump port 80                                 # The two classic commands
# tcpdump host google.com
# tcpdump -i eth0 -X port \(110 or 143\)          # Check if pop or imap is secure
# tcpdump -n -i eth0 icmp                         # Only catch pings
# tcpdump -i eth0 -s 0 -A port 80 | grep GET      # -s 0 for full packet -A for ASCII
```

Additional important options:

`-A`	Print each packets in clear text (without header)
`-X`	Print packets in hex and ASCII
`-l`	Make stdout line buffered
`-D`	Print all interfaces available

On Windows use windump from www.winpcap.org. Use windump -D to list the interfaces.

Scan with nmap

Nmap[3] is a port scanner with OS detection, it is usually installed on most distributions and is also available for Windows. If you don't scan your servers, hackers do it for you...

```
# nmap cb.vu               # scans all reserved TCP ports on the host
# nmap cb.vu -p 80              # scans TCP port 80 on the remote host
# nmap -sP 192.168.16.0/24 # Find out which IP are used and by which host on 0/24
# nmap -sS -sV -O cb.vu    # Do a stealth SYN scan with version and OS detection
PORT      STATE  SERVICE             VERSION
22/tcp    open   ssh                 OpenSSH 3.8.1p1 FreeBSD-20060930 (protocol 2.0)
25/tcp    open   smtp                Sendmail smtpd 8.13.6/8.13.6
80/tcp    open   http                Apache httpd 2.0.59 ((FreeBSD) DAV/2 PHP/4.
[...]
Running: FreeBSD 5.X
Uptime 33.120 days (since Fri Aug 31 11:41:04 2007)
```

Other non standard but useful tools are hping (www.hping.org) an IP packet assembler/analyzer and `fping` (fping.sourceforge.net). fping can check multiple hosts in a round-robin fashion.

4.11 Netcat

Netcat[4] (nc) is better known as the "network Swiss Army Knife", it can manipulate, create or read/

2. http://people.suug.ch/~tgr/bmon/
3. http://insecure.org/nmap/
4. http://netcat.sourceforge.net

write TCP/IP connections. Here some useful examples, there are many more on the net, for example g-loaded.eu[...][5] and here[6].
You might need to use the command `netcat` instead of `nc`. Also see the similar command socat.

File transfer

Copy a large folder over a raw tcp connection. The transfer is very quick (no protocol overhead) and you don't need to mess up with NFS or SMB or FTP or so, simply make the file available on the server, and get it from the client. Here 192.168.1.1 is the server IP address.

```
server# tar -cf - -C VIDEO_TS . | nc -l -p 4444        # Serve tar folder on port 4444
client# nc 192.168.1.1 4444 | tar xpf - -C VIDEO_TS    # Pull the file on port 4444
server# cat largefile | nc -l 5678                     # Server a single file
client# nc 192.168.1.1 5678 > largefile                # Pull the single file
server# dd if=/dev/da0 | nc -l 4444                    # Server partition image
client# nc 192.168.1.1 4444 | dd of=/dev/da0           # Pull partition to clone
client# nc 192.168.1.1 4444 | dd of=da0.img            # Pull partition to file
```

Other hacks

Specially here, you must know what you are doing.

Remote shell

Option -e only on the Windows version? Or use nc 1.10.

```
# nc -lp 4444 -e /bin/bash                          # Provide a remote shell (server backdoor)
# nc -lp 4444 -e cmd.exe                            # remote shell for Windows
```

Emergency web server

Serve a single file on port 80 in a loop.

```
# while true; do nc -l -p 80 < unixtoolbox.xhtml; done
```

Chat

Alice and Bob can chat over a simple TCP socket. The text is transferred with the enter key.

```
alice# nc -lp 4444
bob  # nc 192.168.1.1 4444
```

5 SSH AND SCP

Public key (p13) | Fingerprint (p14) | SCP (p14) | Tunneling (p15) | SSHFS (p17)

See other tricks 25 ssh cmd[7]

5.1 Public key authentication

Connect to a host without password using public key authentication. The idea is to append your public key to the authorized_keys2 file on the remote host. For this example let's **connect *host-client* to *host-server***, the key is generated on the client. With cygwin you might have to create your home directoy and the .ssh directory with `# mkdir -p /home/USER/.ssh`

- Use ssh-keygen to generate a key pair. `~/.ssh/id_dsa` is the private key, `~/.ssh/id_dsa.pub` is the public key.
- Copy only the public key to the server and append it to the file `~/.ssh/authorized_keys2` on your home on the server.

```
# ssh-keygen -t dsa -N ''
# cat ~/.ssh/id_dsa.pub | ssh you@host-server "cat - >> ~/.ssh/authorized_keys2"
```

Using the Windows client from ssh.com

The non commercial version of the ssh.com client can be downloaded the main ftp site: ftp.ssh.com/

5. http://www.g-loaded.eu/2006/11/06/netcat-a-couple-of-useful-examples
6. http://www.terminally-incoherent.com/blog/2007/08/07/few-useful-netcat-tricks
7. http://blog.urfix.com/25-ssh-commands-tricks/

pub/ssh/. Keys generated by the ssh.com client need to be converted for the OpenSSH server. This can be done with the ssh-keygen command.

- Create a key pair with the ssh.com client: Settings - User Authentication - Generate New....
- I use Key type DSA; key length 2048.
- Copy the public key generated by the ssh.com client to the server into the ~/.ssh folder.
- The keys are in C:\Documents and Settings\%USERNAME%\Application Data\SSH\UserKeys.
- Use the ssh-keygen command on the server to convert the key:

```
# cd ~/.ssh
# ssh-keygen -i -f keyfilename.pub >> authorized_keys2
```

Notice: We used a DSA key, RSA is also possible. The key is not protected by a password.

Using putty for Windows

Putty[8] is a simple and free ssh client for Windows.

- Create a key pair with the puTTYgen program.
- Save the public and private keys (for example into C:\Documents and Settings\%USERNAME%\.ssh).
- Copy the public key to the server into the ~/.ssh folder:

```
# scp .ssh/puttykey.pub root@192.168.51.254:.ssh/
```

- Use the ssh-keygen command on the server to convert the key for OpenSSH:

```
# cd ~/.ssh
# ssh-keygen -i -f puttykey.pub >> authorized_keys2
```

- Point the private key location in the putty settings: Connection - SSH - Auth

5.2 Check fingerprint

At the first login, ssh will ask if the unknown host with the fingerprint has to be stored in the known hosts. To avoid a man-in-the-middle attack the administrator of the server can send you the server fingerprint which is then compared on the first login. Use `ssh-keygen -l` to get the fingerprint (on the server):

```
# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub      # For RSA key
2048 61:33:be:9b:ae:6c:36:31:fd:83:98:b7:99:2d:9f:cd /etc/ssh/ssh_host_rsa_key.pub
# ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub      # For DSA key (default)
2048 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee /etc/ssh/ssh_host_dsa_key.pub
```

Now the client connecting to this server can verify that he is connecting to the right server:

```
# ssh linda
The authenticity of host 'linda (192.168.16.54)' can't be established.
DSA key fingerprint is 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee.
Are you sure you want to continue connecting (yes/no)? yes
```

5.3 Secure file transfer

Some simple commands:

```
# scp file.txt host-two:/tmp
# scp joe@host-two:/www/*.html /www/tmp
# scp -r joe@host-two:/www /www/tmp
# scp -P 20022 cb@cb.vu:unixtoolbox.xhtml .           # connect on port 20022
```

In Konqueror or Midnight Commander it is possible to access a remote file system with the address **fish://user@gate**. However the implementation is very slow.
Furthermore it is possible to mount a remote folder with **sshfs** a file system client based on SCP. See fuse sshfs[9].

```
ssh_exchange_identification: Connection closed by remote host
```

With this error try the following on the server:

8. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
9. http://fuse.sourceforge.net/sshfs.html

```
echo 'SSHD: ALL' >> /etc/hosts.allow
/etc/init.d/sshd restart
```

5.4 Tunneling

SSH tunneling allows to forward or reverse forward a port over the SSH connection, thus securing the traffic and accessing ports which would otherwise be blocked. This only works with TCP. The general nomenclature for forward and reverse is (see also ssh and NAT example):

```
# ssh -L localport:desthost:destport user@gate  # desthost as seen from the gate
# ssh -R destport:desthost:localport user@gate  # forwards your localport to destination
    # desthost:localport as seen from the client initiating the tunnel
# ssh -X user@gate   # To force X forwarding
```

This will connect to gate and forward the local port to the host desthost:destport. Note desthost is the destination host *as seen by the gate*, so if the connection is to the gate, then desthost is localhost. More than one port forward is possible.

Direct forward on the gate

Let say we want to access the CVS (port 2401) and http (port 80) which are running on the gate. This is the simplest example, desthost is thus localhost, and we use the port 8080 locally instead of 80 so we don't need to be root. Once the ssh session is open, both services are accessible on the local ports.

```
# ssh -L 2401:localhost:2401 -L 8080:localhost:80 user@gate
```

Netbios and remote desktop forward to a second server

Let say a Windows smb server is behind the gate and is not running ssh. We need access to the smb share and also remote desktop to the server.

```
# ssh -L 139:smbserver:139 -L 3388:smbserver:3389 user@gate
```

The smb share can now be accessed with \\127.0.0.1\, but only if the local share is disabled, because *the local share is listening on port 139*.
It is possible to keep the local share enabled, for this we need to create a new virtual device with a new IP address for the tunnel, the smb share will be connected over this address. Furthermore *the local RDP is already listening on 3389*, so we choose 3388. For this example let's use a virtual IP of 10.1.1.1.

- With putty use Source port=10.1.1.1:139. It is possible to create multiple loop devices and tunnel. On Windows 2000, only putty worked for me. On Windows Vista also forward the port 445 in addition to the port 139. Also on Vista the patch KB942624 prevents the port 445 to be forwarded, so I had to uninstall this path in Vista.
- With the ssh.com client, disable "Allow local connections only". Since ssh.com will bind to all addresses, only a single share can be connected.

Now create the loopback interface with IP 10.1.1.1:

- # System->Control Panel->Add Hardware # Yes, Hardware is already connected # Add a new hardware device (at bottom).
- # Install the hardware that I manually select # Network adapters # Microsoft , Microsoft Loopback Adapter.
- Configure the IP address of the fake device to 10.1.1.1 mask 255.255.255.0, no gateway.
- advanced->WINS, Enable LMHosts Lookup; Disable NetBIOS over TCP/IP.
- # Enable Client for Microsoft Networks. # Disable File and Printer Sharing for Microsoft Networks.

I HAD to reboot for this to work. Now connect to the smb share with \\10.1.1.1 and remote desktop to 10.1.1.1:3388.

Debug

If it is not working:

- Are the ports forwarded: netstat -an? Look at 0.0.0.0:139 or 10.1.1.1:139
- Does telnet 10.1.1.1 139 connect?
- You need the checkbox "Local ports accept connections from other hosts".
- Is "File and Printer Sharing for Microsoft Networks" disabled on the loopback interface?

Connect two clients behind NAT

Suppose two clients are behind a NAT gateway and client cliadmin has to connect to client cliuser (the destination), both can login to the gate with ssh and are running Linux with sshd. You don't need root access anywhere as long as the ports on gate are above 1024. We use 2022 on gate. Also since the gate is used locally, the option GatewayPorts is not necessary.
On client cliuser (from destination to gate):

```
# ssh -R 2022:localhost:22 user@gate             # forwards client 22 to gate:2022
```

On client cliadmin (from host to gate):

```
# ssh -L 3022:localhost:2022 admin@gate          # forwards client 3022 to gate:2022
```

Now the admin can connect directly to the client cliuser with:

```
# ssh -p 3022 admin@localhost                    # local:3022 -> gate:2022 -> client:22
```

Connect to VNC behind NAT

Suppose a Windows client with VNC listening on port 5900 has to be accessed from behind NAT. On client cliwin to gate:

```
# ssh -R 15900:localhost:5900 user@gate
```

On client cliadmin (from host to gate):

```
# ssh -L 5900:localhost:15900 admin@gate
```

Now the admin can connect directly to the client VNC with:

```
# vncconnect -display :0 localhost
```

Dig a multi-hop ssh tunnel

Suppose you can not reach a server directly with ssh, but only via multiple intermediate hosts (for example because of routing issues). Sometimes it is still necessary to get a direct client - server connection, for example to copy files with scp, or forward other ports like smb or vnc. One way to do this is to chain tunnels together to forward a port to the server along the hops. This "carrier" port only reaches its final destination on the last connection to the server.
Suppose we want to forward the ssh port from a client to a server over two hops. Once the tunnel is build, it is possible to connect to the server directly from the client (and also add an other port forward).

Create tunnel in one shell

client -> host1 -> host2 -> server and dig tunnel 5678

```
client># ssh -L5678:localhost:5678 host1        # 5678 is an arbitrary port for the tunnel
host_1># ssh -L5678:localhost:5678 host2        # chain 5678 from host1 to host2
host_2># ssh -L5678:localhost:22 server         # end the tunnel on port 22 on the server
```

Use tunnel with an other shell

client -> server using tunnel 5678

```
# ssh -p 5678 localhost                         # connect directly from client to  server
# scp -P 5678 myfile localhost:/tmp/            # or copy a file directly using the tunnel
# rsync -e 'ssh -p 5678' myfile localhost:/tmp/ # or rsync a file directly to the server
```

Autoconnect and keep alive script

I use variations of the following script to keep a machine reacheable over a reverse ssh tunnel. The connection is automatically rebuilt if closed. You can add multiple -L or -R tunnels on one line.

```
#!/bin/sh
COMMAND="ssh -N -f -g -R 3022:localhost:22 colin@cb.vu"
pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND
exit 0
```

```
1 * * * * colin /home/colin/port_forward.sh     # crontab entry (here hourly)
```

5.1 sshfs

Mount a filesystem with ssh.

```
# sshfs cb@cb.vu:/ /Users/barschel/cbvu -oauto_cache,reconnect,defer_permissions \
    ,noappledouble,negative_vncache,volname=cbvu
```

Or via a two hops tunnel

```
# ssh -Y -A -t -L20022:127.0.0.1:20022 cbarsche@lbgw ssh -Y -A -t -L20022:127.0.0.1:22 rootbgv@bgvctrl
# sshfs -p 20022 cb@cb.vu:/ /Users/barschel/cbvu -oauto_cache,reconnect,defer_permissions \
    ,noappledouble,negative_vncache,volname=cbvu
```

6 VPN WITH SSH

As of version 4.3, OpenSSH can use the tun/tap device to encrypt a tunnel. This is very similar to other TLS based VPN solutions like OpenVPN. One advantage with SSH is that there is no need to install and configure additional software. Additionally the tunnel uses the SSH authentication like pre shared keys. The drawback is that the encapsulation is done over TCP which might result in poor performance on a slow link. Also the tunnel is relying on a single (fragile) TCP connection. This technique is very useful for a quick IP based VPN setup. There is no limitation as with the single TCP port forward, all layer 3/4 protocols like ICMP, TCP/UDP, etc. are forwarded over the VPN. In any case, the following options are needed in the sshd_conf file:

```
PermitRootLogin yes
PermitTunnel yes
```

6.1 Single P2P connection

Here we are connecting two hosts, hclient and hserver with a peer to peer tunnel. The connection is *started from hclient* to hserver and is done as root. The tunnel end points are 10.0.1.1 (server) and 10.0.1.2 (client) and we create a device tun5 (this could also be an other number). The procedure is very simple:

- Connect with SSH using the tunnel option -w
- Configure the IP addresses of the tunnel. Once on the server and once on the client.

Connect to the server

Connection started on the client and commands are executed on the server.

Server is on Linux

```
cli># ssh -w5:5 root@hserver
srv># ifconfig tun5 10.0.1.1 netmask 255.255.255.252   # Executed on the server shell
```

Server is on FreeBSD

```
cli># ssh -w5:5 root@hserver
srv># ifconfig tun5 10.0.1.1 10.0.1.2                  # Executed on the server shell
```

Configure the client

Commands executed on the client:

```
cli># ifconfig tun5 10.0.1.2 netmask 255.255.255.252   # Client is on Linux
cli># ifconfig tun5 10.0.1.2 10.0.1.1                  # Client is on FreeBSD
```

The two hosts are now connected and can transparently communicate with any layer 3/4 protocol using the tunnel IP addresses.

6.2 Connect two networks

In addition to the p2p setup above, it is more useful to connect two private networks with an SSH VPN using two gates. Suppose for the example, netA is 192.168.51.0/24 and netB 192.168.16.0/24.

The procedure is similar as above, we only need to add the routing. NAT must be activated on the private interface only if the gates are not the same as the default gateway of their network.
192.168.51.0/24 (netA)|gateA <-> gateB|192.168.16.0/24 (netB)

- Connect with SSH using the tunnel option -w.
- Configure the IP addresses of the tunnel. Once on the server and once on the client.
- Add the routing for the two networks.
- If necessary, activate NAT on the private interface of the gate.

The setup is *started from gateA in netA*.

Connect from gateA to gateB

Connection is started from gateA and commands are executed on gateB.

gateB is on Linux

```
gateA># ssh -w5:5 root@gateB
gateB># ifconfig tun5 10.0.1.1 netmask 255.255.255.252 # Executed on the gateB shell
gateB># route add -net 192.168.51.0 netmask 255.255.255.0 dev tun5
gateB># echo 1 > /proc/sys/net/ipv4/ip_forward        # Only needed if not default gw
gateB># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

gateB is on FreeBSD

```
gateA># ssh -w5:5 root@gateB                          # Creates the tun5 devices
gateB># ifconfig tun5 10.0.1.1 10.0.1.2               # Executed on the gateB shell
gateB># route add 192.168.51.0/24 10.0.1.2
gateB># sysctl net.inet.ip.forwarding=1               # Only needed if not default gw
gateB># natd -s -m -u -dynamic -n fxp0                # see NAT (page 10)
gateA># sysctl net.inet.ip.fw.enable=1
```

Configure gateA

Commands executed on gateA:

gateA is on Linux

```
gateA># ifconfig tun5 10.0.1.2 netmask 255.255.255.252
gateA># route add -net 192.168.16.0 netmask 255.255.255.0 dev tun5
gateA># echo 1 > /proc/sys/net/ipv4/ip_forward
gateA># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

gateA is on FreeBSD

```
gateA># ifconfig tun5 10.0.1.2 10.0.1.1
gateA># route add 192.168.16.0/24 10.0.1.2
gateA># sysctl net.inet.ip.forwarding=1
gateA># natd -s -m -u -dynamic -n fxp0                # see NAT (page 10)
gateA># sysctl net.inet.ip.fw.enable=1
```

The two private networks are now transparently connected via the SSH VPN. The IP forward and NAT settings are only necessary if the gates are not the default gateways. In this case the clients would not know where to forward the response, and nat must be activated.

7 RSYNC

Rsync can almost completely replace cp and scp, furthermore interrupted transfers are efficiently restarted. A trailing slash (and the absence thereof) has different meanings, the man page is good... Here some examples:
Copy the directories with full content:

```
# rsync -a /home/colin/ /backup/colin/                # "archive" mode. e.g keep the same
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # use relative (see below)
# /opt/local/bin/rsync -azv --iconv=UTF-8-MAC,UTF-8 ~/Music/flac/ me@server:/dst/
                           # convert filenames OSX UTF8 to Windows UTF8
```

Same as before but over the network and with compression. Rsync uses SSH for the transport per default and will use the ssh key if they are set. Use ":" as with SCP. A typical remote copy:

```
# rsync -axSRzv /home/user/ user@server:/backup/user/ # Copy to remote
# rsync -a 'user@server:My\ Documents' My\ Documents  # Quote AND escape spaces for the remote shell
```

Exclude any directory tmp within /home/user/ and keep the relative folders hierarchy, that is the remote directory will have the structure /backup/home/user/. This is typically used for backups.

```
# rsync -azR --exclude=tmp/ /home/user/ user@server:/backup/
```

Use port 20022 for the ssh connection:

```
# rsync -az -e 'ssh -p 20022' /home/colin/ user@server:/backup/colin/
```

Using the rsync daemon (used with "::") is much faster, but not encrypted over ssh. The location of /backup is defined by the configuration in /etc/rsyncd.conf. The variable RSYNC_PASSWORD can be set to avoid the need to enter the password manually.

```
# rsync -axSRz /home/ ruser@hostname::rmodule/backup/
# rsync -axSRz ruser@hostname::rmodule/backup/ /home/    # To copy back
```

Some important options:

`-a, --archive`	archive mode; same as -rlptgoD (no -H)
`-r, --recursive`	recurse into directories
`-R, --relative`	use relative path names
`-H, --hard-links`	preserve hard links
`-S, --sparse`	handle sparse files efficiently
`-x, --one-file-system`	don't cross file system boundaries
`--exclude=PATTERN`	exclude files matching PATTERN
`--delete-during`	receiver deletes during xfer, not before
`--delete-after`	receiver deletes after transfer, not before

7.1 Rsync on Windows

Rsync is available for Windows through cygwin or as stand-alone packaged in cwrsync[10]. This is very convenient for automated backups. Install one of them (*not both*) and add the path to the Windows system variables: # Control Panel -> System -> tab Advanced, button Environment Variables. Edit the "Path" system variable and add the full path to the installed rsync, e.g. C:\Program Files\cwRsync\bin or C:\cygwin\bin. This way the commands `rsync` and `ssh` are available in a Windows command shell.

Public key authentication

Rsync is automatically tunneled over SSH and thus uses the SSH authentication on the server. Automatic backups have to avoid a user interaction, for this the SSH public key authentication can be used and the rsync command will run without a password.
All the following commands are executed within a Windows console. In a console (Start -> Run -> cmd) create and upload the key as described in SSH, change "user" and "server" as appropriate. If the file authorized_keys2 does not exist yet, simply copy id_dsa.pub to authorized_keys2 and upload it.

```
# ssh-keygen -t dsa -N ''                      # Creates a public and a private key
# rsync user@server:.ssh/authorized_keys2 . # Copy the file locally from the server
# cat id_dsa.pub >> authorized_keys2        # Or use an editor to add the key
# rsync authorized_keys2 user@server:.ssh/  # Copy the file back to the server
# del authorized_keys2                      # Remove the local copy
```

Now test it with (in one line):

```
rsync -rv "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \
'user@server:My\ Documents/'
```

Automatic backup

Use a batch file to automate the backup and add the file in the scheduled tasks (Programs -> Accessories -> System Tools -> Scheduled Tasks). For example create the file backup.bat and replace user@server.

10. http://sourceforge.net/projects/sereds

```
@ECHO OFF
REM rsync the directory My Documents
SETLOCAL
SET CWRSYNCHOME=C:\PROGRAM FILES\CWRSYNC
SET CYGWIN=nontsec
SET CWOLDPATH=%PATH%
REM uncomment the next line when using cygwin
SET PATH=%CWRSYNCHOME%\BIN;%PATH%
echo Press Control-C to abort
rsync -av "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \
'user@server:My\ Documents/'
pause
```

8 SUDO

Sudo is a standard way to give users some administrative rights without giving out the root password. Sudo is very useful in a multi user environment with a mix of server and workstations. Simply call the command with sudo:

```
# sudo /etc/init.d/dhcpd restart            # Run the rc script as root
# sudo -u sysadmin whoami                   # Run cmd as an other user
```

8.1 Configuration

Sudo is configured in `/etc/sudoers` and must only be edited with `visudo`. The basic syntax is (the lists are comma separated):

```
user hosts = (runas) commands          # In /etc/sudoers
```

`users`	one or more users or %group (like %wheel) to gain the rights
`hosts`	list of hosts (or ALL)
`runas`	list of users (or ALL) that the command rule can be run as. It is enclosed in ( )!
`commands`	list of commands (or ALL) that will be run as root or as (runas)

Additionally those keywords can be defined as alias, they are called User_Alias, Host_Alias, Runas_Alias and Cmnd_Alias. This is useful for larger setups. Here a sudoers example:

```
# cat /etc/sudoers
# Host aliases are subnets or hostnames.
Host_Alias   DMZ     = 212.118.81.40/28
Host_Alias   DESKTOP = work1, work2

# User aliases are a list of users which can have the same rights
User_Alias   ADMINS  = colin, luca, admin
User_Alias   DEVEL   = joe, jack, julia
Runas_Alias  DBA     = oracle,pgsql

# Command aliases define the full path of a list of commands
Cmnd_Alias   SYSTEM  = /sbin/reboot,/usr/bin/kill,/sbin/halt,/sbin/shutdown,/etc/init.d/
Cmnd_Alias   PW      = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root # Not root pwd!
Cmnd_Alias   DEBUG   = /usr/sbin/tcpdump,/usr/bin/wireshark,/usr/bin/nmap
```

```
# The actual rules
root,ADMINS  ALL     = (ALL) NOPASSWD: ALL    # ADMINS can do anything w/o a password.
DEVEL        DESKTOP = (ALL) NOPASSWD: ALL    # Developers have full right on desktops
DEVEL        DMZ     = (ALL) NOPASSWD: DEBUG  # Developers can debug the DMZ servers.

# User sysadmin can mess around in the DMZ servers with some commands.
sysadmin     DMZ     = (ALL) NOPASSWD: SYSTEM,PW,DEBUG
sysadmin     ALL,!DMZ = (ALL) NOPASSWD: ALL   # Can do anything outside the DMZ.
%dba         ALL     = (DBA) ALL              # Group dba can run as database user.

# anyone can mount/unmount a cd-rom on the desktop machines
ALL          DESKTOP = NOPASSWD: /sbin/mount /cdrom,/sbin/umount /cdrom
```

9 ENCRYPT FILES

9.1 OpenSSL

A single file

Encrypt and decrypt:

```
# openssl aes-128-cbc -salt -in file -out file.aes
# openssl aes-128-cbc -d -salt -in file.aes -out file
```

Note that the file can of course be a tar archive.

tar and encrypt a whole directory

```
# tar -cf - directory | openssl aes-128-cbc -salt -out directory.tar.aes      # Encrypt
# openssl aes-128-cbc -d -salt -in directory.tar.aes | tar -x -f -            # Decrypt
```

tar zip and encrypt a whole directory

```
# tar -zcf - directory | openssl aes-128-cbc -salt -out directory.tar.gz.aes  # Encrypt
# openssl aes-128-cbc -d -salt -in directory.tar.gz.aes | tar -xz -f -        # Decrypt
```

- Use -k mysecretpassword after aes-128-cbc to avoid the interactive password request. However note that this is highly insecure.
- Use **aes-256-cbc** instead of **aes-128-cbc** to get even stronger encryption. This uses also more CPU.

9.2 GPG

GnuPG is well known to encrypt and sign emails or any data. Furthermore gpg and also provides an advanced key management system. This section only covers files encryption, not email usage, signing or the Web-Of-Trust.
The simplest encryption is with a symmetric cipher. In this case the file is encrypted with a password and anyone who knows the password can decrypt it, thus the keys are not needed. Gpg adds an extention ".gpg" to the encrypted file names.

```
# gpg -c file                         # Encrypt file with password
# gpg file.gpg                        # Decrypt file (optionally -o otherfile)
```

Using keys

For more details see GPG Quick Start[11] and GPG/PGP Basics[12] and the gnupg documentation[13] among others.
The private and public keys are the heart of asymmetric cryptography. What is important to remember:

- Your public key is used by *others* to encrypt files that only you as the receiver can decrypt (not even the one who encrypted the file can decrypt it). The public key is thus meant to be distributed.
- Your private key is encrypted with your passphrase and is used to decrypt files which were encrypted with *your* public key. The private key must be kept **secure**. Also if the key or passphrase is lost, so are all the files encrypted with your public key.
- The key files are called keyrings as they can contain more than one key.

First generate a key pair. The defaults are fine, however you will have to enter at least your full name and email and optionally a comment. The comment is useful to create more than one key with the same name and email. Also you should use a "passphrase", not a simple password.

```
# gpg --gen-key                       # This can take a long time
```

The keys are stored in ~/.gnupg/ on Unix, on Windows they are typically stored in
C:/Documents and Settings/%USERNAME%/Application Data/gnupg/.

11. http://www.madboa.com/geek/gpg-quickstart
12. http://aplawrence.com/Basics/gpg.html
13. http://gnupg.org/documentation

```
~/.gnupg/pubring.gpg                 # Contains your public keys and all others imported
~/.gnupg/secring.gpg                 # Can contain more than one private key
```

Short reminder on most used options:

-e encrypt data
-d decrypt data
-r NAME encrypt for recipient NAME (or 'Full Name' or 'email@domain')
-a create ascii armored output of a key
-o use as output file

The examples use 'Your Name' and 'Alice' as the keys are referred to by the email or full name or partial name. For example I can use 'Colin' or 'c@cb.vu' for my key [Colin Barschel (cb.vu) <c@cb.vu>].

Encrypt for personal use only

No need to export/import any key for this. You have both already.

```
# gpg -e -r 'Your Name' file                       # Encrypt with your public key
# gpg -o file -d file.gpg                          # Decrypt. Use -o or it goes to stdout
```

Encrypt - Decrypt with keys

First you need to export your public key for someone else to use it. And you need to import the public say from Alice to encrypt a file for her. You can either handle the keys in simple ascii files or use a public key server.
For example Alice export her public key and you import it, you can then encrypt a file for her. That is only Alice will be able to decrypt it.

```
# gpg -a -o alicekey.asc --export 'Alice'          # Alice exported her key in ascii file.
# gpg --send-keys --keyserver subkeys.pgp.net KEYID   # Alice put her key on a server.
# gpg --import alicekey.asc                        # You import her key into your pubring.
# gpg --search-keys --keyserver subkeys.pgp.net 'Alice' # or get her key from a server.
```

Once the keys are imported it is very easy to encrypt or decrypt a file:

```
# gpg -e -r 'Alice' file                           # Encrypt the file for Alice.
# gpg -d file.gpg -o file                          # Decrypt a file encrypted by Alice for you.
```

Key administration

```
# gpg --list-keys                                  # list public keys and see the KEYIDS
    The KEYID follows the '/' e.g. for: pub   1024D/D12B77CE the KEYID is D12B77CE
# gpg --gen-revoke 'Your Name'                     # generate revocation certificate
# gpg --list-secret-keys                           # list private keys
# gpg --delete-keys NAME                           # delete a public key from local key ring
# gpg --delete-secret-key NAME                     # delete a secret key from local key ring
# gpg --fingerprint KEYID                          # Show the fingerprint of the key
# gpg --edit-key KEYID                             # Edit key (e.g sign or add/del email)
```

10 SSL CERTIFICATES

So called SSL/TLS certificates are cryptographic public key certificates and are composed of a public and a private key. The certificates are used to authenticate the endpoints and encrypt the data. They are used for example on a web server (https) or mail server (imaps).

10.1 Procedure

- We need a certificate authority to sign our certificate. This step is usually provided by a vendor like Thawte, Verisign, etc., however we can also create our own.
- Create a certificate signing request. This request is like an unsigned certificate (the public part) and already contains all necessary information. The certificate request is normally sent to the authority vendor for signing. This step also creates the private key on the local machine.

- Sign the certificate with the certificate authority.
- If necessary join the certificate and the key in a single file to be used by the application (web server, mail server etc.).

10.2 Configure OpenSSL

We use /usr/local/certs as directory for this example check or edit /etc/ssl/openssl.cnf accordingly to your settings so you know where the files will be created. Here are the relevant part of openssl.cnf:

```
[ CA_default ]
dir             = /usr/local/certs/CA       # Where everything is kept
certs           = $dir/certs                # Where the issued certs are kept
crl_dir         = $dir/crl                  # Where the issued crl are kept
database        = $dir/index.txt            # database index file.
```

Make sure the directories exist or create them

```
# mkdir -p /usr/local/certs/CA
# cd /usr/local/certs/CA
# mkdir certs crl newcerts private
# echo "01" > serial                        # Only if serial does not exist
# touch index.txt
```

If you intend to get a signed certificate from a vendor, you only need a certificate signing request (CSR). This CSR will then be signed by the vendor for a limited time (e.g. 1 year).

10.3 Create a certificate authority

If you do not have a certificate authority from a vendor, you'll have to create your own. This step is not necessary if one intend to use a vendor to sign the request. To make a certificate authority (CA):

```
# openssl req -new -x509 -days 730 -config /etc/ssl/openssl.cnf \
-keyout CA/private/cakey.pem -out CA/cacert.pem
```

10.4 Create a certificate signing request

To make a new certificate (for mail server or web server for example), first create a request certificate with its private key. If your application do not support encrypted private key (for example UW-IMAP does not), then disable encryption with `-nodes`.

```
# openssl req -new -keyout newkey.pem -out newreq.pem \
-config /etc/ssl/openssl.cnf
# openssl req -nodes -new -keyout newkey.pem -out newreq.pem \
-config /etc/ssl/openssl.cnf                # No encryption for the key
```

Keep this created CSR (`newreq.pem`) as it can be signed again at the next renewal, the signature onlt will limit the validity of the certificate. This process also created the private key `newkey.pem`.

10.5 Sign the certificate

The certificate request has to be signed by the CA to be valid, this step is usually done by the vendor. *Note: replace "servername" with the name of your server in the next commands.*

```
# cat newreq.pem newkey.pem > new.pem
# openssl ca -policy policy_anything -out servernamecert.pem \
-config /etc/ssl/openssl.cnf -infiles new.pem
# mv newkey.pem servernamekey.pem
```

Now servernamekey.pem is the private key and servernamecert.pem is the server certificate.

10.6 Create united certificate

The IMAP server wants to have both private key and server certificate in the same file. And in general, this is also easier to handle, but the file has to be kept securely!. Apache also can deal with it well. Create a file servername.pem containing both the certificate and key.

- Open the private key (servernamekey.pem) with a text editor and copy the private key into the "servername.pem" file.

- Do the same with the server certificate (servernamecert.pem).

The final servername.pem file should look like this:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDutWy+o/XZ/[...]qK5LqQgT3c9dU6fcR+WuSs6aejdEDDqBRQ
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIERzCCA7CgAwIBAgIBBDANB[...]iG9w0BAQQFADCBxTELMAkGA1UEBhMCREUx
-----END CERTIFICATE-----
```

What we have now in the directory /usr/local/certs/:

CA/private/cakey.pem *(CA server private key)*
CA/cacert.pem *(CA server public key)*
certs/servernamekey.pem *(server private key)*
certs/servernamecert.pem *(server signed certificate)*
certs/servername.pem *(server certificate with private key)*

Keep the private key secure!

10.7 View certificate information

To view the certificate information simply do:

```
# openssl x509 -text -in servernamecert.pem      # View the certificate info
# openssl req -noout -text -in server.csr        # View the request info
# openssl s_client -connect cb.vu:443            # Check a web server certificate
```

11 USEFUL COMMANDS

less (p24) | vi (p24) | mail (p0) | tar (p25) | zip (p26) | dd (p26) | screen (p27) | find (p28) | Miscellaneous (p28)

11.1 less

The `less` command displays a text document on the console. It is present on most installation.

```
# less unixtoolbox.xhtml
```

Some important commands are (^N stands for [control]-[N]):

h H good help on display
f ^F ^V SPACE Forward one window (or N lines).
b ^B ESC-v Backward one window (or N lines).
F Forward forever; like "tail -f".
/pattern Search forward for (N-th) matching line.
?pattern Search backward for (N-th) matching line.
n Repeat previous search (for N-th occurrence).
N Repeat previous search in reverse direction.
q quit

11.2 vi

Vi is present on ANY Linux/Unix installation (not gentoo?) and it is therefore useful to know some basic commands. There are two modes: command mode and insertion mode. The commands mode is accessed with **[ESC]**, the insertion mode with **i**. Use `: help` if you are lost.
The editors nano and `pico` are usually available too and are easier (IMHO) to use.

Quit

:w newfilename save the file to newfilename
:wq or :x save and quit

:q! quit without saving

Search and move

/string	Search forward for string
?string	Search back for string
n	Search for next instance of string
N	Search for previous instance of string
{	Move a paragraph back
}	Move a paragraph forward
1G	Move to the first line of the file
nG	Move to the n th line of the file
G	Move to the last line of the file
:%s/OLD/NEW/g	Search and replace every occurrence

Delete copy paste text

dd (dw)	Cut current line (word)
D	Cut to the end of the line
x	Delete (cut) character
yy (yw)	Copy line (word) after cursor
P	Paste after cursor
u	Undo last modification
U	Undo all changes to current line

11.3 tar

The command `tar` (tape archive) creates and extracts archives of file and directories. The archive .tar is uncompressed, a compressed archive has the extension .tgz or .tar.gz (zip) or .tbz (bzip2). Do not use absolute path when creating an archive, you probably want to unpack it somewhere else. Some typical commands are:

Create

```
# cd /
# tar -cf home.tar home/        # archive the whole /home directory (c for create)
# tar -czf home.tgz home/       # same with zip compression
# tar -cjf home.tbz home/       # same with bzip2 compression
```

Only include one (or two) directories from a tree, but keep the relative structure. For example archive /usr/local/etc and /usr/local/www and the first directory in the archive should be local/.

```
# tar -C /usr -czf local.tgz local/etc local/www
# tar -C /usr -xzf local.tgz    # To untar the local dir into /usr
# cd /usr; tar -xzf local.tgz   # Is the same as above
```

Extract

```
# tar -tzf home.tgz             # look inside the archive without extracting (list)
# tar -xf home.tar              # extract the archive here (x for extract)
# tar -xzf home.tgz             # same with zip compression (-xjf for bzip2 compression)
                                # remove leading path gallery2 and extract into gallery
# tar --strip-components 1 -zxvf gallery2.tgz -C gallery/
# tar -xjf home.tbz home/colin/file.txt    # Restore a single file
# tar -xOf home.tbz home/colin/file.txt    # Print file to stdout (no extraction)
```

More advanced

```
# tar c dir/ | gzip | ssh user@remote 'dd of=dir.tgz' # arch dir/ and store remotely.
# tar cvf - `find . -print` > backup.tar              # arch the current directory.
# tar -cf - -C /etc . | tar xpf - -C /backup/etc      # Copy directories
# tar -cf - -C /etc . | ssh user@remote tar xpf - -C /backup/etc      # Remote copy.
# tar -czf home.tgz --exclude '*.o' --exclude 'tmp/' home/
```

11.4 zip/unzip

Zip files can be easier to share with Windows.

```
# zip -r fileName.zip /path/to/dir                    # zip dir into file fileName.zip
# unzip fileName.zip                                  # uncompress zip file
# unzip -l fileName.zip                               # list files inside archive
# unzip -c fileName.zip fileinside.txt                # print one file to stdout (no extraction)
# unzip fileName.zip fileinside.txt                   # extract one file only
```

11.5 dd

The program dd (disk dump or destroy disk or see the meaning of dd) is used to copy partitions and disks and for other copy tricks. Typical usage:

```
# dd if=<source> of=<target> bs=<byte size> conv=<conversion>
# kill -INFO PID                                      # View dd progress (FreeBSD, OSX)
```

Important conv options:

`notrunc`	do not truncate the output file, all zeros will be written as zeros.
`noerror`	continue after read errors (e.g. bad blocks)
`sync`	pad every input block with Nulls to ibs-size

The default byte size is 512 (one block). The MBR, where the partition table is located, is on the first block, the first 63 blocks of a disk are empty. Larger byte sizes are faster to copy but require also more memory.

Backup and restore

```
# dd if=/dev/hda of=/dev/hdc bs=16065b                # Copy disk to disk (same size)
# dd if=/dev/sda7 of=/home/root.img bs=4096 conv=notrunc,noerror # Backup /
# dd if=/home/root.img of=/dev/sda7 bs=4096 conv=notrunc,noerror # Restore /
# dd bs=1M if=/dev/ad4s3e | gzip -c > ad4s3e.gz                  # Zip the backup
# gunzip -dc ad4s3e.gz | dd of=/dev/ad0s3e bs=1M                 # Restore the zip
# dd bs=1M if=/dev/ad4s3e | gzip | ssh eedcoba@fry 'dd of=ad4s3e.gz' # also remote
# gunzip -dc ad4s3e.gz | ssh eedcoba@host 'dd of=/dev/ad0s3e bs=1M'
# dd if=/dev/ad0 of=/dev/ad2 skip=1 seek=1 bs=4k conv=noerror    # Skip MBR
    # This is necessary if the destination (ad2) is smaller.
# dd if=/vm/FreeBSD-8.2-RELEASE-amd64-memstick.img of=/dev/disk1 bs=10240 conv=sync
    # Copy FreeBSD image to USB memory stick
```

Recover

The command dd will read *every single block* of the partition. In case of problems it is better to use the option `conv=sync,noerror` so dd will skip the bad block and write zeros at the destination. Accordingly it is important to set the block size equal or smaller than the disk block size. A 1k size seems safe, set it with bs=1k. If a disk has bad sectors and the data should be recovered from a partition, create an image file with dd, mount the image and copy the content to a new disk. With the option `noerror`, dd will skip the bad sectors and write zeros instead, thus only the data contained in the bad sectors will be lost.

```
# dd if=/dev/hda of=/dev/null bs=1m                   # Check for bad blocks
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc | gzip | ssh \ # Send to remote
root@fry 'dd of=hda1.gz bs=1k'
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc of=hda1.img    # Store into an image
# mount -o loop /hda1.img /mnt                        # Mount the image (page 8)
# rsync -ax /mnt/ /newdisk/                           # Copy on a new disk
# dd if=/dev/hda of=/dev/hda                          # Refresh the magnetic state
  # The above is useful to refresh a disk. It is perfectly safe, but must be unmounted.
```

Delete

```
# dd if=/dev/zero of=/dev/hdc                         # Delete full disk
# dd if=/dev/urandom of=/dev/hdc                      # Delete full disk better
# kill -USR1 PID                                      # View dd progress (Linux)
# kill -INFO PID                                      # View dd progress (FreeBSD)
```

MBR tricks

The MBR contains the boot loader and the partition table and is 512 bytes small. The first 446 are for the boot loader, the bytes 446 to 512 are for the partition table.

```
# dd if=/dev/sda of=/mbr_sda.bak bs=512 count=1       # Backup the full MBR
# dd if=/dev/zero of=/dev/sda bs=512 count=1          # Delete MBR and partition table
# dd if=/mbr_sda.bak of=/dev/sda bs=512 count=1       # Restore the full MBR
# dd if=/mbr_sda.bak of=/dev/sda bs=446 count=1       # Restore only the boot loader
# dd if=/mbr_sda.bak of=/dev/sda bs=1 count=64 skip=446 seek=446 # Restore partition table
```

11.6 screen

Screen (a must have) has two main functionalities:

- Run multiple terminal session within a single terminal.
- A started program is decoupled from the real terminal and can thus run in the background. The real terminal can be closed and reattached later.

Short start example

start screen with:

```
# screen
```

Within the screen session we can start a long lasting program (like top).

```
# top
```

Now detach with **Ctrl-a Ctrl-d**. Reattach the terminal with:

```
# screen -R -D
```

In detail this means: If a session is running, then reattach. If necessary detach and logout remotely first. If it was not running create it and notify the user. Or:

```
# screen -x
```

Attach to a running screen in a multi display mode. The console is thus shared among multiple users. Very useful for team work/debug!

Screen commands (within screen)

All screen commands start with **Ctrl-a**.

- **Ctrl-a ?** help and summary of functions
- **Ctrl-a c** create an new window (terminal)
- **Ctrl-a Ctrl-n and Ctrl-a Ctrl-p** to switch to the next or previous window in the list, by number.
- **Ctrl-a Ctrl-N** where N is a number from 0 to 9, to switch to the corresponding window.
- **Ctrl-a "** to get a navigable list of running windows
- **Ctrl-a a** to clear a missed Ctrl-a
- **Ctrl-a Ctrl-d** to disconnect and leave the session running in the background
- **Ctrl-a x** lock the screen terminal with a password
- **Ctrl-a [** enter into **scrollback** mode, exit with **esc**.
  Use `echo "defscrollback 5000" > ~/.screenrc` to increase buffer (default is 100)
  - **C-u** Scrolls a half page up
  - **C-b** Scroll a full page up
  - **C-d** Scroll a half page down
  - **C-f** Scroll a full page down
  - **/** Search forward
  - **?** Search backward

Configuration in ~/.screenrc:

```
defscrollback 100000                 # increase scrollback buffer (default is 100)
termcapinfo xterm* ti@:te@           # avoid alternate text buffer to allow scrolling
```

The screen session is terminated when the program within the running terminal is closed and you logout from the terminal.

11.7 Find

Some important options:

-x (on BSD) -xdev (on Linux) Stay on the same file system (dev in fstab).
-exec cmd {} \; Execute the command and replace {} with the full path
-iname Like -name but is case insensitive
-ls Display information about the file (like ls -la)
-size n n is +-n (k M G T P)
-cmin n File's status was last changed n minutes ago.

```
# find . -type f ! -perm -444        # Find files not readable by all
# find . -type d ! -perm -111        # Find dirs not accessible by all
# find /home/user/ -cmin 10 -print   # Files created or modified in the last 10 min.
# find . -name '*.[ch]' | xargs grep -E 'expr' # Search 'expr' in this dir and below.
# find / -name "*.core" | xargs rm   # Find core dumps and delete them (also try core.*)
# find / -name "*.core" -print -exec rm {} \;  # Other syntax
      # Find images and create an archive, iname is not case sensitive. -r for append
# find . \( -iname "*.png" -o -iname "*.jpg" \) -print -exec tar -rf images.tar {} \;
# find . -type f -name "*.txt" ! -name README.txt -print  # Exclude README.txt files
# find /var/ -size +10M -exec ls -lh {} \;     # Find large files > 10 MB
# find /var/ -size +10M -ls           # This is simpler
# find . -size +10M -size -50M -print
# find /usr/ports/ -name work -type d -print -exec rm -rf {} \;  # Clean the ports
      # Find files with SUID; those file are vulnerable and must be kept secure
# find / -type f -user root -perm -4000 -exec ls -l {} \;
# find flac/ -iname *.flac -print -size +500k -exec /Applications/Fluke.app/Contents/MacOS/Fluke {} \;
                                      # I use above to add flac files to iTunes on OSX
```

Be careful with xarg or exec as it might or might not honor quotings and can return wrong results when files or directories contain spaces. In doubt use "-print0 | xargs -0" instead of "| xargs". The option -print0 must be the last in the find command. See this nice mini tutorial for find[14].

```
# find . -type f | xargs ls -l       # Will not work with spaces in names
# find . -type f -print0 | xargs -0 ls -l  # Will work with spaces in names
# find . -type f -exec ls -l '{}' \; # Or use quotes '{}' with -exec
```

Duplicate directory tree:

```
# find . -type d -exec mkdir -p /tmp/new_dest/{} \;
```

11.8 Miscellaneous

```
# which command                      # Show full path name of command
# time command                       # See how long a command takes to execute
# time cat                           # Use time as stopwatch. Ctrl-c to stop
# set | grep $USER                   # List the current environment
# cal -3                             # Display a three month calendar
# date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
# date 10022155                      # Set date and time
# whatis grep                        # Display a short info on the command or word
# whereis java                       # Search path and standard directories for word
# setenv varname value               # Set env. variable varname to value (csh/tcsh)
# export varname="value"             # set env. variable varname to value (sh/ksh/bash)
# pwd                                # Print working directory
# mkdir -p /path/to/dir              # no error if existing, make parent dirs as needed
# mkdir -p project/{bin,src,obj,doc/{html,man,pdf},debug/some/more/dirs}
# rmdir /path/to/dir                 # Remove directory
# rm -rf /path/to/dir                # Remove directory and its content (force)
# rm -- -badchar.txt                 # Remove file which starts with a dash (-)
# cp -la /dir1 /dir2                 # Archive and hard link files instead of copy
# cp -lpR /dir1 /dir2                # Same for FreeBSD
# cp unixtoolbox.xhtml{,.bak}        # Short way to copy the file with a new extension
# mv /dir1 /dir2                     # Rename a directory
# ls -1                              # list one file per line
# history | tail -50                 # Display the last 50 used commands
```

14. http://www.hccfl.edu/pollock/Unix/FindCmd.htm

```
# cd -                              # cd to previous ($OLDPWD) directory
# /bin/ls| grep -v .py | xargs rm -r # pipe file names to rm with xargs
```

Check file hashes with openssl. This is a nice alternative to the commands md5sum or sha1sum (FreeBSD uses md5 and sha1) which are not always installed.

```
# openssl md5 file.tar.gz           # Generate an md5 checksum from file
# openssl sha1 file.tar.gz          # Generate an sha1 checksum from file
# openssl rmd160 file.tar.gz        # Generate a RIPEMD-160 checksum from file
```

12 INSTALL SOFTWARE

Usually the package manager uses the proxy variable for http/ftp requests. In .bashrc:

```
export http_proxy=http://proxy_server:3128
export ftp_proxy=http://proxy_server:3128
```

12.1 List installed packages

```
# rpm -qa                           # List installed packages (RH, SuSE, RPM based)
# dpkg -l                           # Debian, Ubuntu
```

More on RPM:

```
# rpm -ql package-name              # list the files for INSTALLED package
# rpm -qlp package.rpm              # list the files inside package
```

12.2 Add/remove software

Front ends: yast2/yast for SuSE, redhat-config-packages for Red Hat.

```
# rpm -i pkgname.rpm                # install the package (RH, SuSE, RPM based)
# rpm -e pkgname                    # Remove package
```

SuSE zypper (see doc and cheet sheet)[15]

```
# zypper refresh                    # Refresh repositorie
# zypper install vim                # Install the package vim
# zypper remove vim                 # Remove the package vim
# zypper search vim                 # Search packages with vim
# zypper update vim                 # Search packages with vim
```

Debian

```
# apt-get update                    # First update the package lists
# apt-get install emacs             # Install the package emacs
# dpkg --remove emacs               # Remove the package emacs
# dpkg -S file                      # find what package a file belongs to
```

12.3 Library path

Due to complex dependencies and runtime linking, programs are difficult to copy to an other system or distribution. However for small programs with little dependencies, the missing libraries can be copied over. The runtime libraries (and the missing one) are checked with ldd and managed with ldconfig.

```
# ldd /usr/bin/rsync                # List all needed runtime libraries
# otool -L /usr/bin/rsync           # OS X equivalent to ldd
# ldconfig -n /path/to/libs/        # Add a path to the shared libraries directories
# LD_LIBRARY_PATH                   # The variable set the link library path
```

15. http://en.opensuse.org/SDB:Zypper_usage

13 DATABASES

13.1 PostgreSQL

Change root or a username password

```
# psql -d template1 -U pgsql
> alter user pgsql with password 'pgsql_password';  # Use username instead of "pgsql"
```

Create user and database

The commands `createuser`, `dropuser`, `createdb` and `dropdb` are convenient shortcuts equivalent to the SQL commands. The new user is bob with database bobdb ; use as root with pgsql the database super user:

```
# createuser -U pgsql -P bob         # -P will ask for password
# createdb -U pgsql -O bob bobdb     # new bobdb is owned by bob
# dropdb bobdb                       # Delete database bobdb
# dropuser bob                       # Delete user bob
```

The general database authentication mechanism is configured in pg_hba.conf

Grant remote access

The file `$PGSQL_DATA_D/postgresql.conf` specifies the address to bind to. Typically `listen_addresses = '*'` for Postgres 8.x.
The file `$PGSQL_DATA_D/pg_hba.conf` defines the access control. Examples:

```
# TYPE  DATABASE    USER        IP-ADDRESS        IP-MASK          METHOD
host    bobdb       bob        212.117.81.42     255.255.255.255   password
host    all         all        0.0.0.0/0                           password
```

Backup and restore

The backups and restore are done with the user pgsql or postgres. Backup and restore a single database:

```
# pg_dump --clean dbname > dbname_sql.dump
# psql dbname < dbname_sql.dump
```

Backup and restore all databases (including users):

```
# pg_dumpall --clean > full.dump
# psql -f full.dump postgres
```

In this case the restore is started with the database postgres which is better when reloading an empty cluster.

13.2 MySQL

Change mysql root or username password

Method 1

```
# /etc/init.d/mysql stop
or
# killall mysqld
# mysqld --skip-grant-tables
# mysqladmin -u root password 'newpasswd'
# /etc/init.d/mysql start
```

Method 2

```
# mysql -u root mysql
mysql> UPDATE USER SET PASSWORD=PASSWORD("newpassword") where user='root';
mysql> FLUSH PRIVILEGES;                           # Use username instead of "root"
mysql> quit
```

Create user and database (see MySQL doc[16])

```
# mysql -u root mysql
mysql> CREATE USER 'bob'@'localhost' IDENTIFIED BY 'pwd'; # create only a user
mysql> CREATE DATABASE bobdb;
mysql> GRANT ALL ON *.* TO 'bob'@'%' IDENTIFIED BY 'pwd'; # Use localhost instead of %
                                                     # to restrict the network access
mysql> DROP DATABASE bobdb;                          # Delete database
mysql> DROP USER bob;                                # Delete user
mysql> DELETE FROM mysql.user WHERE user='bob and host='hostname'; # Alt. command
mysql> FLUSH PRIVILEGES;
```

Grant remote access

Remote access is typically permitted for a database, and not all databases. The file `/etc/my.cnf` contains the IP address to bind to. (On FreeBSD `my.cnf` not created per fedault, copy one `.cnf` file from `/usr/local/share/mysql` to `/usr/local/etc/my.cnf`) Typically comment the line `bind-address =` out.

```
# mysql -u root mysql
mysql> GRANT ALL ON bobdb.* TO bob@'xxx.xxx.xxx.xxx' IDENTIFIED BY 'PASSWORD';
mysql> REVOKE GRANT OPTION ON foo.* FROM bar@'xxx.xxx.xxx.xxx';
mysql> FLUSH PRIVILEGES;                  # Use 'hostname' or also '%' for full access
```

Backup and restore

Backup and restore a single database:

```
# mysqldump -u root -psecret --add-drop-database dbname > dbname_sql.dump
# mysql -u root -psecret -D dbname < dbname_sql.dump
```

Backup and restore all databases:

```
# mysqldump -u root -psecret --add-drop-database --all-databases > full.dump
# mysql -u root -psecret < full.dump
```

Here is "secret" the mysql root password, there is no space after -p. When the -p option is used alone (w/o password), the password is asked at the command prompt.

13.3 SQLite

SQLite[17] is a small powerful self-contained, serverless, zero-configuration SQL database.

Dump and restore

It can be useful to dump and restore an SQLite database. For example you can edit the dump file to change a column attribute or type and then restore the database. This is easier than messing with SQL commands. Use the command `sqlite3` for a 3.x database.

```
# sqlite database.db .dump > dump.sql                # dump
# sqlite database.db < dump.sql                      # restore
```

Convert 2.x to 3.x database

```
sqlite database_v2.db .dump | sqlite3 database_v3.db
```

14 DISK QUOTA

A disk quota allows to limit the amount of disk space and/or the number of files a user or (or member of group) can use. The quotas are allocated on a per-file system basis and are enforced by the kernel.

14.1 Linux setup

The quota tools package usually needs to be installed, it contains the command line tools.
Activate the user quota in the fstab and remount the partition. If the partition is busy, either all

16. http://dev.mysql.com/doc/refman/5.1/en/adding-users.html
17. http://www.sqlite.org

locked files must be closed, or the system must be rebooted. Add `usrquota` to the fstab mount options, for example:

```
/dev/sda2     /home    reiserfs     rw,acl,user_xattr,usrquota 1 1
# mount -o remount /home
# mount                              # Check if usrquota is active, otherwise reboot
```

Initialize the quota.user file with `quotacheck`.

```
# quotacheck -vum /home
# chmod 644 /home/aquota.user        # To let the users check their own quota
```

Activate the quota either with the provided script (e.g. /etc/init.d/quotad on SuSE) or with `quotaon`:

```
quotaon -vu /home
```

Check that the quota is active with:

```
quota -v
```

14.2 FreeBSD setup

The quota tools are part of the base system, however the kernel needs the option quota. If it is not there, add it and recompile the kernel.

```
options QUOTA
```

As with Linux, add the quota to the fstab options (userquota, not usrquota):

```
/dev/ad0s1d    /home    ufs     rw,noatime,userquota    2  2
# mount /home                        # To remount the partition
```

Enable disk quotas in /etc/rc.conf and start the quota.

```
# grep quotas /etc/rc.conf
enable_quotas="YES"                  # turn on quotas on startup (or NO).
check_quotas="YES"                   # Check quotas on startup (or NO).
# /etc/rc.d/quota start
```

14.3 Assign quota limits

The quotas are not limited per default (set to 0). The limits are set with `edquota` for single users. A quota can be also duplicated to many users. The file structure is different between the quota implementations, but the principle is the same: the values of blocks and inodes can be limited. *Only change the values of soft and hard*. If not specified, the blocks are 1k. The grace period is set with `edquota -t`. For example:

```
# edquota -u colin
```

Linux

```
Disk quotas for user colin (uid 1007):
  Filesystem         blocks       soft       hard     inodes     soft     hard
  /dev/sda8            108       1000       2000          1        0        0
```

FreeBSD

```
Quotas for user colin:
/home: kbytes in use: 504184, limits (soft = 700000, hard = 800000)
   inodes in use: 1792, limits (soft = 0, hard = 0)
```

For many users

The command `edquota -p` is used to duplicate a quota to other users. For example to duplicate a reference quota to all users:

```
# edquota -p refuser `awk -F: '$3 > 499 {print $1}' /etc/passwd`
# edquota -p refuser user1 user2     # Duplicate to 2 users
```

Checks

Users can check their quota by simply typing `quota` (the file quota.user must be readable). Root can check all quotas.

```
# quota -u colin                     # Check quota for a user
# repquota /home                     # Full report for the partition for all users
```

15 SHELLS

Most Linux distributions use the bash shell, the bourne shell, which is also commonly used for writing scripts. Filters are very useful and can be piped:

`grep`	Pattern matching
`sed`	Search and Replace strings or characters
`cut`	Print specific columns from a marker
`sort`	Sort alphabetically or numerically
`uniq`	Remove duplicate lines from a file

For example used all at once:

```
# ifconfig | sed 's/  / /g' | cut -d" " -f1 | uniq | grep -E "[a-z0-9]+" | sort -r
# ifconfig | sed '/.*inet addr:/!d;s///;s/ .*//'|sort -t. -k1,1n -k2,2n -k3,3n -k4,4n
```

The first character in the sed pattern is a tab. To write a tab on the console, use ctrl-v ctrl-tab.

15.1 bash

Redirects and pipes for bash and sh:

```
# cmd 1> file                         # Redirect stdout to file.
# cmd 2> file                         # Redirect stderr to file.
# cmd 1>> file                        # Redirect and append stdout to file.
# cmd &> file                         # Redirect both stdout and stderr to file.
# cmd >file 2>&1                      # Redirects stderr to stdout and then to file.
# cmd1 | cmd2                         # pipe stdout to cmd2
# cmd1 2>&1 | cmd2                    # pipe stdout and stderr to cmd2
```

Modify your configuration in ~/.bashrc (it can also be ~/.bash_profile). The following entries are useful, reload with ". .bashrc". With cygwin use ~/.bash_profile; with rxvt past with shift + left-click.

```
# in .bashrc
bind '"\e[A"':history-search-backward # Use up and down arrow to search
bind '"\e[B"':history-search-forward  # the history. Invaluable!
set -o emacs                          # Set emacs mode in bash (see below)
set bell-style visible                # Do not beep, inverse colors
    # Set a nice prompt like [user@host]/path/todir>
PS1="\[\033[1;30m\][\[\033[1;34m\]\u\[\033[1;30m\]"
PS1="$PS1@\[\033[0;33m\]\h\[\033[1;30m\]]\[\033[0;37m\]"
PS1="$PS1\w\[\033[1;30m\]>\[\033[0m\]"
```

```
# To check the currently active aliases, simply type alias
alias  ls='ls -aF'                    # Append indicator (one of */=>@|)
alias  ll='ls -aFls'                  # Listing
alias  la='ls -all'
alias ..='cd ..'
alias ...='cd ../..'
export HISTFILESIZE=5000              # Larger history
export CLICOLOR=1                     # Use colors (if possible)
export LSCOLORS=ExGxFxdxCxDxDxBxBxExEx
```

16 SCRIPTING

Basics (p34) | Script example (p35) | awk (p35) | sed (p35) | Regular Expressions (p35) | useful commands (p36)

The Bourne shell (/bin/sh) is present on all Unix installations and scripts written in this language are (quite) portable; `man 1 sh` is a good reference.

16.1 Basics

Variables and arguments

Assign with variable=value and get content with $variable

```
MESSAGE="Hello World"                            # Assign a string
PI=3.1415                                        # Assign a decimal number
N=8
TWON=`expr $N * 2`                               # Arithmetic expression (only integers)
TWON=$(($N * 2))                                 # Other syntax
TWOPI=`echo "$PI * 2" | bc -l`                   # Use bc for floating point operations
ZERO=`echo "c($PI/4)-sqrt(2)/2" | bc -l`
```

The command line arguments are

```
$0, $1, $2, ...                                  # $0 is the command itself
$#                                               # The number of arguments
$*                                               # All arguments (also $@)
```

Special Variables

```
$$                                               # The current process ID
$?                                               # exit status of last command
  command
  if [ $? != 0 ]; then
    echo "command failed"
  fi
mypath=`pwd`
mypath=${mypath}/file.txt
echo ${mypath##*/}                               # Display the filename only
echo ${mypath%%.*}                               # Full path without extention
foo=/tmp/my.dir/filename.tar.gz
path = ${foo%/*}                                 # Full path without extention
var2=${var:=string}                              # Use var if set, otherwise use string
                                                 # assign string to var and then to var2.
size=$(stat -c%s "$file")                        # get file size in bourne script
filesize=${size:=-1}
```

Constructs

```
for file in `ls`
do
    echo $file
done

count=0
while [ $count -lt 5 ]; do
    echo $count
    sleep 1
    count=$(($count + 1))
done

myfunction() {
    find . -type f -name "*.$1" -print           # $1 is first argument of the function
}
myfunction "txt"
```

Generate a file

```
MYHOME=/home/colin
cat > testhome.sh << _EOF
# All of this goes into the file testhome.sh
if [ -d "$MYHOME" ] ; then
    echo $MYHOME exists
else
    echo $MYHOME does not exist
fi
_EOF
sh testhome.sh
```

16.2 Bourne script example

As a small example, the script used to create a PDF booklet from this xhtml document:

```
#!/bin/sh
# This script creates a book in pdf format ready to print on a duplex printer
if [ $# -ne 1 ]; then                         # Check the argument
  echo 1>&2 "Usage: $0 HtmlFile"
  exit 1                                      # non zero exit if error
fi

file=$1                                       # Assign the filename
fname=${file%.*}                              # Get the name of the file only
fext=${file#*.}                               # Get the extension of the file

prince $file -o $fname.pdf                    # from www.princexml.com
pdftops -paper A4 -noshrink $fname.pdf $fname.ps # create postscript booklet
cat $fname.ps |psbook|psnup -Pa4 -2 |pstops -b "2:0,1U(21cm,29.7cm)" > $fname.book.ps

ps2pdf13 -sPAPERSIZE=a4 -sAutoRotatePages=None $fname.book.ps $fname.book.pdf
                                              # use #a4 and #None on Windows!
exit 0                                        # exit 0 means successful
```

16.3 Some awk commands

Awk is useful for field stripping, like cut in a more powerful way. Search this document for other examples. See for example gnulamp.com and one-liners for awk for some nice examples.

```
awk '{ print $2, $1 }' file                   # Print and inverse first two columns
awk '{printf("%5d : %s\n", NR,$0)}' file      # Add line number left aligned
awk '{print FNR "\t" $0}' files               # Add line number right aligned
awk NF test.txt                               # remove blank lines (same as grep '.')
awk 'length > 80'                             # print line longer than 80 char)
```

16.4 Some sed commands

Here is the one liner gold mine[18]. And a good introduction and tutorial to sed[19].

```
sed 's/string1/string2/g'                     # Replace string1 with string2
sed -i 's/wroong/wrong/g' *.txt               # Replace a recurring word with g
sed 's/\(.*\)1/\12/g'                         # Modify anystring1 to anystring2
sed '/<p>/,/<\/p>/d' t.xhtml                  # Delete lines that start with <p>
                                              # and end with </p>
sed '/ *#/d; /^ *$/d'                         # Remove comments and blank lines
sed 's/[ \t]*$//'                             # Remove trailing spaces (use tab as \t)
sed 's/^[ \t]*//;s/[ \t]*$//'                 # Remove leading and trailing spaces
sed 's/[^*]/[&]/'                             # Enclose first char with [] top->[t]op
sed = file | sed 'N;s/\n/\t/' > file.num      # Number lines on a file
```

16.5 Regular Expressions

Some basic regular expression useful for sed too. See Basic Regex Syntax[20] for a good primer.

```
[\^$.|?*+()                          # special characters any other will match themselves
\                                    # escapes special characters and treat as literal
*                                    # repeat the previous item zero or more times
.                                    # single character except line break characters
.*                                   # match zero or more characters
^                                    # match at the start of a line/string
$                                    # match at the end of a line/string
.$                                   # match a single character at the end of line/string
^ $                                  # match line with a single space
^[A-Z]                               # match any line beginning with any char from A to Z
```

18. http://student.northpark.edu/pemente/sed/sed1line.txt
19. http://www.grymoire.com/Unix/Sed.html
20. http://www.regular-expressions.info/reference.html

16.6 Some useful commands

The following commands are useful to include in a script or as one liners.

```
sort -t. -k1,1n -k2,2n -k3,3n -k4,4n        # Sort IPv4 ip addresses
echo 'Test' | tr '[:lower:]' '[:upper:]'     # Case conversion
echo foo.bar | cut -d . -f 1                # Returns foo
PID=$(ps | grep script.sh | grep bin | awk '{print $1}')    # PID of a running script
PID=$(ps axww | grep [p]ing | awk '{print $1}')            # PID of ping (w/o grep pid)
IP=$(ifconfig $INTERFACE | sed '/.*inet addr:/!d;s///;s/ .*//')   # Linux
IP=$(ifconfig $INTERFACE | sed '/.*inet /!d;s///;s/ .*//')        # FreeBSD
if [ `diff file1 file2 | wc -l` != 0 ]; then [...] fi       # File changed?
cat /etc/master.passwd | grep -v root | grep -v \*: | awk -F":" \ # Create http passwd
'{ printf("%s:%s\n", $1, $2) }' > /usr/local/etc/apache2/passwd

testuser=$(cat /usr/local/etc/apache2/passwd | grep -v \    # Check user in passwd
root | grep -v \*: | awk -F":" '{ printf("%s\n", $1) }' | grep ^user$)
:(){ :|:& };:                               # bash fork bomb. Will kill your machine
tail +2 file > file2                        # remove the first line from file
```

I use this little trick to change the file extension for many files at once. For example from .cxx to .cpp. Test it first without the `| sh` at the end. You can also do this with the command `rename` if installed. Or with bash builtins.

```
# ls *.cxx | awk -F. '{print "mv "$0" "$1".cpp"}' | sh
# ls *.c | sed "s/.*/cp & &.$(date "+%Y%m%d")/" | sh # e.g. copy *.c to *.c.20080401
# rename .cxx .cpp *.cxx                             # Rename all .cxx to cpp
# for i in *.cxx; do mv $i ${i%%.cxx}.cpp; done      # with bash builtins
```

17 ONLINE HELP

17.1 Documentation

Linux Documentation	en.tldp.org
Linux Man Pages	www.linuxmanpages.com
Linux commands directory	www.oreillynet.com/linux/cmd
Linux doc man howtos	linux.die.net
FreeBSD Handbook	www.freebsd.org/handbook
FreeBSD Man Pages	www.freebsd.org/cgi/man.cgi
FreeBSD user wiki	www.freebsdwiki.net
Solaris Man Pages	docs.sun.com/app/docs/coll/40.10

17.2 Other Unix/Linux references

Rosetta Stone for Unix	bhami.com/rosetta.html (a Unix command translator)
Unix guide cross reference	unixguide.net/unixguide.shtml
Linux commands line list	www.linuxcmd.org
Short Linux reference	www.pixelbeat.org/cmdline.html
Little command line goodies	www.shell-fu.org

That's all folks!