UNIX TOOLBOX

This document is a collection of Unix/Linux/BSD commands and tasks which are useful for IT work or for advanced users. This is a practical guide with concise explanations, however the reader is supposed to know what s/he is doing.

Unix Toolbox revision 13.2
The latest version of this document can be found at http://cb.vu/unixtoolbox.xhtml. Replace .xhtml on the link with .pdf for the PDF version and with .book.pdf for the booklet version. On a duplex printer the booklet will create a small book ready to bind. See also the about page.

Error reports and comments are most welcome - c@cb.vu Colin Barschel.

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/SuSE-release                 # Get SuSE 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
# lshal                                 # Show a list of all devices with their properties
# dmidecode                             # Show DMI/SMBIOS: hw info from the BIOS
```

FreeBSD

```
# sysctl hw.model                       # CPU model
# sysctl hw                             # Gives a lot of hardware information
# sysctl vm                             # Memory usage
# dmesg | grep "real mem"               # Hardware memory
# sysctl -a | grep mem                  # Kernel memory settings and info
# sysctl dev                            # Configured devices
# pciconf -l -cv                        # Show PCI devices
# usbdevs -v                            # Show USB devices
# atacontrol list                       # Show ATA devices
# camcontrol devlist -v                 # Show SCSI devices
```

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)
# systat -vmstat 1                      # BSD summary of system statistics (1 s intervals)
# systat -tcp 1                         # BSD tcp connections (try also -ip)
```

```
# systat -netstat 1                    # BSD active network connections
# systat -ifstat 1                     # BSD network traffic through active interfaces
# systat -iostat 1                     # BSD CPU and and disk throughput
# tail -n 500 /var/log/messages        # Last 500 kernel/syslog messages
# tail /var/log/warn                   # System warnings messages see syslog.conf
```

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 admin                       # Add group "admin" and user colin (Linux/Solaris)
# useradd -c "Colin Barschel" -g admin -m colin
# usermod -a -G <group> <user>         # Add existing user to group (Debian)
# groupmod -A <user> <group>           # Add existing user to group (SuSE)
# userdel colin                        # Delete user colin (Linux/Solaris)
# adduser joe                          # FreeBSD add user joe (interactive)
# rmuser joe                           # FreeBSD delete user joe (interactive)
# pw groupadd admin                    # Use pw on FreeBSD
# pw groupmod admin -m newmember       # Add a new member to a group
# pw useradd colin -c "Colin Barschel" -g admin -m -s /bin/tcsh
# pw userdel colin; pw groupdel admin
```

Encrypted passwords are stored in /etc/shadow for Linux and Solaris and /etc/master.passwd on FreeBSD. If the master.passwd is modified manually (say to delete a password), run `# pwd_mkdb -p master.passwd` to rebuild the database.

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)
# echo "Sorry no login now" > /var/run/nologin   # (FreeBSD)
```

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
```

FreeBSD

Per shell/script

Use the command `limits` in csh or tcsh or as in Linux, use `ulimit` in an sh or bash shell.

Per user/process

The default limits on login are set in `/etc/login.conf`. An unlimited value is still limited by the system maximal value.

System wide

Kernel limits are also set with sysctl. Permanent limits are set in `/etc/sysctl.conf` or `/boot/loader.conf`. The syntax is the same as Linux but the keys are different.

```
# sysctl -a                           # View all system limits
# sysctl kern.maxfiles=XXXX           # maximum number of file descriptors
kern.ipc.nmbclusters=32768            # Permanent entry in /etc/sysctl.conf
kern.maxfiles=65536                   # Typical values for Squid
kern.maxfilesperproc=32768
kern.ipc.somaxconn=8192               # TCP queue. Better for apache/sendmail
# sysctl kern.openfiles               # How many file descriptors are in use
# sysctl kern.ipc.numopensockets      # How many open sockets are in use
# sysctl -w net.inet.ip.portrange.last=50000 # Default is 1024-5000
# netstat -m                          # network memory buffers statistics
```

See The FreeBSD handbook Chapter 11[1] for details.

Solaris

The following values in `/etc/system` will increase the maximum file descriptors per proc:

```
set rlim_fd_max = 4096                # Hard limit on file descriptors for a single proc
set rlim_fd_cur = 1024                # Soft limit on file descriptors for a single proc
```

1.5 Runlevels

Linux

Once booted, the kernel starts `init` 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:

```
# grep default: /etc/inittab
id:3:initdefault:
```

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

```
# init 5                              # Enters runlevel 5
```

0 Shutdown and halt
1 Single-User mode (also S)
2 Multi-user without network
3 Multi-user with network
5 Multi-user with X
6 Reboot

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

```
# chkconfig --list                    # List all init scripts
# chkconfig --list sshd               # Report the status of sshd
```

1.http://www.freebsd.org/handbook/configtuning-kernel-limits.html

```
# chkconfig sshd --level 35 on        # Configure sshd for levels 3 and 5
# chkconfig sshd off                  # Disable sshd for all runlevels
```

Debian and Debian based distributions like Ubuntu or Knoppix use the command `update-rc.d` to manage the runlevels scripts. Default is to start in 2,3,4 and 5 and shutdown in 0,1 and 6.

```
# update-rc.d sshd defaults           # Activate sshd with the default runlevels
# update-rc.d sshd start 20 2 3 4 5 . stop 20 0 1 6 .  # With explicit arguments
# update-rc.d -f sshd remove          # Disable sshd for all runlevels
# shutdown -h now (or # poweroff)     # Shutdown and halt the system
```

FreeBSD

The BSD boot approach is different from the SysV, there are no runlevels. The final boot state (single user, with or without X) is configured in `/etc/ttys`. All OS scripts are located in `/etc/rc.d/` and in `/usr/local/etc/rc.d/` for third-party applications. The activation of the service is configured in `/etc/rc.conf` and `/etc/rc.conf.local`. The default behavior is configured in `/etc/defaults/rc.conf`. The scripts responds at least to start|stop|status.

```
# /etc/rc.d/sshd status
sshd is running as pid 552.
# shutdown now                        # Go into single-user mode
# exit                                # Go back to multi-user mode
# shutdown -p now                     # Shutdown and halt the system
# shutdown -r now                     # Reboot
```

The process `init` can also be used to reach one of the following states level. For example `# init 6` for reboot.

0	Halt and turn the power off (signal `USR2`)
1	Go to single-user mode (signal `TERM`)
6	Reboot the machine (signal `INT`)
c	Block further logins (signal `TSTP`)
q	Rescan the ttys(5) file (signal `HUP`)

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
```

FreeBSD method 1

On FreeBSD, boot in single user mode, remount / rw and use passwd. You can select the single user mode on the boot menu (option 4) which is displayed for 10 seconds at startup. The single user mode will give you a root shell on the / partition.

```
# mount -u /; mount -a                # will mount / rw
# passwd
# reboot
```

Unixes and FreeBSD and Linux method 2

Other Unixes might not let you go away with the simple init trick. The solution is to mount the root partition from an other OS (like a rescue CD) and change the password on the disk.

- Boot a live CD or installation CD into a rescue mode which will give you a shell.
- Find the root partition with fdisk e.g. fdisk /dev/sda
- Mount it and use chroot:

```
# 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)
```

FreeBSD

```
# kldstat                              # List all modules loaded in the kernel
# kldload crypto                       # To load a module (here crypto)
```

1.8 Compile Kernel

Linux

```
# cd /usr/src/linux
# make mrproper                        # Clean everything, including config files
# make oldconfig                       # Reuse the old .config if existent
# make menuconfig                      # or xconfig (Qt) or gconfig (GTK)
# make                                 # Create a compressed kernel image
# make modules                         # Compile the modules
# make modules_install                 # Install the modules
# make install                         # Install the kernel
# reboot
```

FreeBSD

Optionally update the source tree (in `/usr/src`) with csup (as of FreeBSD 6.2 or later):

```
# csup <supfile>
```

I use the following supfile:

```
*default host=cvsup5.FreeBSD.org  # www.freebsd.org/handbook/cvsup.html#CVSUP-MIRRORS
*default prefix=/usr
*default base=/var/db
*default release=cvs delete tag=RELENG_7
src-all
```

To modify and rebuild the kernel, copy the generic configuration file to a new name and edit it as needed (you can also edit the file `GENERIC` directly). To restart the build after an interruption, add the option `NO_CLEAN=YES` to the make command to avoid cleaning the objects already build.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYKERNEL
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

To rebuild the full OS:

```
# make buildworld                      # Build the full OS but not the kernel
# make buildkernel                     # Use KERNCONF as above if appropriate
```

```
# make installkernel
# reboot
# mergemaster -p                    # Compares only files known to be essential
# make installworld
# mergemaster -i -U                 # Update all configurations and other files
# reboot
```

For small changes in the source you can use NO_CLEAN=yes to avoid rebuilding the whole tree.

```
# make buildworld NO_CLEAN=yes       # Don't delete the old objects
# make buildkernel KERNCONF=MYKERNEL NO_CLEAN=yes
```

1.9 Repair grub

So you broke grub? Boot from a live cd, [find your linux partition under `/dev` and use `fdisk` to find the linux partion] mount the linux partition, add /proc and /dev and use `grub-install /dev/xyz`. Suppose linux lies on `/dev/sda6`:

```
# mount /dev/sda6 /mnt              # mount the linux partition on /mnt
# mount --bind /proc /mnt/proc      # mount the proc subsystem into /mnt
# mount --bind /dev /mnt/dev        # mount the devices into /mnt
# chroot /mnt                       # change root to the linux partition
# grub-install /dev/sda             # reinstall grub with your old settings
```

2 PROCESSES

2.1 Listing and PIDs

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

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

However more typical usage is with a pipe or with `pgrep`:

```
# ps axww | grep cron
  586  ??  Is     0:01.48 /usr/sbin/cron -s
# 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)
# fuser -va /home                   # List processes accessing the /home partition
# strace df                         # Trace system calls and signals
# truss df                          # same as above on FreeBSD/Solaris/Unixware
# history | tail -50                # Display the last 50 used commands
```

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 which can bring a machine to its knees while

still in a lower priority. The command is only available on Linux (AFAIK). 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
# ionice -c2 -n0 firefox             # Run firefox with best effort and high priority
# ionice -c3 -p$$                    # Set the actual shell to idle priority
```

For example last command is very useful to compile (or debug) a large project. Every command launched from this shell will have a lover priority and will not disturb the system. $$ is your shell pid (try echo $$).

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`. For example start two processes, bring them in the background, list the processes with `jobs` and bring one in the foreground.

```
# ping cb.vu > 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
```

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. The program `htop` from htop.sourceforge.net is a very nice alternative and a more powerful version of top. Runs on Linux and FreeBSD (`ports/sysutils/htop/`).

```
# top
```

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.

2.5 Signals/Kill

Terminate or send a signal with `kill` or `killall`.

```
# ping -i 60 cb.vu > 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 TERM 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 or /etc/login.conf for FreeBSD. 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]))
# chmod 640 /var/log/maillog          # Restrict the log -rw-r-----
# chmod u=rw,g=r,o= /var/log/maillog # 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

```
# diskinfo -v /dev/ad2                # information about disk (sector/size) FreeBSD
# 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 Boot

FreeBSD

To boot an old kernel if the new kernel doesn't boot, stop the boot at during the count down.

```
# unload
# load kernel.old
# boot
```

3.4 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
```

3.5 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
```

FreeBSD and most Unixes

```
# fstat -f /home                     # for a mount point
# fstat -p PID                       # for an application with PID
# fstat -u user                      # for a user name
```

Find opened log file (or other opened files), say for Xorg:

```
# ps ax | grep Xorg | awk '{print $1}'
1252
# fstat -p 1252
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DV R/W
root     Xorg        1252 root /             2 drwxr-xr-x     512  r
root     Xorg        1252 text /usr     216016 -rws--x--x  1679848 r
root     Xorg        1252    0 /var     212042 -rw-r--r--   56987  w
```

The file with inum 212042 is the only file in /var:

```
# find -x /var -inum 212042
/var/log/Xorg.0.log
```

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
```

3.6 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

FreeBSD

```
# mount -v -t cd9660 /dev/cd0c /mnt  # cdrom
# mount_cd9660 /dev/wcd0c /cdrom     # other method
# mount -v -t msdos /dev/fd0c /mnt   # floppy
```

Entry in /etc/fstab:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/acd0               /cdrom          cd9660  ro,noauto       0       0
```

To let users do it:

```
# sysctl vfs.usermount=1  # Or insert the line "vfs.usermount=1" in /etc/sysctl.conf
```

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 /                      # FreeBSD
```

Copy the raw data from a cdrom into an iso image:

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

3.7 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
```

3.8 Mount an SMB share

Suppose we want to access the SMB share myshare on the computer smbserver, the address as typed on a Windows PC is \\smbserver\myshare\. We mount on /mnt/smbshare. Warning> cifs wants an IP or DNS name, not a Windows name.

Linux

```
# smbclient -U user -I 192.168.16.229 -L //smbshare/    # List the shares
# mount -t smbfs -o username=winuser //smbserver/myshare /mnt/smbshare
# mount -t cifs -o username=winuser,password=winpwd //192.168.16.229/myshare /mnt/share
```

Additionally with the package mount.cifs it is possible to store the credentials in a file, for example `/home/user/.smb`:

```
username=winuser
password=winpwd
```

And mount as follow:

```
# mount -t cifs -o credentials=/home/user/.smb //192.168.16.229/myshare /mnt/smbshare
```

FreeBSD

Use -I to give the IP (or DNS name); smbserver is the Windows name.

```
# smbutil view -I 192.168.16.229 //winuser@smbserver    # List the shares
# mount_smbfs -I 192.168.16.229 //winuser@smbserver/myshare /mnt/smbshare
```

3.9 Mount an image

Linux loop-back

```
# mount -t iso9660 -o loop file.iso /mnt                # Mount a CD image
# mount -t ext3 -o loop file.img /mnt                   # Mount an image with ext3 fs
```

FreeBSD

With memory device (do # kldload md.ko if necessary):

```
# mdconfig -a -t vnode -f file.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
# umount /mnt; mdconfig -d -u 0                         # Cleanup the md device
```

Or with virtual node:

```
# vnconfig /dev/vn0c file.iso; mount -t cd9660 /dev/vn0c /mnt
# umount /mnt; vnconfig -u /dev/vn0c                    # Cleanup the vn device
```

Solaris and FreeBSD

with loop-back file interface or lofi:

```
# lofiadm -a file.iso
# mount -F hsfs -o ro /dev/lofi/1 /mnt
# umount /mnt; lofiadm -d /dev/lofi/1                   # Cleanup the lofi device
```

3.10 Create and burn an ISO image

This will copy the cd or DVD sector for sector. Without `conv=notrunc`, the image will be smaller if there is less content on the cd. See below and the dd examples (page 41).

```
# dd if=/dev/hdc of=/tmp/mycd.iso bs=2048 conv=notrunc
```

Use mkisofs to create a CD/DVD image from files in a directory. To overcome the file names restrictions: -r enables the Rock Ridge extensions common to UNIX systems, -J enables Joliet extensions used by Microsoft systems. -L allows ISO9660 filenames to begin with a period.

```
# mkisofs -J -L -r -V TITLE -o imagefile.iso /path/to/dir
```

On FreeBSD, mkisofs is found in the ports in sysutils/cdrtools.

Burn a CD/DVD ISO image

FreeBSD

FreeBSD does not enable DMA on ATAPI drives by default. DMA is enabled with the sysctl command and the arguments below, or with /boot/loader.conf with the following entries:

```
hw.ata.ata_dma="1"
hw.ata.atapi_dma="1"
```

Use `burncd` with an ATAPI device (`burncd` is part of the base system) and `cdrecord` (in sysutils/cdrtools) with a SCSI drive.

```
# burncd -f /dev/acd0 data imagefile.iso fixate      # For ATAPI drive
# cdrecord -scanbus                  # To find the burner device (like 1,0,0)
# cdrecord dev=1,0,0 imagefile.iso
```

Linux

Also use `cdrecord` with Linux as described above. Additionally it is possible to use the native ATAPI interface which is found with:

```
# cdrecord dev=ATAPI -scanbus
```

And burn the CD/DVD as above.

dvd+rw-tools

The dvd+rw-tools package (FreeBSD: ports/sysutils/dvd+rw-tools) can do it all and includes `growisofs` to burn CDs or DVDs. The examples refer to the dvd device as `/dev/dvd` which could be a symlink to `/dev/scd0` (typical scsi on Linux) or `/dev/cd0` (typical FreeBSD) or `/dev/rcd0c` (typical NetBSD/OpenBSD character SCSI) or `/dev/rdsk/c0t1d0s2` (Solaris example of a character SCSI/ATAPI CD-ROM device). There is a nice documentation with examples on the FreeBSD handbook chapter 18.7[2].

```
                        # -dvd-compat closes the disk
# growisofs -dvd-compat -Z /dev/dvd=imagefile.iso     # Burn existing iso image
# growisofs -dvd-compat -Z /dev/dvd -J -R /p/to/data  # Burn directly
```

Convert a Nero .nrg file to .iso

Nero simply adds a 300Kb header to a normal iso image. This can be trimmed with dd.

```
# dd bs=1k if=imagefile.nrg of=imagefile.iso skip=300
```

Convert a bin/cue image to .iso

The little `bchunk` program[3] can do this. It is in the FreeBSD ports in sysutils/bchunk.

```
# bchunk imagefile.bin imagefile.cue imagefile.iso
```

3.11 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.

FreeBSD

```
# dd if=/dev/random of=/usr/vdisk.img bs=1K count=1M
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0         # Creates device /dev/md1
# bsdlabel -w /dev/md0
# newfs /dev/md0c
# mount /dev/md0c /mnt
# umount /mnt; mdconfig -d -u 0; rm /usr/vdisk.img    # Cleanup the md device
```

The file based image can be automatically mounted during boot with an entry in /etc/rc.conf and /etc/fstab. Test your setup with `# /etc/rc.d/mdconfig start` (first delete the md0 device with `# mdconfig -d -u 0`).
Note however that this automatic setup will only work if the file image is NOT on the root partition. The reason is that the /etc/rc.d/mdconfig script is executed very early during boot and the root partition is still read-only. Images located outside the root partition will be mounted later with the script /etc/rc.d/mdconfig2.
/boot/loader.conf:

```
md_load="YES"
```

/etc/rc.conf:

```
# mdconfig_md0="-t vnode -f /usr/vdisk.img"          # /usr is not on the root partition
```

/etc/fstab: (The 0 0 at the end is important, it tell fsck to ignore this device, as is does not exist yet)

```
/dev/md0                /usr/vdisk      ufs     rw              0       0
```

It is also possible to increase the size of the image afterward, say for example 300 MB larger.

```
# umount /mnt; mdconfig -d -u 0
# dd if=/dev/zero bs=1m count=300 >> /usr/vdisk.img
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0
```

2.http://www.freebsd.org/handbook/creating-dvds.html
3.http://freshmeat.net/projects/bchunk/

```
# growfs /dev/md0
# mount /dev/md0c /mnt                                  # File partition is now 300 MB larger
```

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
```

3.12 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:

FreeBSD

```
# mount_mfs -o rw -s 64M md /memdisk
# umount /memdisk; mdconfig -d -u 0                     # Cleanup the md device
md     /memdisk     mfs     rw,-s64M    0    0         # /etc/fstab entry
```

Linux

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

3.13 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

4.1 Debugging (See also Traffic analysis) (page 19)

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
```

Other OSes

```
# ifconfig fxp0               # Check the "media" field on FreeBSD
# arp -a                      # Check the router (or host) ARP entry (all OS)
# ping cb.vu                  # The first thing to try...
# traceroute cb.vu            # Print the route path to destination
# ifconfig fxp0 media 100baseTX mediaopt full-duplex # 100Mbit full duplex (FreeBSD)
# netstat -s                  # System-wide statistics for each network protocol
```

Additional commands which are not always installed per default but easy to find:

```
# arping 192.168.16.254       # Ping on ethernet layer
# tcptraceroute -f 5 cb.vu    # uses tcp instead of icmp to trace through firewalls
```

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

FreeBSD

```
# route add 212.117.0.0/16 192.168.1.1
# route delete 212.117.0.0/16
# route add default 192.168.1.1
```

Add the route permanently in /etc/rc.conf

```
static_routes="myroute"
route_myroute="-net 212.117.0.0/16 192.168.1.1"
```

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
```

Solaris

```
# route add -net 192.168.20.0 -netmask 255.255.255.0 192.168.16.254
# route add default 192.168.51.254 1                    # 1 = hops to the next gateway
# route change default 192.168.50.254 1
```

Permanent entries are set in entry in `/etc/defaultrouter`.

Windows

```
# Route add 192.168.50.0 mask 255.255.255.0 192.168.51.253
# Route add 0.0.0.0 mask 0.0.0.0 192.168.51.254
```

Use add -p to make the route persistent.

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 addr add 192.168.51.254/24 dev eth0 label eth0:1
```

FreeBSD

```
# ifconfig fxp0 inet 192.168.50.254/24                      # First IP
# ifconfig fxp0 alias 192.168.51.254 netmask 255.255.255.0 # Second IP
# ifconfig fxp0 -alias 192.168.51.254                       # Remove second IP alias
```

Permanent entries in /etc/rc.conf

```
ifconfig_fxp0="inet 192.168.50.254  netmask 255.255.255.0"
ifconfig_fxp0_alias0="192.168.51.254 netmask 255.255.255.0"
```

Solaris

Check the settings with `ifconfig -a`

```
# ifconfig hme0 plumb                                       # Enable the network card
# ifconfig hme0 192.168.50.254 netmask 255.255.255.0 up     # First IP
# ifconfig hme0:1 192.168.51.254 netmask 255.255.255.0 up   # Second IP
```

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
# ifconfig fxp0 link 00:01:02:03:04:05          # FreeBSD
# ifconfig hme0 ether 00:01:02:03:04:05         # Solaris
# sudo ifconfig en0 ether 00:01:02:03:04:05     # Mac OS X Tiger
# sudo ifconfig en0 lladdr 00:01:02:03:04:05    # Mac OS X Leopard
```

Many tools exist for Windows. For example etherchange[4]. Or look for "Mac Makeup", "smac".

4.5 Ports in use

Listening open ports:

```
# netstat -an | grep LISTEN
# lsof -i                   # Linux list all Internet connections
# socklist                  # Linux display list of open sockets
# sockstat -4               # FreeBSD application listing
# netstat -anp --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 -L -n -v                 # 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.http://ntsecurity.nu/toolbox/etherchange

FreeBSD

```
# ipfw show                          # For status
# ipfw list 65535 # if answer is "65535 deny ip from any to any" the fw is disabled
# sysctl net.inet.ip.fw.enable=0     # Disable
# sysctl net.inet.ip.fw.enable=1     # Enable
```

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
```

FreeBSD

Check and enable with:

```
# sysctl net.inet.ip.forwarding      # Check IP forward 0=off, 1=on
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet.ip.fastforwarding=1        # For dedicated router or firewall
Permanent with entry in /etc/rc.conf:
gateway_enable="YES"                 # Set to YES if this host will be a gateway.
```

Solaris

```
# ndd -set /dev/ip ip_forwarding 1   # Set IP forward 0=off, 1=on
```

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.

FreeBSD

```
# natd -s -m -u -dynamic -f /etc/natd.conf -n fxp0
Or edit /etc/rc.conf with:
firewall_enable="YES"           # Set to YES to enable firewall functionality
firewall_type="open"            # Firewall type (see /etc/rc.firewall)
natd_enable="YES"               # Enable natd (if firewall_enable == YES).
natd_interface="tun0"           # Public interface or IP address to use.
natd_flags="-s -m -u -dynamic -f /etc/natd.conf"
```

Port forward with:

```
# cat /etc/natd.conf
same_ports yes
use_sockets yes
unregistered_only
# redirect_port tcp insideIP:2300-2399 3300-3399  # port range
redirect_port udp 192.168.51.103:7777 7777
```

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
# ipconfig /flushdns                   # Flush the DNS cache
```

Forward queries

Dig is you friend to test the DNS settings. For example the public DNS server `213.133.105.2 ns.second-ns.de` 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 sun.com          # To test the local server
# dig @204.97.212.10 NS MX heise.de  # Query an external server
# 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
# 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 DHCP

Linux

Some distributions (SuSE) use dhcpcd as client. The default interface is eth0.

```
# dhcpcd -n eth0            # Trigger a renew (does not always work)
# dhcpcd -k eth0            # release and shutdown
```

The lease with the full information is stored in:

```
/var/lib/dhcpcd/dhcpcd-eth0.info
```

FreeBSD

FreeBSD (and Debian) uses dhclient. To configure an interface (for example bge0) run:

```
# dhclient bge0
```

The lease with the full information is stored in:

```
/var/db/dhclient.leases.bge0
```

Use

```
/etc/dhclient.conf
```

to prepend options or force different options:

```
# cat /etc/dhclient.conf
interface "rl0" {
    prepend domain-name-servers 127.0.0.1;
    default domain-name "sleepyowl.net";
    supersede domain-name "sleepyowl.net";
}
```

Windows

The dhcp lease can be renewed with `ipconfig`:

```
# ipconfig /renew           # renew all adapters
# ipconfig /renew LAN       # renew the adapter named "LAN"
# ipconfig /release WLAN    # release the adapter named "WLAN"
```

Yes it is a good idea to rename you adapter with simple names!

4.11 Traffic analysis

Bmon[5] 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

5.http://people.suug.ch/~tgr/bmon/

`-D` Print all interfaces available

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

Scan with nmap

Nmap[6] 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 -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.

4.12 Traffic control (QoS)

Traffic control manages the queuing, policing, scheduling, and other traffic parameters for a network. The following examples are simple practical uses of the Linux and FreeBSD capabilities to better use the available bandwidth.

Limit upload

DSL or cable modems have a long queue to improve the upload throughput. However filling the queue with a fast device (e.g. ethernet) will dramatically decrease the interactivity. It is therefore useful to limit the device upload rate to match the physical capacity of the modem, this should greatly improve the interactivity. Set to about 90% of the modem maximal (cable) speed.

Linux

For a 512 Kbit upload modem.

```
# tc qdisc add dev eth0 root tbf rate 480kbit latency 50ms burst 1540
# tc -s qdisc ls dev eth0                          # Status
# tc qdisc del dev eth0 root                       # Delete the queue
# tc qdisc change dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

FreeBSD

FreeBSD uses the `dummynet` traffic shaper which is configured with ipfw. Pipes are used to set limits the bandwidth in units of [K|M]{bit/s|Byte/s}, 0 means unlimited bandwidth. Using the same pipe number will reconfigure it. For example limit the upload bandwidth to 500 Kbit.

```
# kldload dummynet                                 # load the module if necessary
# ipfw pipe 1 config bw 500Kbit/s                  # create a pipe with limited bandwidth
# ipfw add pipe 1 ip from me to any                # divert the full upload into the pipe
```

Quality of service

Linux

Priority queuing with `tc` to optimize VoIP. See the full example on voip-info.org or www.howtoforge.com. Suppose VoIP uses udp on ports 10000:11024 and device eth0 (could also be ppp0 or so). The following commands define the QoS to three queues and force the VoIP traffic to queue 1 with QoS `0x1e` (all bits set). The default traffic flows into queue 3 and QoS *Minimize-Delay* flows into queue 2.

6.http://insecure.org/nmap/

```
# tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0
# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
# tc qdisc add dev eth0 parent 1:2 handle 20: sfq
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
# tc filter add dev eth0 protocol ip parent 1: prio 1 u32 \
  match ip dport 10000 0x3C00 flowid 1:1          # use server port range
  match ip dst 123.23.0.1 flowid 1:1              # or/and use server IP
```

Status and remove with

```
# tc -s qdisc ls dev eth0                         # queue status
# tc qdisc del dev eth0 root                      # delete all QoS
```

Calculate port range and mask

The tc filter defines the port range with port and mask which you have to calculate. Find the 2^N *ending* of the port range, deduce the range and convert to HEX. This is your mask. Example for 10000 -> 11024, the range is 1024.

```
# 2^13 (8192) < 10000 < 2^14 (16384)              # ending is 2^14 = 16384
# echo "obase=16;(2^14)-1024" | bc                # mask is 0x3C00
```

FreeBSD

The max link bandwidth is 500Kbit/s and we define 3 queues with priority 100:10:1 for VoIP:ssh:all the rest.

```
# ipfw pipe 1 config bw 500Kbit/s
# ipfw queue 1 config pipe 1 weight 100
# ipfw queue 2 config pipe 1 weight 10
# ipfw queue 3 config pipe 1 weight 1
# ipfw add 10 queue 1 proto udp dst-port 10000-11024
# ipfw add 11 queue 1 proto udp dst-ip 123.23.0.1 # or/and use server IP
# ipfw add 20 queue 2 dsp-port ssh
# ipfw add 30 queue 3 from me to any              # all the rest
```

Status and remove with

```
# ipfw list                                       # rules status
# ipfw pipe list                                  # pipe status
# ipfw flush                                      # deletes all rules but default
```

4.13 NIS Debugging

Some commands which should work on a well configured NIS client:

```
# ypwhich                  # get the connected NIS server name
# domainname               # The NIS domain name as configured
# ypcat group              # should display the group from the NIS server
# cd /var/yp && make       # Rebuild the yp database
```

Is ypbind running?

```
# ps auxww | grep ypbind
/usr/sbin/ypbind -s -m -S servername1,servername2        # FreeBSD
/usr/sbin/ypbind           # Linux
# yppoll passwd.byname
Map passwd.byname has order number 1190635041. Mon Sep 24 13:57:21 2007
The master server is servername.domain.net.
```

Linux

```
# cat /etc/yp.conf
ypserver servername
domain domain.net broadcast
```

4.14 Netcat

Netcat[7] (nc) is better known as the "network Swiss Army Knife", it can manipulate, create or read/write TCP/IP connections. Here some useful examples, there are many more on the net, for example g-loaded.eu[...][8] and here[9].
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 SCP

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"
```

7.http://netcat.sourceforge.net
8.http://www.g-loaded.eu/2006/11/06/netcat-a-couple-of-useful-examples
9.http://www.terminally-incoherent.com/blog/2007/08/07/few-useful-netcat-tricks

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/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[10] 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
```

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[11].

10.http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

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
# 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?

11.http://fuse.sourceforge.net/sshfs.html

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
```

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 17)
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 17)
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/
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # use relative (see below)
```

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/
```

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[12]. 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.

```
@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:

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

```
# 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                 # 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             # 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[13] and GPG/PGP Basics[14] and the gnupg documentation[15] 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/.

```
~/.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.

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

```
# 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 ENCRYPT PARTITIONS

Linux with LUKS (p31) | Linux dm-crypt only (p32) | FreeBSD GELI (p32) | FBSD pwd only (p33)

There are (many) other alternative methods to encrypt disks, I only show here the methods I know and use. Keep in mind that the security is only good as long the OS has not been tempered with. An intruder could easily record the password from the keyboard events. Furthermore the data is freely accessible when the partition is *attached* and will not prevent an intruder to have access to it in this state.

10.1 Linux

Those instructions use the Linux `dm-crypt` (device-mapper) facility available on the 2.6 kernel. In this example, lets encrypt the partition `/dev/sdc1`, it could be however any other partition or disk, or USB or a file based partition created with `losetup`. In this case we would use `/dev/loop0`. See file image partition. The device mapper uses labels to identify a partition. We use `sdc1` in this example, but it could be any string.

dm-crypt with LUKS

LUKS with dm-crypt has better encryption and makes it possible to have multiple passphrase for the same partition or to change the password easily. To test if LUKS is available, simply type `# cryptsetup --help`, if nothing about LUKS shows up, use the instructions below Without LUKS. First create a partition if necessary: `fdisk /dev/sdc`.

Create encrypted partition

```
# dd if=/dev/urandom of=/dev/sdc1           # Optional. For paranoids only (takes days)
# cryptsetup -y luksFormat /dev/sdc1        # This destroys any data on sdc1
# cryptsetup luksOpen /dev/sdc1 sdc1
# mkfs.ext3 /dev/mapper/sdc1                # create ext3 file system
```

```
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt
# cryptsetup luksClose sdc1                # Detach the encrypted partition
```

Attach

```
# cryptsetup luksOpen /dev/sdc1 sdc1
# mount -t ext3 /dev/mapper/sdc1 /mnt
```

Detach

```
# umount /mnt
# cryptsetup luksClose sdc1
```

dm-crypt without LUKS

```
# cryptsetup -y create sdc1 /dev/sdc1      # or any other partition like /dev/loop0
# dmsetup ls                               # check it, will display: sdc1 (254, 0)
# mkfs.ext3 /dev/mapper/sdc1               # This is done only the first time!
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt/
# cryptsetup remove sdc1                   # Detach the encrypted partition
```

Do exactly the same (without the mkfs part!) to re-attach the partition. If the password is not correct, the mount command will fail. In this case simply remove the map sdc1 (`cryptsetup remove sdc1`) and create it again.

10.2 FreeBSD

The two popular FreeBSD disk encryption modules are `gbde` and `geli`. I now use geli because it is faster and also uses the crypto device for hardware acceleration. See The FreeBSD handbook Chapter 18.6[16] for all the details. The geli module must be loaded or compiled into the kernel:

```
options GEOM_ELI
device crypto                                    # or as module:
# echo 'geom_eli_load="YES"' >> /boot/loader.conf   # or do: kldload geom_eli
```

Use password and key

I use those settings for a typical disk encryption, it uses a passphrase AND a key to encrypt the master key. That is you need both the password and the generated key `/root/ad1.key` to attach the partition. The master key is stored inside the partition and is not visible. See below for typical USB or file based image.

Create encrypted partition

```
# dd if=/dev/random of=/root/ad1.key bs=64 count=1  # this key encrypts the mater key
# geli init -s 4096 -K /root/ad1.key /dev/ad1       # -s 8192 is also OK for disks
# geli attach -k /root/ad1.key /dev/ad1             # DO make a backup of /root/ad1.key
# dd if=/dev/random of=/dev/ad1.eli bs=1m           # Optional and takes a long time
# newfs /dev/ad1.eli                                # Create file system
# mount /dev/ad1.eli /mnt
```

Attach

```
# geli attach -k /root/ad1.key /dev/ad1
# fsck -ny -t ffs /dev/ad1.eli                      # In doubt check the file system
# mount /dev/ad1.eli /mnt
```

Detach

The detach procedure is done automatically on shutdown.

```
# umount /mnt
# geli detach /dev/ad1.eli
```

16.http://www.freebsd.org/handbook/disks-encrypting.html

/etc/fstab

The encrypted partition can be configured to be mounted with /etc/fstab. The password will be prompted when booting. The following settings are required for this example:

```
# grep geli /etc/rc.conf
geli_devices="ad1"
geli_ad1_flags="-k /root/ad1.key"
# grep geli /etc/fstab
/dev/ad1.eli         /home/private              ufs             rw      0       0
```

Use password only

It is more convenient to encrypt a USB stick or file based image with a passphrase only and no key. In this case it is not necessary to carry the additional key file around. The procedure is very much the same as above, simply without the key file. Let's encrypt a file based image `/cryptedfile` of 1 GB.

```
# dd if=/dev/zero of=/cryptedfile bs=1M count=1000  # 1 GB file
# mdconfig -at vnode -f /cryptedfile
# geli init /dev/md0                                # encrypts with password only
# geli attach /dev/md0
# newfs -U -m 0 /dev/md0.eli
# mount /dev/md0.eli /mnt
# umount /dev/md0.eli
# geli detach md0.eli
```

It is now possible to mount this image on an other system with the password only.

```
# mdconfig -at vnode -f /cryptedfile
# geli attach /dev/md0
# mount /dev/md0.eli /mnt
```

11 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).

11.1 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.).

11.2 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).

11.3 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
```

11.4 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`.

11.5 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.

11.6 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!

11.7 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
```

12 CVS

Server setup (p35) | CVS test (p36) | SSH tunneling (p37) | CVS usage (p37)

12.1 Server setup

Initiate the CVS

Decide where the main repository will rest and create a root cvs. For example /usr/local/cvs (as root):

```
# mkdir -p /usr/local/cvs
# setenv CVSROOT /usr/local/cvs      # Set CVSROOT to the new location (local)
# cvs init                           # Creates all internal CVS config files
# cd /root
# cvs checkout CVSROOT               # Checkout the config files to modify them
# cd CVSROOT
edit config ( fine as it is)
# cvs commit config
cat >> writers                       # Create a writers file (optionally also readers)
colin
^D                                   # Use [Control][D] to quit the edit
# cvs add writers                    # Add the file writers into the repository
# cvs edit checkoutlist
# cat >> checkoutlist
writers
^D                                   # Use [Control][D] to quit the edit
# cvs commit                         # Commit all the configuration changes
```

Add a **readers** file if you want to differentiate read and write permissions *Note:* Do not (ever) edit files directly into the main cvs, but rather checkout the file, modify it and check it in. We did this with the file **writers** to define the write access.
There are three popular ways to access the CVS at this point. The first two don't need any further configuration. See the examples on CVSROOT below for how to use them:

- Direct local access to the file system. The user(s) need sufficient file permission to access the CS directly and there is no further authentication in addition to the OS login. However this is only useful if the repository is local.
- Remote access with ssh with the ext protocol. Any use with an ssh shell account and read/write permissions on the CVS server can access the CVS directly with ext over ssh without any additional tunnel. There is no server process running on the CVS for this to work. The ssh login does the authentication.
- Remote access with pserver. This is the preferred use for larger user base as the users are authenticated by the CVS pserver with a dedicated password database, there is therefore no need for local users accounts. This setup is explained below.

Network setup with inetd

The CVS can be run locally only if a network access is not needed. For a remote access, the daemon inetd can start the pserver with the following line in /etc/inetd.conf (/etc/xinetd.d/cvs on SuSE):

```
cvspserver        stream  tcp  nowait  cvs  /usr/bin/cvs        cvs \
--allow-root=/usr/local/cvs pserver
```

It is a good idea to block the cvs port from the Internet with the firewall and use an ssh tunnel to access the repository remotely.

Separate authentication

It is possible to have cvs users which are not part of the OS (no local users). This is actually probably wanted too from the security point of view. Simply add a file named **passwd** (in the CVSROOT directory) containing the users login and password in the crypt format. This is can be done with the apache htpasswd tool.
Note: This passwd file is the only file which has to be edited directly in the CVSROOT directory. Also it won't be checked out. More info with htpasswd --help

```
# htpasswd -cb passwd user1 password1  # -c creates the file
# htpasswd -b passwd user2 password2
```

Now add `:cvs` at the end of each line to tell the cvs server to change the user to cvs (or whatever your cvs server is running under). It looks like this:

```
# cat passwd
user1:xsFjhU22u8Fuo:cvs
user2:vnefJOsnnvToM:cvs
```

12.2 Test it

Test the login as normal user (for example here me)

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs login
Logging in to :pserver:colin@192.168.50.254:2401/usr/local/cvs
CVS password:
```

CVSROOT variable

This is an environment variable used to specify the location of the repository we're doing operations on. For local use, it can be just set to the directory of the repository. For use over the network, the transport protocol must be specified. Set the CVSROOT variable with `setenv CVSROOT string` on a csh, tcsh shell, or with `export CVSROOT=string` on a sh, bash shell.

```
# setenv CVSROOT :pserver:<username>@<host>:/cvsdirectory
For example:
# setenv CVSROOT /usr/local/cvs                                  # Used locally only
# setenv CVSROOT :local:/usr/local/cvs                           # Same as above
# setenv CVSROOT :ext:user@cvsserver:/usr/local/cvs              # Direct access with SSH
# setenv CVS_RSH ssh                                             # for the ext access
# setenv CVSROOT :pserver:user@cvsserver.254:/usr/local/cvs      # network with pserver
```

When the login succeeded one can import a new project into the repository: **cd into** your project root directory

```
cvs import <module name> <vendor tag> <initial tag>
cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs import MyProject MyCompany START
```

Where MyProject is the name of the new project in the repository (used later to checkout). Cvs will import the current directory content into the new project.

To checkout:

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs checkout MyProject
or
```

```
# setenv CVSROOT :pserver:colin@192.168.50.254:/usr/local/cvs
# cvs checkout MyProject
```

12.3 SSH tunneling for CVS

We need 2 shells for this. On the first shell we connect to the cvs server with ssh and port-forward the cvs connection. On the second shell we use the cvs normally as if it where running locally.
on shell 1:

```
# ssh -L2401:localhost:2401 colin@cvs_server   # Connect directly to the CVS server. Or:
# ssh -L2401:cvs_server:2401 colin@gateway     # Use a gateway to reach the CVS
```

on shell 2:

```
# setenv CVSROOT :pserver:colin@localhost:/usr/local/cvs
# cvs login
Logging in to :pserver:colin@localhost:2401/usr/local/cvs
CVS password:
# cvs checkout MyProject/src
```

12.4 CVS commands and usage

Import

The import command is used to add a whole directory, it must be run from within the directory to be imported. Say the directory /devel/ contains all files and subdirectories to be imported. The directory name on the CVS (the module) will be called "myapp".

```
# cvs import [options] directory-name vendor-tag release-tag
# cd /devel                          # Must be inside the project to import it
# cvs import myapp Company R1_0      # Release tag can be anything in one word
```

After a while a new directory "/devel/tools/" was added and it has to be imported too.

```
# cd /devel/tools
# cvs import myapp/tools Company R1_0
```

Checkout update add commit

```
# cvs co myapp/tools                 # Will only checkout the directory tools
# cvs co -r R1_1 myapp               # Checkout myapp at release R1_1 (is sticky)
# cvs -q -d update -P                # A typical CVS update
# cvs update -A                      # Reset any sticky tag (or date, option)
# cvs add newfile                    # Add a new file
# cvs add -kb newfile                # Add a new binary file
# cvs commit file1 file2             # Commit the two files only
# cvs commit -m "message"            # Commit all changes done with a message
```

Create a patch

It is best to create and apply a patch from the working development directory related to the project, or from within the source directory.

```
# cd /devel/project
# diff -Naur olddir newdir > patchfile # Create a patch from a directory or a file
# diff -Naur oldfile newfile > patchfile
```

Apply a patch

Sometimes it is necessary to strip a directory level from the patch, depending how it was created. In case of difficulties, simply look at the first lines of the patch and try -p0, -p1 or -p2.

```
# cd /devel/project
# patch --dry-run -p0 < patchfile    # Test the path without applying it
# patch -p0 < patchfile
# patch -p1 < patchfile              # strip off the 1st level from the path
```

13 SVN

Server setup (p38) | SVN+SSH (p38) | SVN over http (p38) | SVN usage (p39)

Subversion (SVN)[17] is a version control system designed to be the successor of CVS (Concurrent Versions System). The concept is similar to CVS, but many shortcomings where improved. See also the SVN book[18].

13.1 Server setup

The initiation of the repository is fairly simple (here for example `/home/svn/` must exist):

```
# svnadmin create --fs-type fsfs /home/svn/project1
```

Now the access to the repository is made possible with:

- `file://` Direct file system access with the svn client with. This requires local permissions on the file system.
- `svn://` or `svn+ssh://` Remote access with the svnserve server (also over SSH). This requires local permissions on the file system.
- `http://` Remote access with webdav using apache. No local users are necessary for this method.

Using the local file system, it is now possible to import and then check out an existing project. Unlike with CVS it is not necessary to cd into the project directory, simply give the full path:

```
# svn import /project1/ file:///home/svn/project1/trunk -m 'Initial import'
# svn checkout file:///home/svn/project1
```

The new directory "trunk" is only a convention, this is not required.

Remote access with ssh

No special setup is required to access the repository via ssh, simply replace `file://` with `svn+ssh/hostname`. For example:

```
# svn checkout svn+ssh://hostname/home/svn/project1
```

As with the local file access, every user needs an ssh access to the server (with a local account) and also read/write access. This method might be suitable for a small group. All users could belong to a subversion group which owns the repository, for example:

```
# groupadd subversion
# groupmod -A user1 subversion
# chown -R root:subversion /home/svn
# chmod -R 770 /home/svn
```

Remote access with http (apache)

Remote access over http (https) is the only good solution for a larger user group. This method uses the apache authentication, not the local accounts. This is a typical but small apache configuration:

```
LoadModule dav_module         modules/mod_dav.so
LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so    # Only for access control
```

17.http://subversion.tigris.org/
18.http://svnbook.red-bean.com/en/1.4/

```
<Location /svn>
  DAV svn
  # any "/svn/foo" URL will map to a repository /home/svn/foo
  SVNParentPath /home/svn
  AuthType Basic
  AuthName "Subversion repository"
  AuthzSVNAccessFile /etc/apache2/svn.acl
  AuthUserFile /etc/apache2/svn-passwd
  Require valid-user
</Location>
```

The apache server needs full access to the repository:

```
# chown -R www:www /home/svn
```

Create a user with htpasswd2:

```
# htpasswd -c /etc/svn-passwd user1  # -c creates the file
```

Access control svn.acl example

```
# Default it read access. "* =" would be default no access
[/]
* = r
[groups]
project1-developers = joe, jack, jane
# Give write access to the developers
[project1:]
@project1-developers = rw
```

13.2 SVN commands and usage

See also the Subversion Quick Reference Card[19]. Tortoise SVN[20] is a nice Windows interface.

Import

A new project, that is a directory with some files, is imported into the repository with the `import` command. Import is also used to add a directory with its content to an existing project.

```
# svn help import                                  # Get help for any command
    # Add a new directory (with content) into the src dir on project1
# svn import /project1/newdir http://host.url/svn/project1/trunk/src -m 'add newdir'
```

Typical SVN commands

```
# svn co http://host.url/svn/project1/trunk      # Checkout the most recent version
    # Tags and branches are created by copying
# svn mkdir http://host.url/svn/project1/tags/   # Create the tags directory
# svn copy -m "Tag rc1 rel." http://host.url/svn/project1/trunk \
                             http://host.url/svn/project1/tags/1.0rc1
# svn status [--verbose]                         # Check files status into working dir
# svn add src/file.h src/file.cpp                # Add two files
# svn commit -m 'Added new class file'           # Commit the changes with a message
# svn ls http://host.url/svn/project1/tags/      # List all tags
# svn move foo.c bar.c                           # Move (rename) files
# svn delete some_old_file                       # Delete files
```

19.http://www.cs.put.poznan.pl/csobaniec/Papers/svn-refcard.pdf
20.http://tortoisesvn.tigris.org

14 USEFUL COMMANDS

14.1 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

14.2 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 text

dd delete current line
D Delete to the end of the line
dw Delete word
x Delete character
u Undo last
U Undo all changes to current line

14.3 mail

The `mail` command is a basic application to read and send email, it is usually installed. To send an email simply type "mail user@domain". The first line is the subject, then the mail content. Terminate and send the email with a single dot (.) in a new line. Example:

```
# mail c@cb.vu
Subject: Your text is full of typos
"For a moment, nothing happened. Then, after a second or so,
nothing continued to happen."
.
EOT
#
```

This is also working with a pipe:

```
# echo "This is the mail body" | mail c@cb.vu
```

This is also a simple way to test the mail server.

14.4 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
# tar -xjf home.tbz             # same with bzip2 compression
# tar -xjf home.tbz home/colin/file.txt    # Restore a single file
```

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/
```

14.5 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>
```

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.
```

Recover

The command `dd` will read *every single block* of the partition, even the blocks. 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 13)
# 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
```

14.6 screen

Screen 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

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

14.7 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 {} \;
```

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[21].

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

```
# 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
```

14.8 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)
# 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
```

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
```

15 INSTALL SOFTWARE

15.1 List installed packages

```
# rpm -qa                            # List installed packages (RH, SuSE, RPM based)
# dpkg -l                            # Debian, Ubuntu
# pkg_info                           # FreeBSD list all installed packages
# pkg_info -W smbd                   # FreeBSD show which package smbd belongs to
# pkginfo                            # Solaris
```

15.2 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
```

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
```

Gentoo

Gentoo uses emerge as the heart of its "Portage" package management system.

```
# emerge --sync                         # First sync the local portage tree
# emerge -u packagename                 # Install or upgrade a package
# emerge -C packagename                 # Remove the package
# revdep-rebuild                        # Repair dependencies
```

Solaris

The <cdrom> path is usually `/cdrom/cdrom0`.

```
# pkgadd -d <cdrom>/Solaris_9/Product SUNWgtar
# pkgadd -d SUNWgtar                    # Add downloaded package (bunzip2 first)
# pkgrm SUNWgtar                        # Remove the package
```

FreeBSD

```
# pkg_add -r rsync                      # Fetch and install rsync.
# pkg_delete /var/db/pkg/rsync-xx       # Delete the rsync package
```

Set where the packages are fetched from with the `PACKAGESITE` variable. For example:

```
# export PACKAGESITE=ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages/Latest/
# or ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6-stable/Latest/
```

FreeBSD ports

The port tree `/usr/ports/` is a collection of software ready to compile and install. The ports are updated with the program `portsnap`.

```
# portsnap fetch extract                # Create the tree when running the first time
# portsnap fetch update                 # Update the port tree
# cd /usr/ports/net/rsync/              # Select the package to install
# make install distclean                # Install and cleanup (also see man ports)
# make package                          # Make a binary package for the port
```

15.3 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
# ldconfig -n /path/to/libs/            # Add a path to the shared libraries directories
# ldconfig -m /path/to/libs/            # FreeBSD
# LD_LIBRARY_PATH                       # The variable set the link library path
```

16 CONVERT MEDIA

Sometimes one simply need to convert a video, audio file or document to another format.

16.1 Text encoding

Text encoding can get totally wrong, specially when the language requires special characters like àäç. The command `iconv` can convert from one encoding to an other.

```
# iconv -f <from_encoding> -t <to_encoding> <input_file>
# iconv -f ISO8859-1 -t UTF-8 -o file.input > file_utf8
# iconv -l                              # List known coded character sets
```

Without the -f option, iconv will use the local char-set, which is usually fine if the document displays well.

16.2 Unix - DOS newlines

Convert DOS (CR/LF) to Unix (LF) newlines and back **within a Unix shell**. See also `dos2unix` and `unix2dos` if you have them.

```
# sed 's/.$//' dosfile.txt > unixfile.txt                  # DOS to UNIX
# awk '{sub(/\r$/,"");print}' dosfile.txt > unixfile.txt   # DOS to UNIX
# awk '{sub(/$/,"\r");print}' unixfile.txt > dosfile.txt   # UNIX to DOS
```

Convert Unix to DOS newlines **within a Windows environment**. Use sed or awk from mingw or cygwin.

```
# sed -n p unixfile.txt > dosfile.txt
# awk 1 unixfile.txt > dosfile.txt   # UNIX to DOS (with a cygwin shell)
```

16.3 PDF to Jpeg and concatenate PDF files

Convert a PDF document with `gs` (GhostScript) to jpeg (or png) images for each page. Also much shorter with `convert` (from ImageMagick or GraphicsMagick).

```
# gs -dBATCH -dNOPAUSE -sDEVICE=jpeg -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 \
 -dMaxStripSize=8192 -sOutputFile=unixtoolbox_%d.jpg unixtoolbox.pdf
# convert unixtoolbox.pdf unixtoolbox-%03d.png
# convert *.jpeg images.pdf          # Create a simple PDF with all pictures
```

Ghostscript can also concatenate multiple pdf files into a single one. This only works well if the PDF files are "well behaved".

```
# gs -q -sPAPERSIZE=a4 -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=all.pdf \
file1.pdf file2.pdf ...              # On Windows use '#' instead of '='
```

16.4 Convert video

Compress the Canon digicam video with an mpeg4 codec and repair the crappy sound.

```
# mencoder -o videoout.avi -oac mp3lame -ovc lavc -srate 11025 \
-channels 1 -af-adv force=1 -lameopts preset=medium -lavcopts \
vcodec=msmpeg4v2:vbitrate=600 -mc 0 vidoein.AVI
```

See sox for sound processing.

16.5 Copy an audio cd

The program `cdparanoia`[22] can save the audio tracks (FreeBSD port in audio/cdparanoia/), `oggenc` can encode in Ogg Vorbis format, `lame` converts to mp3.

```
# cdparanoia -B                      # Copy the tracks to wav files in current dir
# lame -b 256 in.wav out.mp3         # Encode in mp3 256 kb/s
# for i in *.wav; do lame -b 256 $i `basename $i .wav`.mp3; done
# oggenc in.wav -b 256 out.ogg       # Encode in Ogg Vorbis 256 kb/s
```

17 PRINTING

17.1 Print with lpr

```
# lpr unixtoolbox.ps                 # Print on default printer
# export PRINTER=hp4600              # Change the default printer
# lpr -Php4500 #2 unixtoolbox.ps     # Use printer hp4500 and print 2 copies
# lpr -o Duplex=DuplexNoTumble ...   # Print duplex along the long side
# lpr -o PageSize=A4,Duplex=DuplexNoTumble ...
```

```
# lpq                                # Check the queue on default printer
# lpq -l -Php4500                    # Queue on printer hp4500 with verbose
```

22.http://xiph.org/paranoia/

```
# lprm -                                # Remove all users jobs on default printer
# lprm -Php4500 3186                    # Remove job 3186. Find job nbr with lpq
# lpc status                            # List all available printers
# lpc status hp4500                     # Check if printer is online and queue length
```

Some devices are not postscript and will print garbage when fed with a pdf file. This might be solved with:

```
# gs -dSAFER -dNOPAUSE -sDEVICE=deskjet -sOutputFile=\|lpr file.pdf
```

18 DATABASES

18.1 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.

18.2 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

```
# mysql -u root mysql
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. 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.

18.3 SQLite

SQLite[23] 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
```

23.http://www.sqlite.org

19 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.

19.1 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 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
```

19.2 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
```

19.3 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
```

20 SHELLS

Most Linux distributions use the bash shell while the BSDs use tcsh, the bourne shell is only used for 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.

20.1 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".

```
# 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
```

20.2 tcsh

Redirects and pipes for tcsh and csh (simple > and >> are the same as sh):

```
# cmd >& file                           # Redirect both stdout and stderr to file.
# cmd >>& file                          # Append both stdout and stderr to file.
# cmd1 | cmd2                           # pipe stdout to cmd2
# cmd1 |& cmd2                          # pipe stdout and stderr to cmd2
```

The settings for csh/tcsh are set in `~/.cshrc`, reload with "source .cshrc". Examples:

```
# in .cshrc
alias  ls      'ls -aF'
alias  ll      'ls -aFls'
alias  la      'ls -all'
alias  ..      'cd ..'
alias  ...     'cd ../..'
set   prompt    = "%B%n%b@%B%m%b%/> " # like user@host/path/todir>
set   history   =  5000
set   savehist  = ( 6000 merge )
set   autolist                        # Report possible completions with tab
set   visiblebell                     # Do not beep, inverse colors
```

```
# Bindkey and colors
bindkey -e     Select Emacs bindings  # Use emacs keys to edit the command prompt
bindkey -k up history-search-backward # Use up and down arrow to search
bindkey -k down history-search-forward
setenv CLICOLOR 1                     # Use colors (if possible)
setenv LSCOLORS ExGxFxdxCxDxDxBxBxExEx
```

The emacs mode enables to use the emacs keys shortcuts to modify the command prompt line. This is extremely useful (not only for emacs users). The most used commands are:

C-a	Move cursor to beginning of line
C-e	Move cursor to end of line
M-b	Move cursor back one word
M-f	Move cursor forward one word
M-d	Cut the next word
C-w	Cut the last word
C-u	Cut everything before the cursor
C-k	Cut everything after the cursor (rest of the line)
C-y	Paste the last thing to be cut (simply paste)
C-_	Undo

Note: C- = hold control, M- = hold meta (which is usually the alt or escape key).

21 SCRIPTING

Basics (p52) | Script example (p53) | awk (p53) | sed (p53) | Regular Expressions (p53) | useful commands (p54)

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.

21.1 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 ${mypath%%.*}                             # Full path without extention
var2=${var:=string}                            # Use var if set, otherwise use string
                                               # assign string to var and then to var2.
```

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
```

21.2 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
```

21.3 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)
```

21.4 Some sed commands

Here is the one liner gold mine[24]. And a good introduction and tutorial to sed[25].

```
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
```

21.5 Regular Expressions

Some basic regular expression useful for sed too. See Basic Regex Syntax[26] 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
```

24.http://student.northpark.edu/pemente/sed/sed1line.txt
25.http://www.grymoire.com/Unix/Sed.html
26.http://www.regular-expressions.info/reference.html

```
^ $                                     # match line with a single space
[^A-Z]                                  # match any line beginning with any char from A to Z
```

21.6 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
```

22 PROGRAMMING

22.1 C basics

```
strcpy(newstr,str)                      /* copy str to newstr */
expr1 ? expr2 : expr3                   /* if (expr1) expr2 else expr3 */
x = (y > z) ? y : z;                    /* if (y > z) x = y; else x = z; */
int a[]={0,1,2};                        /* Initialized array (or a[3]={0,1,2}; */
int a[2][3]={{1,2,3},{4,5,6}};          /* Array of array of ints */
int i = 12345;                          /* Convert in i to char str */
char str[10];
sprintf(str, "%d", i);
```

22.2 C example

A minimal c program simple.c:

```
#include <stdio.h>
main() {
    int number=42;
    printf("The answer is %i\n", number);
}
```

Compile with:

```
# gcc simple.c -o simple
# ./simple
The answer is 42
```

22.3 C++ basics

```
*pointer                                    // Object pointed to by pointer
&obj                                        // Address of object obj
obj.x                                       // Member x of class obj (object obj)
pobj->x                                     // Member x of class pointed to by pobj
                                            // (*pobj).x and pobj->x are the same
```

22.4 C++ example

As a slightly more realistic program in C++: a class in its own header (IPv4.h) and implementation (IPv4.cpp) and a program which uses the class functionality. The class converts an IP address in integer format to the known quad format.

IPv4 class

IPv4.h:

```
#ifndef IPV4_H
#define IPV4_H
#include <string>

namespace GenericUtils {                            // create a namespace
class IPv4 {                                        // class definition
public:
    IPv4(); ~IPv4();
    std::string IPint_to_IPquad(unsigned long ip);// member interface
};
} //namespace GenericUtils
#endif // IPV4_H
```

IPv4.cpp:

```
#include "IPv4.h"
#include <string>
#include <sstream>
using namespace std;                                // use the namespaces
using namespace GenericUtils;

IPv4::IPv4() {}                                     // default constructor/destructor
IPv4::~IPv4() {}
string IPv4::IPint_to_IPquad(unsigned long ip) {    // member implementation
    ostringstream ipstr;                            // use a stringstream
    ipstr << ((ip &0xff000000) >> 24)               // Bitwise right shift
          << "." << ((ip &0x00ff0000) >> 16)
          << "." << ((ip &0x0000ff00) >> 8)
          << "." << ((ip &0x000000ff));
    return ipstr.str();
}
```

The program simplecpp.cpp

```
#include "IPv4.h"
#include <iostream>
#include <string>
using namespace std;
int main (int argc, char* argv[]) {
    string ipstr;                                   // define variables
    unsigned long ipint = 1347861486;               // The IP in integer form
    GenericUtils::IPv4 iputils;                     // create an object of the class
    ipstr = iputils.IPint_to_IPquad(ipint);         // call the class member
    cout << ipint << " = " << ipstr << endl;        // print the result

    return 0;
}
```

Compile and execute with:

```
# g++ -c IPv4.cpp simplecpp.cpp                # Compile in objects
# g++ IPv4.o simplecpp.o -o simplecpp.exe      # Link the objects to final executable
# ./simplecpp.exe
1347861486 = 80.86.187.238
```

Use `ldd` to check which libraries are used by the executable and where they are located. Also used to check if a shared library is missing or if the executable is static.

```
# ldd /sbin/ifconfig
```

22.5 Simple Makefile

The minimal Makefile for the multi-source program is shown below. The lines with instructions *must begin with a tab*! The back slash "\" can be used to cut long lines.

```
CC = g++
CFLAGS = -O
OBJS = IPv4.o simplecpp.o

simplecpp: ${OBJS}
        ${CC} -o simplecpp ${CFLAGS} ${OBJS}
clean:
        rm -f ${TARGET} ${OBJS}
```

23 ONLINE HELP

23.1 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

23.2 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.linuxguide.it/commands_list.php
Short Linux reference	www.pixelbeat.org/cmdline.html
Little command line goodies	www.shell-fu.org

That's all folks!