Filesystem

bzip2 [opts] [filepattern] ·**bzip2 Compression (better)**
cd [-] [directory] ·**Change directory**
 - :Change to the previous directory you were in
chmod [opts] <mode> <filepattern> ·**Change permissions**
 -R :Change permissions recursively
chown [opts] <user>[.group] <file> ·**Change ownership**
 -R :Change ownership recursively
cp [opts] <from> <to> ·**Copy files and directories**
 -i :Interactive mode. Prompt before overwriting
 -p :Preserve file permissions and ownership
 -R :Copy directories recursively
df [opts] [device name] ·**Print filesystem usage info**
 -a :Show all filesystems
 -h :Human readable format/Quantify byte information
 -i :Show inode usage info
du [opts] [pattern] ·**Show space usage on files and dirs**
 -c :Produce a grand total for all arguments
 -h :Human readable format/Quantify byte information
 -s :Summarize. Only show a total for each argument
find <path> [opts] ·**Search for a file**

Learning find, once and for all!

Find all non-world readable html/htm files and change their user ownership to fred using chmod:

```
find / -type f -name '*.html' -o -name '*.htm' -a
  -perms -644 -exec chown fred {} \;
```

gzip [opts] <filepattern> ·**Compress a file or files**
 -1..9 : Set compression level. 9=highest, 1=lowest
 -d :Decompress file. Same as the **gunzip** command
 -v :List the statistics for a compressed file
ln [opts] <tofile> <linkfile> ·**Create a sym/hard link**
 -s :Create a symbolic link between files (alias name)
 -f :Force creation, even if the link file exists
ls [opts] [pattern] ·**List file and directory entries**
 -a :List all files including ones that start with '.'
 -d :List directories themselves, not their contents
 -l :Long list. Shows permissions and modified time
 -r :Recusively list files in directories
 -S :Sort output by file size
 -h :Human readable format/Quantify byte information
 -X :Sort by filename extension
 -1 :Print output files one per line
 --time=atime :Show last access timestamp for file
mkdir [opts] <dirname> ·**Make a new directory**
 -p :Create parent directories if they don't exist
mv [-i] <frompattern> <tofile> ·**Move/Rename a file**
 -i :Interactive move (Prompt before moving files)
rm [opts] <filepattern> ·**Remove a file**
 -f :Force removal (Don't ask if it's ok to remove)
 -i :Interactive remove (Ask before removing each file)
 -r :Recusively delete directories an their contents
shred [opts] <filepattern> ·**Delete file data securely**
 -n :Number of pattern iterations to run (default 25)
 -u :Truncate and remove the file after overwriting
 -z :Add a final overwrite with zeros to hide shreding
tar [opts] [tarfile] [pattern] ·**Create an archive**
 c :Create mode. Create a tar archive
 x :Extract mode. Untar archive contents
 t :List mode. List the contents of the archive
 f :Specify a tarfile to use
 v :Verbose mode. Show files being added or untared
 z,j :De/compress. Send i/o through gzip(z) or bzip2(j)
touch [opts] <pattern> ·**Update the timestamp on a file**
 -t :Specify a timestamp to use instead of current time

Informational

cat [opts] [filepattern] ·**Print file contents on STOUT**
 -E :Display a $ at the end of each line
 -T :Show tabs as ^I
 -v :Show non-printing characters
date [opts] ·**Print or set the system date and time**
 --date=*STRING* :display time described by *STRING*
 --set=*STRING* :set time described by *STRING*
dmesg [opts] ·**Print or control the kernel ring buffer**
 -c :Clear the contents of the ring buffer
file [opts] [filepattern] ·**Determine the file type**
 -z :Try to look inside compressed files
finger [opts] [userpattern] ·**Show info about system users**
 -m :Match the exact username specified
free [opts] ·**Display free and used memory in the system**
 -b :Display the information in bytes
hexdump [opts] ·**Show all the characters of a file**
 -c :Display the input offset in hexidecimal
last [opts] [username] ·**Show last system logins for users**
 -*num* :Show last *num* of sessions
 -a :Display the hostname in the last column
 -d :Translates IP numbers to their hostname
 -f <file> :Use file as last log
less [opts] [filepattern] ·**View a file a page at a time**
 -i :Do case insensitive searching
 -S :Don't wrap long lines
 +[less commands] :Pass initial commands to less
lsof [opts] [names] ·**List all open files**

Try these useful tasks with lsof

When the CD-ROM is "busy":	**lsof /dev/cdrom**
Programs using audio:	**lsof /dev/dsp**
List open ipv4 network files:	**lsof -i 4 -a**

man [opts] [section] <manpage> ·**View software manual pages**
 -a :View all available manual pages for name
 Ex: 'man ls' or 'man -a nice' or 'man 5 crontab'
md5sum [opts] [filepattern] ·**Show the uniqueness of files**
 -c :Check MD5 sums of files against md5sum listfile
ps [opts] ·**Show what processes are running on the system**
 a :Select all processes on a terminal
 u :Display user oriented format. More columns
 x :Select processes without a controlling TTY
 w :Show an extra line of process entry per w specified
Ex: ps auxwww =Displays all process information on system
quota [opts] [user] ·**Display disk usage and limits**
 -v :Display filesystems where no quota is set
random **</numpattern/>** ·**Print out a random number from numpattern**
 Ex: random /500..1000/ (print a random number between 500 and 1000)
slocate [opts] [pattern] ·**Locate pattern in file index db**
 -i : Case insensitive search
 -r : Search the database using POSIX regular expressions
time [opts] [command] ·**Show resource usage for a command**
top [opts] ·**Display top CPU processes every X seconds**
 -d *sec* :Set the delay to *sec* seconds before refreshing
umask [opts] [mode] ·**Set the default file permissions**
 -S :Show current symbolic umask
uname [opts] ·**Show OS and system information**
 -a :Show everything
uptime ·**Show system uptime and load**
w [opts] [user] ·**Show who is logged in/what they are doing**
whereis [command] ·**Locate the related files for a command**
which [command] ·**Show full path to the specified command**
who [opts] [args] ·**Show who is logged in**

Text Filtering / Mutative

average **[opts] [fileargs]** ·**Print the average of all numbers encountered**
awk [opts] [exp] ·**pattern scanning and processing language**
 -F<*fs*> :Set the field separator to <*fs*>
 Ex: cat access_log | awk {'print $1'} (prints hostnames)
 Do a 'man awk' for more information and examples
comm [opts] [file1] [file2] ·**Compare two sorted files**
 -1 :Suppress lines unique to left file
 -2 :Suppress lines unique to right file
 -3 :Supress lines unique to both files
csplit [opts] [file] [pattern] ·**Split a file on context**
 -f *prefix* :Use *prefix* instead of xx in output filenames
 -n <*digits*> :Use <*digits*> number of digits instead of 2
 -z :Remove empty output files
 Ex: csplit mailspoolfile "/^From /" {*}
cut [opts] [filepattern] ·**Remove sections from each line**
 -c *range* :Output only the characters in *range*
 Ex: cut -c 1-80 *file* (truncate lines at 80 characters)
diff [opts] [file1] [file2] ·**Differentiate two files**
 Ex: diff program-old.c program.c > program.patch
echo [opts] [string] ·**Print a line of text**
 -e :Enable interpretation of backslashed sequences
 -n :Don't automatically insert a newline character
fold [opts] [files] ·**Wrap each line to a specified width**
 -s :Break at spaces instead of in the middle of a word.
 -w <*WIDTH*> :Use <*WIDTH*> columns rather than 80
grep [opts] [pattern] [file] ·**Print lines matching pattern**
 -B <*num*> :Print <*num*> lines of leading context on matches
 -C <*num*> :Print <*num*> lines of trailing context on matches
 -E :Interpret *pattern* as an extended regular expression
 -i :Do case insensitve matching
 -l :Just print the files that match the *pattern*
 -r :Read all files under each directory recursively
 -v :Print the lines that don't match *pattern*
head [opts] [file] ·**Print the first part of a file**
 -n *num* :Print the first *num* lines instead of the first 10
numsum **[opts] [filepattern]** ·**Print the sum of a group of numbers**
 Ex: cat numbers.txt | numsum (Add up all numbers in a file)
numgrep **</numpattern/> [filepattern]** ·**Print lines matching numpattern**
 Ex: cat numbers.txt | numgrep /2..100/ (Print numbers from 2 to 100)
nl [opts] [file] ·**Number the lines of a file**
paste [opts] [files] ·**Merge lines of files horizontally**
patch [opts] [patchfile] ·**Patch a file using a diff file**
sed [expression] [file] ·**Stream editor**
 Ex: cat file | sed 's/*frompattern*/*topattern*/' > output
sort [opts] [file] ·**Sort lines of text files**
 -n :Compare according to string numerical value
 -r :Reverse the result of comparisons
split [opts] [file] ·**Split a file into pieces**
 -l <*num*> :Put <*num*> lines per output file
tail [opts] [file] ·**Print the last lines of a file**
 -f :Output appended data as the file grows
 -n <*num*> :Print last <*num*> lines of instead of the last 10
tee [opts] [file] ·**Send current output stream to file**
 -a :Append to the given file instead of overwriting
tr [opts] <set1> [set2] ·**Translate char. from set1 to set2**
 Ex: cat index.html | tr A-Z a-z > index-new.html
uniq [opts] [input] [output] ·**Remove duplicate lines**
 -c :Prefix lines with number of occurances
 -d :Only print duplicated lines
 -u :Only print unique lines
 -w <*n*> :Check no more than <*n*> characters in lines
wc [opts] [file] ·**Print the number of lines in files, etc.**
 -m :Print the character count
 -l :Print the line count
 -w :Print the word count
 -L :Print the length of the longest line

Network

ifconfig [devicename] [action] [options]
ipchains [opts] ·**Manip. ipchains firewall(kernel 2.2+)**
iptables [opts] ·**Manip. iptables firewall(kernel 2.4+)**
 -F: Flush current set of rules (Careful!)
 -L: List the current rules
 -n: Display rules without doing DNS lookups (faster)
mail [opts] [address] ·**Send mail from the command line**
 -s *subject* :Specify the subject as *subject*
 -c *list* :Send carbon copy to *list* of users
 -b *list* :Send blind carbon copy to *list* of users
 Ex: echo "Meet me at noon." | mail -s "Reminder" -c
 bob@company.com,suzy@company.com jack@company.com
netstat [opts] ·**Print network connections and info**
 -a :Show both listening and non-listening sockets
 -n :Do not attempt to resolve IP addresses
 -t :Only show tcp socket connection table
ping [opts] [host] ·**Send ICMP packets to network hosts**
 -c *count* :Send *count* number of packets and then quit
 -i *sec* :Wait *sec* seconds between sending packets
route [opts] [target] ·**Show/Manipulate IP routing table**
 -n :Show numerical addresses instead of hostnames
scp [opts] [[host:]fromfile] [[host:]to] · **Secure copy**
 -C :Compresses the data that is sent over the session
 -r :Recursively copy directories
ssh [opts] [[user@]host] [command] · **Secure shell/login**
 -C :Compresses the data that is sent over the session
sniffit [opts] ·**Record TCP network traffic**
tcpdump [opts] [expression] ·**Dump traffic on a network**

Some examples of how to use tcpdump

tcpdump host foo (To or from host 'foo')
tcpdump not host foo (Not to or from host foo)
tcpdump port http (All to or from port 80)
tcpdump ip and not net localnet (non-local net.)

telnet [opts] [host] [port] ·**Open TCP socket to a host**
 -n <file> : Opens *file* for recording trace information
 -x :Turns on encryption of the data stream if possible
traceroute [opts] [host] ·**Show the route packets take**
 -n :Don't do DNS lookups of the IP addresses
wget [opts] [URL] ·**Make a HTTP request from the shell**
 -r :Recursive get the URL and all it's links
 -k :Convert the non-relative links to relative ones
whois [opts] <arg[@server]> ·**Query a whois database**
 Ex: whois domain.com
 whois a.b.c.d (IPv4 address)

Bash Shell

> ·**Send STDOUT to a file. overwrite/create a file**
 Ex: ls -l > list-of-files.txt
>> ·**Send STDOUT to a file, appending to te end of the file**
 Ex: ps aux > pslog.txt
| ·**Send the STDOUT from a command to the STDIN of another**
 Ex: cat listofnames | sort
2> ·**Send STDERR to a file, overwriting the filename**
 Ex: startx 2> X-errorlog

Command pipelines in action

(records all running apache processes and kills last 10 in process table)

```
$ ps auxw | grep [a]pache | tee apache-allprocs.txt | grep ^apache |
  awk {'print $2'} | tail -n 10 | xargs kill
```

alias ·**Create a command alias in the shell**
 Ex: alias ls='ls -la --color=auto'
cd [-] [directory] ·**Change the current working directory**
 - :Change to the previous directory you were in
clear ·**Clear the terminal display** (also can use Ctrl-L)
env [opts] [command] ·**Run command in modified environment**
export [opts] [variable] ·**Export an environment variable**
 Ex: export TERM=vt100
for ·**Execute sequence of commands for a list of items**
 Ex: for i in *.mp3 ; do mpg123 $i ; done
history ·**Show the command history up til now**
nice [opts] [command] ·**Set the OS process priority**
 Ex: nice 19 gzip access_log (lowest priority on Linux)
 Ex: nice -20 kswapd (real time priority on Linux)
pwd ·**Print out the current working directory**
range **[opts] </numpattern/>** ·**Print a range of numbers for use in loop**
 Ex: for i in `range /1..20/` ; do mkdir $i ; done
renice [opts] <arg> ·**Change priority of a running process**
 -p <*PID*>: Specify a process id (<*PID*>) to "renice"
 Ex: *bob 6319 ? S 0:20 gzip bigfile.txt* (output line from running ps auxw)
 then run: 'renice 19 -p 6319' (which changes the priority)
reset ·**Initializes the terminal as if you just logged in**
set ·**Set a shell option or variable (run 'help set')**
sleep ·**Pause for specified period before continuing**
 Ex: ps aux ; sleep 3600 ; ps aux
umask ·**Set the default file permissions**
 Ex: umask 022 (files will be created 644 by default)
while ·**Loop that runs commands while a condition is true**
 Ex: while (true) ; do ps auxw ; sleep 1m ; done > pslog
xargs [opts] [command] ·**Execute a command for each arg**
 -n *number* :How many arguments to give each command run
 -p :Prompt the user before each command is run

Admin

adduser [opts] <username> ·**Add a user to the local system**
 -d <*dir*> : Set the home directory for the user to *dir*
 -g <*group*> : Set the primary group for the user to *group*
 -G <*group,group,…*> : Set additional groups for the user
 -s <*shell*> : Set the default shell for the user to *shell*
crontab [opts] ·**Edit user crontab for periodic execution**
 -e : Edit a crontab
 -u <user> : Specify <user> for crontab operation
edquota [opts] <user> ·**Edit a user's or group's quota**
 -g : Edit the group quota instead of user quota
fsck [opts] [filesystem] ·**Check and repair a filesystem**
 -y : Answer yes to any questions. (Use with caution!)
kill [-*signal*] <pid> ·**Terminate a process/Send it a signal**
 -HUP,-1 : Signal usually makes process to reread config
 -9 :Send a SIGKILL, process must die
 -l :Print a list of signal names and numbers
killall [-*signal*] [name]·**Kill processes by name**
 -e :Require an exact name of a process
 -i :Interactively ask for confirmation before killing
 Ex: killall -9 sendmail
ldd [opts] [program] ·**Show a programs library dependencies**
ldconfig ·**Configure dynamic linker run time bindings**
 (run this program after changing /etc/ld.so.conf)
makewhatis ·**Create the whatis db for searching man pages**
mount [opts] <path/device> [mountpoint]·**Mount a filesystem**
 -o <*opts*> : Specify options for mounting. Listed below
 loop - Mount a disk file such as a CD-ROM image or floppy image
 remount - Remount the filesystem with new options
 ro, rw - Mount filesystem in readonly or read-write mode
 user - Allow normal users to mount this filesystem
 -r :Mount the filesystem read-only. Same as '-o ro'
 -t <*vfstype*> :Specify the type of filesystem to mount
 ext2, ext3 - Native Linux partition types.
 reiserfs - Advanced Linux filesystem
 xfs, jfs - Other advanced Linux filesystems
 vfat - Windows 9x 32-bit partition type
 msdos - Old DOS/Windows partition type
 iso9660 - CD-ROM filesystem
 nfs - Network remote filesystem
passwd [opts] [username]·**Change user's system password**
 -l :Lock the password for the account
 -u :Unlock the password for the account
 -S :Show the status of the password for the account
su [-] [username] ·**Switch users or login as the superuser**
 - :Make shell a login shell
 -c <*command*> : Run <*command*> as *username*
umount [opts] [path/device]·**Unmount a mounted filesystem**
 -f :Force unmounting (in case of unreachable NFS system)
 -l :Complete the unmount once filesystem is no longer busy

‡ Commands and options displayed in **red** can only be used by the superuser (root).
¥ Commands that are underlined may not be available by default on some distributions of Linux and will need to be installed.
£ The programs *random*, *average*, *numsum* and *numgrep* are part of the *num-utils* suite of programs which can be found at http://suso.suso.org/programs/num-utils/

Common commands and their syntax for the Linux® OS environment

This quick info sheet is Copyright 2001 by Suso Banderas and Copyright 2005 by Suso Technology Services, Inc. This work is licensed under the Creative Commons Attribution-ShareAlike License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.
For more information, downloads, ordering laminated copies and the original OpenOffice.org file that created this sheet, please visit http://suso.org/infosheets/ v1.2 (2005-03-28)
Linux is a registered trademark of Linus Torvalds. All other trademarks belong to their respective holders. Have a nice day!

suso.org

What follows are some common commands used at the MS-DOS prompt in Windows 9*x*, and in Linux, as well as a basic example of how the command is used at the Linux shell prompt. Note that these commands usually have a number of options. To learn more about each command, read its associated man page (for example, type **`man ls`** at the shell prompt to read about the `ls` command).

Table C-1. Similar Commands

Command's Purpose	***MS-DOS***	***Linux***	***Basic Linux Example***
Copies files	`copy`	`cp`	`cp` *`thisfile.txt /home/thisdirectory`*
Moves files	`move`	`mv`	`mv` *`thisfile.txt /home/thisdirectory`*
Lists files	`dir`	`ls`	`ls`
Clears screen	`cls`	`clear`	`clear`
Closes prompt window	`exit`	`exit`	`exit`
Displays or sets date	`date`	`date`	`date`
Deletes files	`del`	`rm`	`rm` *`thisfile.txt`*
"Echoes" output on the screen	`echo`	`echo`	`echo` *`this message`*
Edits files with simple text editor	`edit`	`pico`[a]	`pico` *`thisfile.txt`*
Compares the contents of files	`fc`	`diff`	`diff` *`file1 file2`*
Finds a string of text in a file	`find`	`grep`	`grep` *`this word or phrase thisfile.txt`*
Formats a floppy	`format a:` (if floppy's in `A:`)	`mke2fs` (or `mformat`[b])	`/sbin/mke2fs /dev/fd0` (`/dev/fd0` is the Linux equivalent of `A:`)
Displays command help	*`command /?`*	`man`[c]	`man` *`command`*
Creates a directory	`mkdir`	`mkdir`	`mkdir` *`directory`*
Screens through a file	`more`	`less`[d]	`less` *`thisfile.txt`*
Renames a file	`ren`	`mv`	`mv` *`thisfile.txt thatfile.txt`*[e]
Shows your location in the file system	`chdir`	`pwd`	`pwd`
Changes directories with a specified path (*absolute path*)	`cd` *`pathname`*	`cd` *`pathname`*	`cd` *`/directory/directory`*
Changes directories with a *relative path*	`cd ..`	`cd ..`	`cd ..`

Command's Purpose	*MS-DOS*	*Linux*	*Basic Linux Example*
Displays the time	`time`	`date`	`date`
Shows amount of RAM and use	`mem`	`free`	`procinfo`

Notes:
a. Pico is a simple text editor; other editors you can use in place of **pico** include **emacs** and **vi**.
b. This formats a disk for the DOS filesystem.
c. Or you can use `info` for some commands.
d. You can also another *pager*, called `more`, to scroll through a file a screen at at time.
e. The `mv` command serves double-duty, because it can both move a file and, if you want to rename a file in the same directory, you "move" that file to the same directory with a new name, as in this example.

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
```

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
```

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
```

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)
```

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 11http://www.freebsd.org/handbook/configtuning-kernel-limits.html 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
```

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
# 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)

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
```

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)
```

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
```

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
```

Processes

Listing | Priority | Background/Foreground | Top | Kill

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 axjf                        # All processes in a tree format (Linux)
# ps aux | grep 'ss[h]'            # Find all ssh pids without the grep pid
# pgrep -l sshd                  # Find the PIDs of processes by (part of) name
# echo $$                        # The PID of your shell
# fuser -va 22/tcp                 # List processes using port 22 (Linux)
# pmap PID                         # Memory map of process (hunt memory leaks) (Linux)
# fuser -va /home                  # List processes accessing the /home partition
# strace df                      # Trace system calls and signals
# truss df                       # same as above on FreeBSD/Solaris/Unixware
```

Priority

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

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

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

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

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

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

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

FreeBSD uses idprio/rtprio (0 = max priority, 31 = most idle):

```
# idprio 31 make              # compile in the lowest priority
# idprio 31 -1234             # set PID 1234 with lowest priority
# idprio -t -1234             # -t removes any real time/idle priority
```

Background/Foreground

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

```
# 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 &
```

Top

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

* u [user name] To display only the processes belonging to the user. Use + or blank to see all users
* k [pid] Kill the process with pid.

* 1 To display all processors statistics (Linux only)
* R Toggle normal/reverse sort.

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)

File System

Disk info | Boot | Disk usage | Opened files | Mount/remount | Mount SMB | Mount image | Burn ISO | Create image | Memory disk | Disk performance

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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.

```
# 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 18.7http://www.freebsd.org/handbook/creating-dvds.html.

```
                # -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 programhttp://freshmeat.net/projects/bchunk/ can do this. It is in the FreeBSD ports in sysutils/bchunk.

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

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
# 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
```

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
```

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
```

Network

Routing | Additional IP | Change MAC | Ports | Firewall | IP Forward | NAT | DNS | DHCP | Traffic | QoS | NIS | Netcat

Debugging (See also Traffic analysis)

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
```

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.

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
```

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 etherchangehttp://ntsecurity.nu/toolbox/etherchange. Or look for "Mac Makeup", "smac".

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
```

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
```

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
```

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
```

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
```

DNS

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

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

Check the system domain name with:

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

Windows

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

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

Flush DNS

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

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

Forward queries

Dig is you friend to test the DNS settings. For example the public DNS server 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

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!

Traffic analysis

Bmonhttp://people.suug.ch/~tgr/bmon/ is a small console bandwidth monitor and can display the flow on different interfaces.
Sniff with tcpdump

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

Additional important options:

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

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

Nmaphttp://insecure.org/nmap/ 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.

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.

```
# 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
```

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
# rpcinfo -p servername     # Report RPC services of the server
```

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
```

Netcat

Netcathttp://netcat.sourceforge.net (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[...]http://www.g-loaded.eu/2006/11/06/netcat-a-couple-of-useful-examples and herehttp://www.terminally-incoherent.com/blog/2007/08/07/few-useful-netcat-tricks.
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
```

SSH SCP

Public key | Fingerprint | SCP | Tunneling

Public key authentication

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

* Use ssh-keygen to generate a key pair. ~/.ssh/id_dsa is the private key, ~/.ssh/id_dsa.pub is the public key.

* Copy only the public key to the server and append it to the file ~/.ssh/authorized_keys2 on your home on the server.

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

Using the Windows client from ssh.com

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

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

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
```

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 sshfshttp://fuse.sourceforge.net/sshfs.html.

Tunneling

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

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

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

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

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

Netbios and remote desktop forward to a second server

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

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

The smb share can now be accessed with \\127.0.0.1\, but only if the local share is disabled, because the local share is listening on port 139.

It is possible to keep the local share enabled, for this we need to create a new virtual device with a new IP address for the tunnel, the smb share will be connected over this address. Furthermore the local RDP is already listening on 3389, so we choose 3388. For this example let's use a virtual IP of 10.1.1.1.

* With putty use Source port=10.1.1.1:139. It is possible to create multiple loop devices and tunnel. On Windows 2000, only putty worked for me. On Windows Vista also forward the port 445 in addition to the port 139. Also on Vista the patch KB942624 prevents the port 445 to be forwarded, so I had to uninstall this path in Vista.

* With the ssh.com client, disable "Allow local connections only". Since ssh.com will bind to all addresses, only a single share can be connected.

Now create the loopback interface with IP 10.1.1.1:

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

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

If it is not working:

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

Connect two clients behind NAT

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

On client cliuser (from destination to gate):

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

On client cliadmin (from host to gate):

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

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

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

Connect to VNC behind NAT

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

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

On client cliadmin (from host to gate):

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

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

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

Dig a multi-hop ssh tunnel

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

Suppose we want to forward the ssh port from a client to a server over two hops. Once the tunnel is build, it is possible to connect to the server directly from the client (and also add an other port forward).
Create tunnel in one shell

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

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

Use tunnel with an other shell

client -> server using tunnel 5678

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

Autoconnect and keep alive script

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

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

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

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

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.

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
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
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.

RSYNC

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

Copy the directories with full content:

```
# rsync -a /home/colin/ /backup/colin/              # "archive" mode. e.g keep the same
# rsync -a /var/ /var_bak/
```

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

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

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

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

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

Use port 20022 for the ssh connection:

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

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

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

Some important options:

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

Rsync on Windows

Rsync is available for Windows through cygwin or as stand-alone packaged in cwrsynchttp://sourceforge.net/projects/sereds. 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
```

SUDO

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

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

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
```

Encrypt Files

OpenSSL

A single file

Encrypt and decrypt:

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

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

tar and encrypt a whole directory

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

tar zip and encrypt a whole directory

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

* Use -k mysecretpassword after aes-128-cbc to avoid the interactive password request. However note that this is highly insecure.

* Use aes-256-cbc instead of aes-128-cbc to get even stronger encryption. This uses also more CPU.

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 Starthttp://www.madboa.com/geek/gpg-quickstart and GPG/PGP Basicshttp://aplawrence.com/Basics/gpg.html and the gnupg documentationhttp://gnupg.org/documentation 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.

```
# 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)
```

Encrypt Partitions

Linux with LUKS | Linux dm-crypt only | FreeBSD GELI | FBSD pwd only

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.

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.

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 18.6http://www.freebsd.org/handbook/disks-encrypting.html 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
```

/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
```

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).
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.).

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).
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
```

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.

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.

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!

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
```

CVS

Server setup | CVS test | SSH tunneling | CVS usage

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 (default port: 2401/tcp). 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
```

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
```

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
```

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
```

SVN

Server setup | SVN+SSH | SVN over http | SVN usage

Subversion (SVN)http://subversion.tigris.org/ 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 bookhttp://svnbook.red-bean.com/en/1.4/.
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 (default port: 2690/tcp).

* 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

<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
```

SVN commands and usage

See also the Subversion Quick Reference Cardhttp://www.cs.put.poznan.pl/csobaniec/Papers/svn-refcard.pdf. Tortoise SVNhttp://tortoisesvn.tigris.org 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
```

Useful Commands

less | vi | mail | tar | dd | screen | find | Miscellaneous

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

vi

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

The editors nano and pico are usually available too and are easier (IMHO) to use.
Quit

* :w newfilename save the file to newfilename
* :wq or :x save and quit
* :q! quit without saving

Search and move

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

Delete copy paste text

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

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.

tar

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

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

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

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

Extract

```
# tar -tzf home.tgz            # look inside the archive without extracting (list)
# tar -xf home.tar             # extract the archive here (x for extract)
# tar -xzf home.tgz            # same with zip compression (-xjf for bzip2 compression)
                               # remove leading path gallery2 and extract into gallery
# tar --strip-components 1 -zxvf gallery2.tgz -C gallery/
# tar -xjf home.tbz home/colin/file.txt    # Restore a single file
```

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/
```

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. 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
# 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
```

screen

Screen (a must have) has two main functionalities:

* Run multiple terminal session within a single terminal.

* A started program is decoupled from the real terminal and can thus run in the background. The real terminal can be closed and reattached later.

Short start example

start screen with:

```
# screen
```

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

```
# top
```

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

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

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

screen -x

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

Screen commands (within screen)

All screen commands start with Ctrl-a.

* Ctrl-a ? help and summary of functions
* Ctrl-a c create an new window (terminal)
* Ctrl-a Ctrl-n and Ctrl-a Ctrl-p to switch to the next or previous window in the list, by number.
* Ctrl-a Ctrl-N where N is a number from 0 to 9, to switch to the corresponding window.
* Ctrl-a " to get a navigable list of running windows
* Ctrl-a a to clear a missed Ctrl-a
* Ctrl-a Ctrl-d to disconnect and leave the session running in the background
* Ctrl-a x lock the screen terminal with a password

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

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 findhttp://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
```

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
# history | tail -50             # Display the last 50 used commands
```

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
```

Install Software

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

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

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
```

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 portshttp://www.freebsd.org/handbook/ports.html

The port tree /usr/ports/ is a collection of software ready to compile and install (see man ports). 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 of this port
```

```
# pkgdb -F                    # Fix the package registry database
```

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
```

Convert Media

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

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 and mogrify (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
# convert image000* -resample 120x120 -compress JPEG -quality 80 images.pdf
# mogrify -format png *.ppm          # convert all ppm images to png format
```

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 '='
```

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.

Copy an audio cd

The program cdparanoiahttp://xiph.org/paranoia/ 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
```

Printing

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
# 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
```

Print to a PDF file even if the application does not support it. Use gs on the print command instead of lpr.

```
# gs -q -sPAPERSIZE=a4 -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=/path/file.pdf
```

Databases

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.

MySQL

Change mysql root or username password

Method 1

```
# /etc/init.d/mysql stop
or
# killall mysqld
# mysqld --skip-grant-tables
```

```
# mysqladmin -u root password 'newpasswd'
# /etc/init.d/mysql start
```

Method 2

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

Create user and database (see MySQL dochttp://dev.mysql.com/doc/refman/5.1/en/adding-users.html)

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

Grant remote access

Remote access is typically permitted for a database, and not all databases. The file /etc/my.cnf contains the IP address to bind to. 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.

SQLite

SQLitehttp://www.sqlite.org 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
```

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.
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
```

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
```

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
```

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.

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
```

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).

Scripting

Basics | Script example | awk | sed | Regular Expressions | useful commands

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.

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
```

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
```

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)
```

Some sed commands

Here is the one liner gold minehttp://student.northpark.edu/pemente/sed/sed1line.txt. And a good introduction and tutorial to sedhttp://www.grymoire.com/Unix/Sed.html.

```
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
```

Regular Expressions

Some basic regular expression useful for sed too. See Basic Regex Syntaxhttp://www.regular-expressions.info/reference.html for a good primer.

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

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
```

Programming

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);
```

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
```

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
```

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                    # list dynamic object dependencies
# ar rcs staticlib.a *.o                 # create static archive
# ar t staticlib.a                     # print the objects list from the archive
# ar x /usr/lib/libc.a version.o            # extract an object file from the archive
# nm version.o                          # show function members provided by object
```

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}
```

Online Help

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

Other Unix/Linux references

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

People who use Windows without DOS, or a Macintosh, or PPP without a terminal, or an ISP's menu without the Unix prompt are at a disadvantage. Something is happening, and they don't know what it is. I like to know what's really going on, so I've been learning some Unix.

The Net is a Unix place. I'm no wizard, but I'm comfortable with basic commands and occasionally type "**rm**" at my DOS prompt instead of "**del**". This is my Unix cheat sheet, so *I* can remember. Uppercase and lowercase matter. These commands (mostly) work with my C-shell account on RAIN. Your account might be different, especially if your prompt ends with a "**$**" (Korn shell) rather than a "**%**", so be cautious. When I need help, I reach for the books **UNIX in a Nutshell** (O'Reilly) and **Unix Unbound** by Harley Hahn (Osborne/McGraw Hill, 1994).

This page won't look right without **table** support. Most of this is available in a text version.

Help on any Unix command. RTFM!

`man {command}`	Type **man ls** to read the manual for the **ls** command.
`man {command} > {filename}`	Redirect help to a file to download.
`whatis {command}`	Give short description of command. (Not on RAIN?)
`apropos {keyword}`	Search for all Unix commands that match keyword, eg **apropos file**. (Not on RAIN?)

List a directory

`ls {path}`	It's ok to combine attributes, eg **ls -laF** gets a long listing of all files with types.
`ls {path_1} {path_2}`	List both {path_1} and {path_2}.
`ls -l {path}`	Long listing, with date, size and permisions.
`ls -a {path}`	Show all files, including important .dot files that don't otherwise show.
`ls -F {path}`	Show type of each file. "/" = directory, "*" = executable.
`ls -R {path}`	Recursive listing, with all subdirs.
`ls {path} > {filename}`	Redirect directory to a file.
`ls {path} | more`	Show listing one screen at a time.
`dir {path}`	Useful alias for DOS people, or use with **ncftp**.

Change to directory

`cd {dirname}`	There must be a space between.
`cd ~`	Go back to home directory, useful if you're lost.
`cd ..`	Go back one directory.
`cdup`	Useful alias, like "cd ..", or use with **ncftp**.

Make a new directory

```
mkdir {dirname}
```

Remove a directory

`rmdir {dirname}`	Only works if {dirname} is empty.
`rm -r {dirname}`	Remove all files and subdirs. Careful!

Print working directory

`pwd`	Show where you are as full path. Useful if you're lost or exploring.

Copy a file or directory

`cp {file1} {file2}`	
`cp -r {dir1} {dir2}`	Recursive, copy directory and all subdirs.
`cat {newfile} >> {oldfile}`	Append newfile to end of oldfile.

Move (or rename) a file

`mv {oldfile} {newfile}`	Moving a file and renaming it are the same thing.
`mv {oldname} {newname}`	

Delete a file

`rm {filespec}`	**?** and ***** wildcards work like DOS should. "?" is any character; "*" is any string of characters.
`ls {filespec}` `rm {filespec}`	Good strategy: first list a group to make sure it's what's you think... ...then delete it all at once.

Download with zmodem

(Use **sx** with xmodem.)

`sz [-a|b] {filename}`	**-a** = ascii, **-b** = binary. Use binary for everything. (It's the default?)
`sz *.zip`	Handy after downloading with FTP. Go talk to your spouse while it does it's stuff.

Upload with zmodem

(Use **rx** with xmodem.)

`rz [-a|b] (filename}`	Give **rz** command in Unix, THEN start upload at home. Works fine with multiple files.

View a text file

`more {filename}`	View file one screen at a time.
`less {filename}`	Like **more**, with extra features.
`cat {filename}`	View file, but it scrolls.
`cat {filename} | more`	View file one screen at a time.
`page {filename}`	Very handy with **ncftp**.
`pico {filename}`	Use text editor and don't save.

Edit a text file.

`pico {filename}`	The same editor PINE uses, so you already know it. **vi** and **emacs** are also available.

Create a text file.

`cat > {filename}`	Enter your text (multiple lines with **enter** are ok) and press **control-d** to save.
`pico {filename}`	Create some text and save it.

Compare two files

`diff {file1} {file2}`	Show the differences.
`sdiff {file1} {file2}`	Show files side by side.

Other text commands

`grep '{pattern}' {file}`	Find regular expression in file.
`sort {file1} > {file2}`	Sort file1 and save as file2.
`sort -o {file} {file}`	Replace file with sorted version.
`spell {file}`	Display misspelled words.
`wc {file}`	Count words in file.

Find files on system

`find {filespec}`	Works with wildcards. Handy for snooping.
`find {filespec} > {filename}`	Redirect find list to file. Can be big!

Make an Alias

`alias {name} '{command}'`	Put the command in 'single quotes'. More useful in your **.cshrc** file.

Wildcards and Shortcuts

`*`	Match any string of characters, eg **page*** gets page1, page10, and page.txt.
`?`	Match any single character, eg **page?** gets page1 and page2, but not page10.
`[...]`	Match any characters in a range, eg **page[1-3]** gets page1, page2, and page3.
`~`	Short for your home directory, eg **cd ~** will take you home, and **rm -r ~** will destroy it.
`.`	The current directory.
`..`	One directory up the tree, eg **ls ..**.

Pipes and Redirection

(You **pipe** a command to another command, and **redirect** it to a file.)

`{command} > {file}`	Redirect output to a file, eg **ls > list.txt** writes directory to file.
`{command} >> {file}`	Append output to an existing file, eg **cat update >> archive** adds update to end of archive.
`{command} < {file}`	Get input from a file, eg **sort < file.txt**
`{command} < {file1} > {file2}`	Get input from `file1`, and write to `file2`, eg **sort < old.txt > new.txt** sorts old.txt and saves as new.txt.
`{command} | {command}`	Pipe one command to another, eg **ls | more** gets directory and sends it to **more** to show it one page at a time.

Permissions, important and tricky!

Unix permissions concern who can **read** a file or directory, **write** to it, and **execute** it. Permissions are granted or withheld with a magic 3-digit number. The three digits correspond to the **owner** (you); the **group** (?); and the **world** (everyone else).

Think of each digit as a sum:

execute permission	= 1
write permission	= 2
write and execute (1+2)	= 3
read permission	= 4
read and execute (4+1)	= 5
read and write (4+2)	= 6
read, write and execute (4+2+1)	= 7

Add the number value of the permissions you want to grant each group to make a three digit number, one digit each for the owner, the group, and the world. Here are some useful combinations. Try to figure them out!

`chmod 600 {filespec}`	You can read and write; the world can't. Good for files.
`chmod 700 {filespec}`	You can read, write, and execute; the world can't. Good for scripts.
`chmod 644 {filespec}`	You can read and write; the world can only read. Good for web pages.
`chmod 755 {filespec}`	You can read, write, and execute; the world can read and execute. Good for programs you want to share, and your `public_html` directory.

Permissions, another way

You can also change file permissions with letters:

u = user (yourself) **g** = group **a** = everyone

r = read **w** = write **x** = execute

`chmod u+rw {filespec}`	Give yourself read and write permission
`chmod u+x {filespec}`	Give yourself execute permission.
`chmod a+rw {filespec}`	Give read and write permission to everyone.

Applications I use

`finger {userid}`	Find out what someone's up to.
`gopher`	Gopher.
`irc`	IRC, but not available on RAIN.
`lynx`	Text-based Web browser, fast and lean.
`ncftp`	Better FTP.
`pico {filename}`	Easy text editor, but limited. **vi** and **emacs** are available.

`pine`	Email.
`telnet {host}`	Start Telnet session to another host.
`tin`	Usenet.
`uudecode {filename}` `uuencode {filename}`	Do it on the server to reduce download size about 1/3.
`ytalk {userid}`	Chat with someone else online, eg **ytalk mkummel**. Please use **w** first so you don't interrupt a big download!

System info

`date`	Show date and time.
`df`	Check system disk capacity.
`du`	Check your disk usage and show bytes in each directory.
`more /etc/motd`	Read message of the day, "motd" is a useful alias..
`printenv`	Show all environmental variables (in C-shell% - use **set** in Korn shell$).
`quota -v`	Check your total disk use.
`uptime`	Find out system load.
`w`	Who's online and what are they doing?

Unix Directory Format

Long listings (**ls -l**) have this format:

```
- file
d directory,                                          * executable
^   symbolic links (?)  file size (bytes)   file name   / directory
^           ^                ^                  ^        ^
drwxr-xr-x 11 mkummel      2560 Mar  7 23:25 public_html/
-rw-r--r--  1 mkummel     10297 Mar  8 23:42 index.html
                                    ^
 ^^^        user permission  (rwx)      date and time last modified
    ^^^     group permission (rwx)
       ^^^  world permission (rwx)
```

How to Make an Alias

An alias lets you type something simple and do something complex. It's a shorthand for a command. If you want to type "dir" instead of "ls -l" then type **alias dir 'ls -l'**. The single quotes tell Unix that the enclosed text is one command.

Aliases are more useful if they're permanent so you don't have to think about them. You can do this by adding the alias to your **.cshrc** file so they're automatically loaded when you start. Type **pico .cshrc** and look for the alias section and add what you want. It will be effective when you start. Just remember that if you make an alias with the name of a Unix command, that command will become unavailable.

Here are a few aliases from my **.cshrc** file:

```
# enter your aliases here in the form:
# alias     this    means this

alias       h       history
alias       m       more
alias       q       quota -v
alias       bye     exit
alias       ls      ls -F
alias       dir     ls
alias       cdup    cd ..
alias       motd    more /etc/motd
```

How to Make a Script

A Unix script is a text file of commands that can be executed, like a **.bat** file in DOS. Unix contains a powerful programming language with loops and variables that I don't really understand. Here's a useful example.

Unix can't rename a bunch of files at once the way DOS can. This is a problem if you develop Web pages on a DOS machine and then upload them to your Unix Server. You might have a bunch of **.htm** files that you want to rename as **.html** files, but Unix makes you do it one by one. This is actually not a defect. (It's a feature!) Unix is just being more consistent than DOS. So make a script!

Make a text file (eg with **pico**) with the following lines. The first line is special. It tells Unix what program or shell should execute the script. Other # lines are comments.

```
#! /bin/csh
# htm2html converts *.htm files to *.html
foreach f ( *.htm )
  set base=`basename $f .htm`
  mv $f $base.html
end
```

Save this in your home directory as **htm2html** (or whatever). Then make it user-executable by typing **chmod 700 htm2html**. After this a ***** will appear by the file name when you **ls -F**, to show that it's executable. Change to a directory with **.htm** files and type **~/htm2html**, and it will do its stuff.

Think about scripts whenever you find yourself doing the same tedious thing over and over.

Dotfiles (aka Hidden Files)

Dotfile names begin with a "." These files and directories don't show up when you list a directory unless you use the **-a** option, so they are also called **hidden files**. Type **ls -la** in your home directory to see what you have.

Some of these dotfiles are crucial. They initialize your shell and the programs you use, like **autoexec.bat** in DOS and **.ini** files in Windows. **rc** means "run commands". These are all text files that can be edited, but change them at your peril. Make backups first!

Here's some of what I get when I type **ls -laF**:

.addressbook	my email addressbook.
.cshrc	my C-shell startup info, important!
.gopherrc	my **gopher** setup.

.history	list of past commands.
.login	login init, important!
.lynxrc	my **lynx** setup for WWW.
.ncftp/	hidden dir of **ncftp** stuff.
.newsrc	my list of subscribed newsgroups.
.pinerc	my **pine** setup for email.
.plan	text appears when I'm **fingered**, ok to edit.
.profile	Korn shell startup info, important!
.project	text appears when I'm **fingered**, ok to edit.
.signature	my signature file for mail and news, ok to edit.
.tin/	hidden dir of my **tin** stuff for usenet.
.ytalkrc	my **ytalk** setup.

DOS and UNIX commands

Action	***DOS***	***UNIX***
change directory	cd	cd
change file protection	attrib	chmod
compare files	comp	diff
copy file	copy	cp
delete file	del	rm
delete directory	rd	rmdir
directory list	dir	ls
edit a file	edit	pico
environment	set	printenv
find string in file	find	grep
help	help	man
make directory	md	mkdir
move file	move	mv
rename file	ren	mv
show date and time	date, time	date
show disk space	chkdsk	df
show file	type	cat
show file by screens	type filename | more	more
sort data	sort	sort

Other Unix Links

- Basic UNIX Guide
- Linux Info
- Quick Reference Card - for pico, pine, elm, and more.
- Stevem's Place On The Web - good Unix info and links from RAIN's sysadmin
- The UNIX FAQ
- The UNIX Guru Universe
- UNIXhelp for users
- #UNIX IRC homepage
- The UNIX Reference Desk

Linux Shortcuts and Commands:

Linux Newbie Administrator Guide
by Stan and Peter Klimas

This is a practical selection of the commands we use most often. Press `<Tab>` to see the listing of all available command (on your PATH). On my small home system, it says there are 2595 executables on my PATH. Many of these "commands" can be accessed from your favourite GUI front-end (probably KDE or Gnome) by clicking on the right menu or button. They can all be run from the command line. Programs that require GUI have to be run from a terminal opened under a GUI.

Legend:
<> = single special or function key on the keyboard. For example <Ctrl> indicates the "control" key.
italic = name of the file or variable you probably want to substitute with your own.
`fixed width` = in-line Linux commands and filenames.

Notes for the UNIX Clueless:
1. LINUX IS CASE-SENSITIVE. For example: Netscape, NETSCAPE and nEtscape are three different commands. Also my_filE, my_file, and my_FILE are three different files. Your user login name and password are also case sensitive. (This goes with the tradition of UNIX and the "c" programming language being case sensitive.)
2. Filenames can be up to 256 characters long and can contain letters, numbers, "." (dot), "_" (underscore), "-" (dash), plus some other not recommended characters.
3. Files with names starting with "." are normally not shown by the `ls` (list) or dir commands. Think of these files as "hidden". Use `ls -a` (list with the option "all") to see these files.
4. "/" is an equivalent to DOS "\" (root directory, meaning the parent of all other directories).
5. Under Linux, all directories appear under a single directory tree (there are no DOS-style drive letters).
6. In a configuration file, a line starting with # is a comment.

7.1 Linux essential shortcuts and sanity commands

`<Ctrl><Alt><F1>`
Switch to the first text terminal. Under Linux you can have several (6 in standard setup) terminals opened at the same time.

`<Ctrl><Alt><Fn>` (n=1..6)
Switch to the nth text terminal.

`tty`
Print the name of the terminal in which you are typing this command.

`<Ctrl><Alt><F7>`
Switch to the first GUI terminal (if X-windows is running on this terminal).

`<Ctrl><Alt><Fn>` (n=7..12)
Switch to the nth GUI terminal (if a GUI terminal is running on screen n-1). On default, nothing is running on terminals 8 to 12, but you can run another server there.

`<Tab>`
(In a text terminal) Autocomplete the command if there is only one option, or else show all the available options. THIS SHORTCUT IS GREAT! It even works at LILO prompt!

`<ArrowUp>`
Scroll and edit the command history. Press <Enter> to execute.

`<Shift><PgUp>`
Scroll terminal output up. Work also at the login prompt, so you can scroll through your bootup messages.

`<Shift><PgDown>`
Scroll terminal output down.

`<Ctrl><Alt><+>`
(in X-windows) Change to the next X-server resolution (if you set up the X-server to more than one resolution). For multiple resolutions on my standard SVGA card/monitor, I have the following line in the file

`/etc/X11/XF86Config` (the first resolution starts on default, the largest determines the size of the "virtual screen"):
`Modes "1024x768" "800x600" "640x480" "512x384" "480x300" "400x300" "1152x864"`

`<Ctrl><Alt><->`
(in X-windows) Change to the previous X-server resolution.

`<Ctrl><Alt><BkSpc>`
(in X-windows) Kill the current X-windows server. Use if the X-windows server crushes and cannot be exited normally.

`<Ctrl><Alt><Del>`
Shut down the system and reboot. This is the normal shutdown command for a user at the text-mode console. Don't just press the "reset" button for shutdown!

`<Ctrl>c`
Kill the current process (mostly in the text mode for small applications).

`<Ctrl>d`
Log out from the current terminal. See also the next command.

`<Ctrl>d`
Send [End-of-File] to the current process. Don't press it twice else you also log out (see the previous command).

`<Ctrl>s`
Stop the transfer to the terminal.

`<Ctrl>q`
Resume the transfer to the terminal. Try if your terminal mysteriously stops responding.

`<Ctrl>z`
Send the current process to the background.

`exit`
Logout. I can also use `logout` for the same effect. (If you have started a second shell, e.g., using `bash` the second shell will be exited and you will be back in the first shell, not logged out.)

`reset`
Restore a screwed-up terminal (a terminal showing funny characters) to default setting. Use if you tried to "cat" a binary file. You may not be able to see the command as you type it.

`<MiddleMouseButton>`
Paste the text which is currently highlighted somewhere else. This is the normal "copy-paste" operation in Linux. (It doesn't work with Netscape and WordPerfect which use the MS Windows-style "copy-paste". It does work in the text terminal if you enabled "gpm" service using "setup".) Best used with a Linux-ready 3-button mouse (Logitech or similar) or else set "3-mouse button emulation").

`~`
(tilde) My home directory (normally the directory `/home/my_login_name`). For example, the command `cd ~/my_dir` will change my working directory to the subdirectory "*my_dir*" under my home directory. Typing just "cd" alone is an equivalent of the command "cd ~".

`.`
(dot) Current directory. For example, `./my_program` will attempt to execute the file "my_program" located in your current working directory.

`..`
(two dots) Directory parent to the current one. For example, the command `cd ..` will change my current working directory one one level up.

7.2 Common Linux commands--system info

`pwd`
Print working directory, i.e., display the name of my current directory on the screen.

`hostname`
Print the name of the local host (the machine on which you are working). Use `netconf` (as root) to change the name of the machine.

`whoami`
Print my login name.

`id` *`username`*
Print user id (uid) and his/her group id (gid), effective id (if different than the real id) and the supplementary groups.

`date`
Print or change the operating system date and time. E.g., I could change the date and time to 2000-12-31 23:57 using this command:
`date 123123572000`
To set the hardware (BIOS) clock from the system (Linux) clock, use the command (as root) `setclock`

`time`
Determine the amount of time that it takes for a process to complete + other info. Don't confuse it with the `date` command. E.g. I can find out how long it takes to display a directory content using:
`time ls`

`who`
Determine the users logged on the machine.

`rwho -a`
(=remote who) Determine all users logged on your network. The rwho service must be enabled for this command to run. If it isn't, run setup as root to enable "rwho".

`finger` *`user_name`*
System info about a user. Try: `finger root`

`last`
Show listing of users last logged-in on your system.

`history | more`
Show the last (1000 or so) commands executed from the command line on the current account. The "| more" causes the display to stop after each screenful.

`uptime`
Show the amount of time since the last reboot.

`ps`
(=print status) List the processes currently run by the current user.

`ps axu | more`
List all the processes currently running, even those without the controlling terminal, together with the name of the user that owns each process.

`top`
Keep listing the currently running processes, sorted by cpu usage (top users first). In KDE, you can get GUI-based Ktop from "K"menu under "System"-"Task Manager" (or by executing "ktop" in an X-terminal).

`uname -a`
(= Unix name with option "all") Info on your (local) server. I can also use `guname` (in X-window terminal) to display the info more nicely.

`free`
Memory info (in kilobytes).

`df -h`
(=disk free) Print disk info about all the filesystems (in human-readable form)

`du / -bh | more`
(=disk usage) Print detailed disk usage for each subdirectory starting at the "/" (root) directory (in human legible form).

`cat /proc/cpuinfo`
Cpu info--it show the content of the file `cpuinfo`. Note that the files in the `/proc` directory are not real files--they are hooks to look at information available to the kernel.

`cat /proc/interrupts`
List the interrupts in use.

`cat /proc/version`
Linux version and other info

`cat /proc/filesystems`
Show the types of filesystems currently in use.

`cat /etc/printcap`
Show the setup of printers.

`lsmod`
(As root. Use `/sbin/lsmod` to execute this command when you are a non-root user.) Show the kernel modules currently loaded.

`set|more`
Show the current user environment.

`echo $PATH`
Show the content of the environment variable "PATH". This command can be used to show other environment variables as well. Use "set" to see the full environment.

`dmesg | less`
Print kernel messages (the content of the so-called kernel ring buffer). Press "q" to quit "less". Use `less /var/log/dmesg` to see what "dmesg" dumped into this file right after the last system bootup.

7.3 Basic operations

`any_command` `--help |more`
Display a brief help on a command (works with most commands). "--help" works similar to DOS "/h" switch. The "more" pipe is needed if the output is longer than one screen.

`man` *`topic`*
Display the contents of the system manual pages (help) on the topic. Try `man man` first. Press "q" to quit the viewer. The command `info` *`topic`* works similar and may contain more up-to-date information. Manual pages can be hard to read. Try *`any_command`* `--help` for short, easy to digest help on a command. If more info needed, have a look to the directory `/usr/doc`. To display manual page from a specific section, I may use something like in this example: `man 3 exit` (this displays an info on the command `exit` from section 3 of the manual pages).

`apropos` *`topic`*
Give me the list of the commands that have something to to do with my topic.

`help command`
Display brief info on a bash (shell) build-in command.

`ls`
List the content of the current directory. Under Linux, the command "dir" is an alias to ls. Many users have "ls" to be an alias to "ls --color".

`ls -al |more`
List the content of the current directory, all files (also those starting with a dot), and in a long form. Pipe the output through the "more" command, so that the display pauses after each screenful.

`cd` *`directory`*
Change directory. Using "cd" without the directory name will take you to your home directory. "cd -" will take you to your previous directory and is a convenient way to toggle between two directories. "cd .." will take you one directory up.

`cp source destination`
Copy files. E.g., `cp /home/stan/existing_file_name .` will copy a file to my current working directory. Use the "-r" option (for recursive) to copy the contents of whole directories, e.g. , `cp -r my_existing/dir/ ~` will copy a subdirectory under my current working directory to my home directory.

`mcopy source destination`
Copy a file from/to a DOS filesystem (no mounting necessary). E.g., `mcopy a:\autoexec.bat ~/junk` . See `man mtools` for related commands: mdir, mcd, mren, mmove, mdel, mmd, mrd, mformat ....

`mv source destination`
Move or rename files. The same command is used for moving and renaming files and directories.

`ln source destination`
Create a hard link called *destination* to the file called *source*. The link appears as a copy of the original files, but in reality only one copy of the file is kept, just two (or more) directory entries point to it. Any changes the file are automatically visible throughout. When one directory entry is removed, the other(s) stay(s) intact. The limitation of the hard links are: the files have to be on the same filesystem, hard links to directories or special files are impossible.

`ln -s source destination`
Create a symbolic (soft) link called "destination" to the file called "source". The symbolic link just specifies a path where to look for the file. In contradistinction to hard links, the source and destination don't not have to tbe on the same filesystem. In comparison to hard links, the drawback of symbolic links are: if the original file is removed, the link is "broken", symbolic links can also create circular references (like circular references in spreadsheets or databases, e.g., "a" points to "b" and "b" points back to "a").

`rm files`
Remove (delete) files. You must own the file in order to be able to remove it. On many systems, you will be asked or confirmation of deleation, if you don't want this, use the "-f" (=force) option, e.g., `rm -f *` will remove all files in my current working directory, no questions asked.

`mkdir directory`
Make a new directory.

`rmdir directory`
Remove an empty directory.

`rm -r files`
(recursive remove) Remove files, directories, and their subdirectories. Careful with this command as root--you can easily remove all files on the system with such a command executed on the top of your directory tree, and there is no undelete in Linux (yet). But if you really wanted to do it (reconsider), here is how (as root): `rm -rf /*`

`cat filename | more`
View the content of a text file called "filename", one page a time. The "|" is the "pipe" symbol (on many American keyboards it shares the key with "\") The pipe makes the output stop after each screenful. For long files, it is sometimes convenient to use the commands head and tail that display just the beginning and the end of the file. If you happened to use "cat" a binary file and your terminal displays funny characters afterwards, you can restore it with the command "reset".

`less filename`
Scroll through a content of a text file. Press q when done. "Less" is roughly equivalent to "more" , the command you know from DOS, although very often "less" is more convenient than "more".

`pico filename`
Edit a text file using the simple and standard text editor called `pico`.

`pico -w filename`
Edit a text file, while disabling the long line wrap. Handy for editing configuration files, e.g. `/etc/fstab`.

`find / -name "filename"`
Find the file called "filename" on your filesystem starting the search from the root directory "/". The "filename" may contain wildcards (*,?).

`locate` *`filename`*
Find the file name of which contains the string "filename". Easier and faster than the previous command but depends on a database that normally rebuilds at night.

`./`*`program_name`*
Run an executable in the current directory, which is not on your PATH.

`touch` *`filename`*
Change the date/time stamp of the file *`filename`* to the current time. Create an empty file if the file does not exist.

`xinit`
Start a barebone X-windows server (without a windows manager).

`startx`
Start an X-windows server and the default windows manager. Works like typing "win" under DOS with Win3.1

`startx -- :1`
Start another X-windows session on the display 1 (the default is opened on display 0). You can have several GUI terminals running concurrently. Switch between them using <Ctrl><Alt><F7>, <Ctrl><Alt><F8>, etc.

`xterm`
(in X terminal) Run a simple X-windows terminal. Typing `exit` will close it. There are other, more advanced "virtual" terminals for X-windows. I like the popular ones: `konsole` and `kvt` (both come with kde) and `gnome-terminal` (comes with gnome). If you need something really fancy-looking, try `Eterm`.

`xboing`
(in X terminal). Very nice, old-fashioned game. Many small games/programs are probably installed on your system. I also like `xboard` (chess).

`shutdown -h now`
(as root) Shut down the system to a halt. Mostly used for a remote shutdown. Use <Ctrl><Alt><Del> for a shutdown at the console (which can be done by any user).

`halt`
`reboot`
(as root, two commands) Halt or reboot the machine. Used for remote shutdown, simpler to type than the previous command.

Network apps

`netscape`
(in X terminal) Run netscape (requires a separate Netscape installation). The current versions of Netscape (4.x) are known to be big and buggy. They occasionally crash by vanishing (no other harm done). Also, when not connected to the network , Netscape likes to refuse to do anything (looks like it hanged)-it revives when you connect.

`netscape -display` *`host`*`:0.0`
(in X terminal) Run netscape on the current machine and direct the output to machine named "host" display 0 screen 0. Your current machine must have a permission to display on the machine "host" (typically given by executing the command `xhost` *`current_machine_name`* in the xterminal of the machine host. Other X-windows program can be run remotely the same way.

`lynx` *`file.html`*
View an html file or browse the net from the text mode.

`pine`
A good text-mode mail reader. Another good and standard one is `elm`. Your Netscape mail will read the mail from your Internet account. `pine` will let you read the "local" mail, e.g. the mail your son or a cron process sends to you from a computer on your home network. The command `mail` could also be used for reading/composing mail, but it would be inconvenient--it is meant to be used in scripts for automation.

`elm`
A good tex-mode mail reader. See the previous command.

`mutt`
A really basic but extremally useful and fast mail reader.

`mail`
A basic operating system tool for e-mail. Look at the previous commands for a better e-mail reader. `mail` is good if you wanted to send an e-mail from a shell script.

`licq`
(in X term) An icq "instant messaging" client. Another good one is `kxicq`. Older distributions don't have an icq client installed, you have to do download one and install it.

`talk` *`username1`*
Talk to another user currently logged on your machine (or use "`talk` *`username1@machinename`*" to talk to a user on a different computer) . To accept the invitation to the conversation, type the command "`talk` *`username2`*". If somebody is trying to talk to you and it disrupts your work, your may use the command "`mesg n`" to refuse accepting messages. You may want to use "`who`" or "`rwho`" to determine the users who are currently logged-in.

`mc`
Launch the "Midnight Commander" file manager (looks like "Norton Commander" for Linux).

`telnet` *`server`*
Connect to another machine using the TELNET protocol. Use a remote machine name or IP address. You will be prompted for your login name and password--you must have an account on the remote machine to login. Telnet will connect you to another machine and let you operate on it as if you were sitting at its keyboard (almost). Telnet is not very secure--everything you type goes in open text, even your password!

`rlogin` *`server`*
(=remote login) Connect to another machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`rsh` *`server`*
(=remote shell) Yet another way to connect to a remote machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`ftp` *`server`*
Ftp another machine. (There is also `ncftp` which adds extra features and `gftp` for GUI .) Ftp is good for copying files to/from a remote machine. Try user "anonymous" if you don't have an account on the remote server. After connection, use "?" to see the list of available ftp commands. The essential ftp command are: `ls` (see the files on the remote system), `ASCII`, `binary` (set the file transfer mode to either text or binary, important that you select the proper one), `get` (copy a file from the remote system to the local system), `mget` (get many files at once), `put` (copy a file from the local system to the remote system), `mput` (put many files at once), `bye` (disconnect). For automation in a script, you may want to use `ncftpput` and `ncftpget`, for example:

```
ncftpput -u my_user_name -p my_password -a remote.host.domain remote_dir
*local.html
```

`minicom`
Minicom program (looks like "Procomm for Linux").

File (de)compression

`tar -zxvf` *`filename.tar.gz`*
(=tape archiver) Untar a tarred and compressed tarball (*.tar.gz or *.tgz) that you downloaded from the Internet.

`tar -xvf` *`filename.tar`*
Untar a tarred but uncompressed tarball (*.tar).

`gunzip` *`filename.gz`*
Decompress a zipped file (*.gz" or *.z). Use gzip (also `zip` or `compress`) if you wanted to compress files to this file format.

`bunzip2` *`filename.bz2`*
(=big unzip) Decompress a file (*.bz2) zipped with bzip2 compression utility. Used for big files.

`unzip filename.zip`
Decompress a file (*.zip) zipped with a compression utility compatible with PKZIP for DOS.

`unarj e filename.arj`
Extract the content of an *.arj archive.

`uudecode -o outputfile filename`
Decode a file encoded with `uuencode`. uu-encoded files are typically used for transfer of non-text files in e-mail (uuencode transforms any file into an ASCII file).

7.4 Process control

`ps`
(=print status) Display the list of currently running processes with their process IDs (PID) numbers. Use `ps axu` to see all processes currently running on your system (also those of other users or without a controlling terminal), each with the name of the owner. Use "top" to keep listing the processes currently running.

`fg PID`
Bring a background or stopped process to the foreground.

`bg PID`
Send the process to the background. Opposite to fg. The same can be accomplished with <Ctrl>z. If you have stopped jobs, you have to type `exit` twice in row to log out.

`any_command&`
Run any command in the background (the symbol "&" means "run the proceeding command in the background").

`batch any_command`
Run any command (usually one that is going to take more time) when the system load is low. I can logout, and the process will keep running.

`at 17:00`
Execute a command at a specified time. You will be prompted for the command(s) to run, until you press <Ctrl>d.

`kill PID`
Force a process shutdown. First determine the PID of the process to kill using ps.

`killall program_name`
Kill program(s) by name.

`xkill`
(in an xwindow terminal) Kill a GUI-based program with mouse. (Point with your mouse cursor at the window of the process you want to kill and click.)

`lpc`
(as root) Check and control the printer(s). Type "?" to see the list of available commands.

`lpq`
Show the content of the printer queue. Under KDE (X-Windows), you may use GUI-based "Printer Queue" available from "K"menu-Utilities.

`lprm job_number`
Remove a printing job "job_number" from the queue.

`nice program_name`
Run *program_name* adjusting its priority. Since the priority is not specified in this example, it will be adjusted by 10 (the process will run slower), from the default value (usually 0). The lower the number (of "niceness" to other users on the system), the higher the priority. The priority value may be in the range -20 to 19. Only root may specify negative values. Use "top" to display the priorities of the running processes.

`renice -1 PID`
(as root) Change the priority of a running process to -1. Normal users can only adjust processes they own, and only up from the current value (make them run slower).

<Ctrl>c, <Ctrl>z, <Ctrl>s, and <Ctrl>q also belong to this chapter but they were described previously. In short they mean: stop the current command, send the current command to the background, stop the data transfer, resume the data transfer.

7.5 Basic administration commands

`printtool`
(as root in X-terminal) Configuration tool for your printer(s). Settings go to the file `/etc/printcap`.

`setup`
(as root) Configure mouse, soundcard, keyboard, X-windows, system services. There are many distibution-specific configuration utilities, `setup` is the default on RedHat. Mandrake 7.0 offers very nice `DrakConf` .

`linuxconfig`
(as root, either in text or graphical mode). You can access and change hundreds of setting from it. Very powerful--don't change too many things at the same time, and be careful with changing entries you don't understand.

`xvidtune`
(in X-terminal). Adjust the settings of the graphical display for all resolutions so as to eliminate black bands, shift the display right/left/up/down, etc. (First use the knobs on your monitor to fit your text mode correctly on the screen.) To make the changes permanent, display the frequencies on the screen and transfer them to the setup file `/etc/X11/XF86Config`.

`alias ls="ls --color=tty"`
Create an alias for the command "ls" to enhance its format with color. In this example, the alias is also called "ls" and the "color" option is only envoke when the output is done to a terminal (not to files). Put the alias into the file `/etc/bashrc` if you would like the alias to be always accessible to all users on the system. Type "`alias`" alone to see the list of aliases on your system.

`adduser` *`user_name`*
Create a new account (you must be root). E.g., `adduser barbara` Don't forget to set up the password for the new user in the next step. The user home directory is `/home/`*`user_name`*.

`useradd` *`user_name`*
The same as the command " `adduser` *`user_name`* ".

`userdel` *`user_name`*
Remove an account (you must be a root). The user's home directory and the undelivered mail must be dealt with separately (manually because you have to decide what to do with the files).

`groupadd` *`group_name`*
Create a new group on your system. Non-essential but can be handy even on a home machine with a small number of users.

`passwd`
Change the password on your current account. If you are root, you can change the password for any user using:
`passwd` *`user_name`*

`chmod perm filename`
(=change mode) Change the file access permission for the files you own (unless you are root in which case you can change any file). You can make a file accessible in three modes: read (r), write (w), execute (x) to three classes of users: owner (u), members of the same group as the owner (g), others on the system (o). Check the current access permissions using:
`ls -l` *`filename`*
If the file is accessible to all users in all modes it will show:
`rwxrwxrwx`
The first triplet shows the file permission for the owner of the file, the second for his/her group, the third for others. A "no" permission is shown as "-".
E.g., this command will **add** the permission to read the file "junk" to all (=user+group+others):
`chmod a+r junk`
This command will remove the permission to execute the file junk from others:

`chmod o-x junk`
Also try here for more info.
You can set the default file permissions for the news files that you create using the command `umask` (see `man umask`).

`chown new_ownername filename`
`chgrp new_groupname filename`
Change the file owner and group. You should use these two commands after you copy a file for use by somebody else.

`su`
(=substitute user id) Assume the superuser (=root) identity (you will be prompted for the password). Type "exit" to return you to your previous login. Don't habitually work on your machine as root. The root account is for administration and the su command is to ease your access to the administration account when you require it. You can also use "su" to assume any other user identity, e.g. `su barbara` will make me "barbara" (password required unless I am a superuser).

`kernelcfg`
(as root in X terminal). GUI to to add/remove kernel modules. You can do the same from the command line using the command "`insmod`", but "`insmode`" is less "newbie-friendly".

`lsmod`
List currently loaded kernel modules. A module is like a device driver--it provides operating system kernel support for a particular piece of hardware or feature.

`modprobe -l |more`
List all the modules available for your kernel. The available modules are determined by how your Linux kernel was compliled. Every possible module/feature can be compiled on linux as either "hard wired" (fast, non-removable), "module" (maybe slower, but loaded/removable on demand), or "no" (no support for this feature at all).

`insmod parport`
`insmod ppa`
(as root) Insert modules into the kernel (a module is roughly an equivalent of a DOS device driver). This example shows how to insert the modules for support of the external parallel port zip drive (it appears to be a problem to get the external zip drive to work in any other way under RH6.0).

`rmmod module_name`
(as root, not essential). Remove the module *module_name* from the kernel.

`setserial /dev/cua0 port 0x03f8 irq 4`
(as root) Set a serial port to a non-standard setting. The example here shows the standard setting for the first serial port (cua0 or ttyS0). The standard PC settings for the second serial port (cua1or ttyS1) are: address of i/o port 0x02f8, irq 3. The third serial port (cua2 or ttyS2): 0x03e8, irq 4. The forth serial port (cua3 or ttyS3): 0x02e8, irq 3. Add your setting to `/etc/rc.d/rc.local` if you want it to be set at the boot time. See man setserial for good a overview.

`fdisk`
(as root) Linux hard drive partitioning utility (DOS has a utility with the same name).

`cd /usr/src/linux-2.0.36`
`make xconfig`
(as root in X terminal). Nice GUI front-end for configuration of the kernel options in preparation for compilation of your customized kernel. (The directory name contains the version of your Linux kernel so you may need to modify the directory name if your Linux kernel version is different than 2.0.36 used in this example. You also need the "Tk" interpreter and the kernel source code installed.) The alternatives to "make xconfig" are: "make config" (runs a scripts that asks you questions in the text mode) and "make menuconfig" (runs a text-based menu-driven configuration utility). Try: `less /usr/doc/HOWTO/Kernel-HOWTO` for more information.
After the configuration, you may choose to proceed with kernel compilation of the new kernel by issuing the following commands:
`make dep`
`make zImage`
The last command will take some time to complete (maybe 0.5 h, depending on your hardware). It produces the file "zImage", which is your new Linux kernel. Next:

`make modules`
`make modules_install`
Read: /usr/doc/HOWTO/Kernel-HOWTO for information on how to install the new kernel. You will probably also find it useful to read "man depmode". Configuration, compilation and installation of a new kernel is not difficult but it CAN lead to problems if you don't know what you are doing.
Compilation of a kernel is a good way to test your hardware, because it involves a massive amount of computing. If your hardware is "flaky", you will most likely receive the "signal 11" error (read the beatiful `/usr/doc/FAQ/txt/GCC-SIG11-FAQ`). **See this for details on kernel upgrade.**

`depmod -a`
(as root) Build the module dependency table for the kernel. This can, for example, be useful after installing and booting a new kernel. Use "`modprobe -a`" to load the modules.

`ldconfig`
(as root) Re-create the bindings and the cache for the loader of dynamic libraries ("ld"). You may want to run ldconfig after an installation of new dynamically linked libraries on your system. (It is also re-run every time you boot the computer, so if you reboot you don't have to run it manually.)

`mknod /dev/fd0 b 2 0`
(=make node, as root) Create a device file. This example shows how to create a device file associated with your first floppy drive and could be useful if you happened to accidentally erase it. The options are: b=block mode device (c=character mode device, p=FIFO device, u=unbuffered character mode device). The two integers specify the major and the minor device number.

`fdformat /dev/fd0H1440`
`mkfs -c -t ext2`
(=floppy disk format, two commands, as root) Perform a low-level formatting of a floppy in the first floppy drive (/dev/fd0), high density (1440 kB). Then make a Linux filesystem (-t ext2), checking/marking bad blocks (-c). Making the files system is an equivalent to the high-level format.

`badblocks /dev/fd01440 1440`
(as root) Check a high-density floppy for bad blocks and display the results on the screen. The parameter "1440" specifies that 1440 blocks are to be checked. This command does not modify the floppy.

`fsck -t ext2 /dev/hda2`
(=file system check, as root) Check and repair a filesystem. The example uses the partition hda2, filesystem type ext2.

`dd if=/dev/fd0H1440 of=floppy_image`
`dd if=floppy_image of=/dev/fd0H1440`
(two commands, dd="data duplicator") Create an image of a floppy to the file called "floppy_image" in the current directory. Then copy `floppy_image` (file) to another floppy disk. Works like DOS "DISKCOPY".

Program installation

`rpm -ivh` *`filename.rpm`*
(=RedhatPackageManager, install, verbose, hashes displayed to show progress, as root.) Install a content of RedHat rpm package(s) and print info on what happened. Keep reading if you prefer a GUI installation.

`rpm -qpi` *`filename.rpm`*
(=RedhatPackageManager, query, package, list.) Read the info on the content of a yet uninstalled package *filename.rpm*.

`rpm -qpl` *`filename.rpm`*
(=RedhatPackageManager, query, package, information.) List the files contained in a yet uninstalled package *filename.rpm*.

`rpm -qf` *`filename`*
(=RedhatPackageManager, query, file.) Find out the name of the *.rpm package to which the file *filename* (on your hardrive) belongs.

`rpm -e packagename`
(=RedhatPackageManager, erase=uninstall.) Uninstall a package *pagckagename*. *Packagname* is the same as the beginning of the *.rpm package file but without the dash and version number.

`kpackage`
`gnorpm`
`glint`
(in X terminal, as root if you want to be able to install packages) GUI fronts to the Red Hat Package Manager (rpm). "glint" comes with RH5.2, "gnorpm" with RH6.0, "kpackage" comes with RH6.1 or must be installed separately but is the best of the three. Use any of them to view which software packages are installed on your system and the what not-yet-installed packages are available on your RedHat CD, display the info about the packages, and install them if you want (installation must be done as root).

Accessing drives/partitions

`mount`
See here for details on mounting drives. Examples are shown in the next commands.

`mount -t auto /dev/fd0 /mnt/floppy`
(as root) Mount the floppy. The directory `/mnt/floppy` must exist, be empty and NOT be your current directory.

`mount -t auto /dev/cdrom /mnt/cdrom`
(as root) Mount the CD. You may need to create/modify the `/dev/cdrom` file depending where your CDROM is. The directory `/mnt/cdrom` must exist, be empty and NOT be your current directory.

`mount /mnt/floppy`
(as user or root) Mount a floppy as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/floppy` must not be your current directory.

`mount /mnt/cdrom`
(as user or root) Mount a CD as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/cdrom` must not be your current directory.

`umount /mnt/floppy`
Unmount the floppy. The directory `/mnt/floppy` must not be your (or anybody else's) current working directory. Depending on your setup, you might not be able to unmount a drive that you didn't mount.

7.6 Network administration tools

`netconf`
(as root) A very good menu-driven setup of your network.

`ping`*`machine_name`*
Check if you can contact another machine (give the machine's name or IP), press <Ctrl>C when done (it keeps going).

`route -n`
Show the kernel routing table.

`nslookup` *`host_to_find`*
Query your default domain name server (DNS) for an Internet name (or IP number) *host_to_find*. This way you can check if your DNS works. You can also find out the name of the host of which you only know the IP number.

`traceroute` *`host_to_trace`*
Have a look how you messages trave to *`host_to_trace`* (which is either a host name or IP number).

`ipfwadm -F -p m`
(for RH5.2, seen next command for RH6.0) Set up the firewall IP forwarding policy to masquerading. (Not very secure but simple.) Purpose: all computers from your home network will appear to the outside world as one very busy machine and, for example, you will be allowed to browse the Internet from all computers at once.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipfwadm-wrapper -F -p deny
ipfwadm-wrapper -F -a m -S xxx.xxx.xxx.0/24 -D 0.0.0.0/0
```
(three commands, RH6.0). Does the same as the previous command. Substitute the "x"s with digits of your class "C" IP address that you assigned to your home network. See here for more details. In RH6.1, masquarading seems broken to me--I think I will install Mandrake Linux:).

`ifconfig`
(as root) Display info on the network interfaces currently active (ethernet, ppp, etc). Your first ethernet should show up as eth0, second as eth1, etc, first ppp over modem as ppp0, second as ppp1, etc. The "lo" is the "loopback only" interface which should be always active. Use the options (see `ifconfig --help`) to configure the interfaces.

`ifup` *`interface_name`*
(`/sbin/ifup` to it run as a user) Startup a network interface. E.g.:
```
ifup eth0
ifup ppp0
```
Users can start up or shutdown the ppp interface only when the right permission was checked during the ppp setup (using `netconf`). To start a ppp interface (dial-up connection), I normally use kppp available under kde menu "internet".

`ifdown` *`interface_name`*
(`/sbin/ifdown` to run it as a user). Shut down the network interface. E.g.: `ifdown ppp0` Also, see the previous command.

`netstat | more`
Displays a lot (too much?) information on the status of your network.

Music-related commands

`cdplay play 1`
Play the first track from a audio CD.

`eject`
Get a free coffee cup holder :))). (Eject the CD ROM tray).

`play` *`my_file.wav`*
Play a wave file.

`mpg123` *`my_file.mp3`*
Play an mp3 file.

`mpg123 -w` *`my_file.wav my_file.mp3`*
Create a wave audio file from an mp3 audio file.

`knapster`
(in X terminal) Start the program to downolad mp3 files that other users of napster have displayed for downloading. Really cool!

`cdparanoia -B "1-"`
(CD ripper) Read the contents of an audio CD and save it into wavefiles in the current directories, one track per wavefile. The "1-"
means "from track 1 to the last". -B forces putting each track into a separate file.

`playmidi` *`my_file.mid`*
Play a midi file. `playmidi -r` *`my_file.mid`* will display text mode effects on the screen.

`sox`
(argument not given here) Convert from almost any audio file format to another (but not mp3s). See `man sox`.

Graphics-related commands

`kghostview` *`my_file.ps`*
Display a postscript file on screen. I can also use the older-looking `ghostview` or `gv` for the same end effect.

`ps2pdf my_file.ps my_file.pdf`
Make a pdf (Adobe portable document format) file from a postscript file.

`gimp`
(in X terminal) A humble looking but very powerful image processor. Takes some learning to use, but it is great for artists, there is almost nothing you can't do with gimp. Use your mouse right button to get local menus, and learn how to use layers. Save your file in the native gimp file format *.xcf (to preserve layers) and only then flatten it and save as png (or whatever). There is a large user manual /usr/

`gphoto`
(in X terminal) Powerful photo editor.

`giftopnm` *`my_file.giff > my_file.pnm`*
`pnmtopng` *`my_file.pnm > my_file.png`*
Convert the propriatory giff graphics into a raw, portable pnm file. Then convert the pnm into a png file, which is a newer and better standard for Internet pictures (better technically plus there is no danger of being sued by the owner of giff patents).

Dumping a mySQL to a sql file

```
mysqldump -l --opt databasename > /root/file/location/filename.sql  -u user
--password=whateverthepass
```

Importing mySQL dump file

```
mysql databasename < /root/file/location/filename.sql -u user --password=whateverthepass
```

Copying Entire Folder of Files

```
cp - Ru /root/file/location/* /where/it/should/go --reply=yes
```

Making a tgz archive of an entire folder for FTP export

```
tar zcf localfolder.tgz localfolder/
```

Copying an entire folder to another server

```
tar zcf - localfolder/ \ | ssh 192.1.1.1 "cd folder/to/copy/to; tar zpxvf -"
```

If you want to extract one file from the .tar.gz file use

```
gzip -dc file.tar.gz | tar xf - pathname/filename
The pathname and filename should be exactly as given in the .tar.gz file. If you want more than
one file append their names, again include pathname, at the end of the command.
```

Symlinking

```
Source > Location of symlink
ln -s /source/foldername /folder/where/symlink/will/be
```

Set the rights

```
chmod -R 775 /foldername
chown -R username:groupname /foldername
```

[edit] A-Z index of Linux BASH commands

```
alias    Create an alias
awk      Find and Replace text within file(s)

break    Exit from a loop
builtin  Run a shell builtin

cal      Display a calendar
case     Conditionally perform a command
cat      Display the contents of a file
cd       Change Directory
chgrp    Change group ownership
chmod    Change access permissions
chown     Change file owner and group
chroot   Run a command with a different root directory
cksum    Print CRC checksum and byte counts
clear    Clear terminal screen
cmp      Compare two files
comm     Compare two sorted files line by line
command  Run a command - ignoring shell functions
continue Resume the next iteration of a loop
cp       Copy one or more files to another location
cron     Daemon to execute scheduled commands
crontab  Schedule a command to run at a later time
csplit   Split a file into context-determined pieces
cut      Divide a file into several parts

date     Display or change the date & time
dc       Desk Calculator
```

```
dd       Data Dump - Convert and copy a file
declare  Declare variables and give them attributes
df       Display free disk space
diff     Display the differences between two files
diff3    Show differences among three files
dir      Briefly list directory contents
dircolors Colour setup for `ls'
dirname  Convert a full pathname to just a path
dirs     Display list of remembered directories
du       Estimate file space usage

echo     Display message on screen
ed       A line-oriented text editor (edlin)
egrep    Search file(s) for lines that match an extended expression
eject    Eject CD-ROM
enable   Enable and disable builtin shell commands
env      Display, set, or remove environment variables
eval     Evaluate several commands/arguments
exec     Execute a command
exit     Exit the shell
expand   Convert tabs to spaces
export   Set an environment variable
expr     Evaluate expressions

factor   Print prime factors
false    Do nothing, unsuccessfully
fdformat Low-level format a floppy disk
fdisk    Partition table manipulator for Linux
fgrep    Search file(s) for lines that match a fixed string
find     Search for files that meet a desired criteria
fmt      Reformat paragraph text
fold     Wrap text to fit a specified width.
for      Expand words, and execute commands
format   Format disks or tapes
free     Display memory usage
fsck     Filesystem consistency check and repair.
function Define Function Macros

gawk     Find and Replace text within file(s)
getopts  Parse positional parameters
grep     Search file(s) for lines that match a given pattern
groups   Print group names a user is in
gzip     Compress or decompress named file(s)

hash     Remember the full pathname of a name argument
head     Output the first part of file(s)history  Command History
hostname Print or set system name

id       Print user and group id's
if       Conditionally perform a command
import   Capture an X server screen and save the image to file
info     Help info
install  Copy files and set attributes

join     Join lines on a common field

kill     Stop a process from running

less     Display output one screen at a time
let      Perform arithmetic on shell variables
ln       Make links between files
local    Create variables
locate   Find files
logname  Print current login name
logout   Exit a login shell
```

```
lpc      Line printer control program
lpr      Off line print
lprint   Print a file
lprintd  Abort a print job
lprintq  List the print queue
lprm     Remove jobs from the print queue
ls       List information about file(s)

m4       Macro processor
man      Help manual
mkdir    Create new folder(s)
mkfifo   Make FIFOs (named pipes)
mknod    Make block or character special files
more     Display output one screen at a time
mount    Mount a file system
mtools   Manipulate MS-DOS files
mv       Move or rename files or directories

nice     Set the priority of a command or job
nl       Number lines and write files
nohup    Run a command immune to hangups

passwd   Modify a user password
paste    Merge lines of files
pathchk  Check file name portability
popd     Restore the previous value of the current directory
pr       Convert text files for printing
printcap Printer capability database
printenv Print environment variables
printf   Format and print data
ps       Process status
pushd    Save and then change the current directory
pwd      Print Working Directory

quota    Display disk usage and limits
quotacheck Scan a file system for disk usage
quotactl Set disk quotas

ram      ram disk device
rcp      Copy files between two machines.
read     read a line from standard input
readonly Mark variables/functions as readonly
remsync  Synchronize remote files via email
return   Exit a shell function
rm       Remove files
rmdir    Remove folder(s)
rpm      Remote Package Manager
rsync    Remote file copy (Synchronize file trees)

screen   Terminal window manager
sdiff    Merge two files interactively
sed      Stream Editor
select   Accept keyboard input
seq      Print numeric sequences
set      Manipulate shell variables and functions
shift    Shift positional parameters
shopt    Shell Options
shutdown Shutdown or restart linux
sleep    Delay for a specified time
sort     Sort text files
source   Run commands from a file `.'
split    Split a file into fixed-size pieces
su       Substitute user identity
sum      Print a checksum for a file
symlink  Make a new name for a file
sync     Synchronize data on disk with memory
```

```
tac       Concatenate and write files in reverse
tail      Output the last part of files
tar       Tape ARchiver
tee       Redirect output to multiple files
test      Evaluate a conditional expression
time      Measure Program Resource Use
times     User and system times
touch     Change file timestamps
top       List processes running on the system
traceroute Trace Route to Host
trap      Run a command when a signal is set(bourne)
tr        Translate, squeeze, and/or delete characters
true      Do nothing, successfully
tsort     Topological sort
tty       Print filename of terminal on stdin
type      Describe a command

ulimit    Limit user resources
umask     Users file creation mask
umount    Unmount a device
unalias   Remove an alias
uname     Print system information
unexpand Convert spaces to tabs
uniq      Uniquify files
units     Convert units from one scale to another
unset     Remove variable or function names
unshar    Unpack shell archive scripts
until     Execute commands (until error)
useradd   Create new user account
usermod   Modify user account
users     List users currently logged in
uuencode Encode a binary file uudecode Decode a file created by uuencode

v         Verbosely list directory contents (`ls -l -b')
vdir      Verbosely list directory contents (`ls -l -b')

watch     Execute/display a program periodically
wc        Print byte, word, and line counts
whereis   Report all known instances of a command
which     Locate a program file in the user's path.
while     Execute commands
who       Print all usernames currently logged in
whoami    Print the current user id and name (`id -un')

xargs     Execute utility, passing constructed argument list(s)
yes       Print a string until interrupted

.period   Run commands from a file
###       Comment / Remark
```

This is a linux command line reference for common operations.
Examples marked with • are valid/safe to paste without modification into a terminal, so you may want to keep a terminal window open while reading this so you can cut & paste.
All these commands have been tested both on Fedora and Ubuntu.

Command	Description
• apropos whatis	Show commands pertinent to string. See also threadsafe
• man -t man | ps2pdf - > man.pdf	make a pdf of a manual page
which command	Show full path name of command
time command	See how long a command takes
• time cat	Start stopwatch. Ctrl-d to stop. See also sw
• nice info	Run a low priority command (The "info" reader in this case)
• renice 19 -p $$	Make shell (script) low priority. Use for non interactive tasks
dir navigation	
• cd -	Go to previous directory
• cd	Go to $HOME directory
(cd dir && command)	Go to dir, execute command and return to current dir
• pushd **.**	Put current dir on stack so you can **popd** back to it
file searching	
• alias l='ls -l --color=auto'	quick dir listing
• ls -lrt	List files by date. See also newest and find_mm_yyyy
• ls /usr/bin | pr -T9 -W$COLUMNS	Print in 9 columns to width of terminal
find -name '*.[ch]' | xargs grep -E 'expr'	Search 'expr' in this dir and below. See also findrepo
find -type f -print0 | xargs -r0 grep -F 'example'	Search all regular files for 'example' in this dir and below
find -maxdepth 1 -type f | xargs grep -F 'example'	Search all regular files for 'example' in this dir
find -maxdepth 1 -type d | while read dir; do echo $dir; echo cmd2; done	Process each item with multiple commands (in while loop)
• find -type f ! -perm -444	Find files not readable by all (useful for web site)
• find -type d ! -perm -111	Find dirs not accessible by all (useful for web site)
• locate -r 'file[^/]*\.txt'	Search cached index for names. This re is like glob *file*.txt
• look reference	Quickly search (sorted) dictionary for prefix

• grep --color reference /usr/share/dict/words	Highlight occurances of regular expression in dictionary

archives and compression

gpg -c file	Encrypt file
gpg file.gpg	Decrypt file
tar -c dir/ | bzip2 > dir.tar.bz2	Make compressed archive of dir/
bzip2 -dc dir.tar.bz2 | tar -x	Extract archive (use gzip instead of bzip2 for tar.gz files)
tar -c dir/ | gzip | gpg -c | ssh user@remote 'dd of=dir.tar.gz.gpg'	Make encrypted archive of dir/ on remote machine
find dir/ -name '*.txt' | tar -c --files-from=- | bzip2 > dir_txt.tar.bz2	Make archive of subset of dir/ and below
find dir/ -name '*.txt' | xargs cp -a --target-directory=dir_txt/ --parents	Make copy of subset of dir/ and below
(tar -c /dir/to/copy) | (cd /where/to/ && tar -x -p)	Copy (with permissions) copy/ dir to /where/to/ dir
(cd /dir/to/copy && tar -c .) | (cd /where/to/ && tar -x -p)	Copy (with permissions) contents of copy/ dir to /where/to/
(tar -c /dir/to/copy) | ssh -C user@remote 'cd /where/to/ && tar -x -p'	Copy (with permissions) copy/ dir to remote:/where/to/ dir
dd bs=1M if=/dev/sda | gzip | ssh user@remote 'dd of=sda.gz'	Backup harddisk to remote machine

rsync (Network efficient file copier: Use the --dry-run option for testing)

rsync -P rsync://rsync.server.com/path/to/file file	Only get diffs. Do multiple times for troublesome downloads
rsync --bwlimit=1000 fromfile tofile	Locally copy with rate limit. It's like nice for I/O
rsync -az -e ssh --delete ~/public_html/ remote.com:'~/public_html'	Mirror web site (using compression and encryption)
rsync -auz -e ssh remote:/dir/ . && rsync -auz -e ssh . remote:/dir/	Synchronize current directory with remote one

ssh (Secure SHell)

ssh $USER@$HOST command	Run command on $HOST as $USER (default command=shell)
• ssh -f -Y $USER@$HOSTNAME xeyes	Run GUI command on $HOSTNAME as $USER
scp -p -r $USER@$HOST: file dir/	Copy with permissions to $USER's home directory on $HOST
ssh -g -L 8080:localhost:80 root@$HOST	Forward connections to $HOSTNAME:8080 out to $HOST:80
ssh -R 1434:imap:143 root@$HOST	Forward connections from $HOST:1434 in to imap:143

wget (multi purpose download tool)

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

networking (Note ifconfig, route, mii-tool, nslookup commands are obsolete)

ethtool eth0	Show status of ethernet interface eth0
ethtool --change eth0 autoneg off speed 100 duplex full	Manually set ethernet interface speed
iwconfig eth1	Show status of wireless interface eth1
iwconfig eth1 rate 1Mb/s fixed	Manually set wireless interface speed
• iwlist scan	List wireless networks in range
• ip link show	List network interfaces
ip link set dev eth0 name wan	Rename interface eth0 to wan
ip link set dev eth0 up	Bring interface eth0 up (or down)
• ip addr show	List addresses for interfaces
ip addr add 1.2.3.4/24 brd + dev eth0	Add (or del) ip and mask (255.255.255.0)
• ip route show	List routing table
ip route add default via 1.2.3.254	Set default gateway to 1.2.3.254
• tc qdisc add dev lo root handle 1:0 netem delay 20msec	Add 20ms latency to loopback device (for testing)
• tc qdisc del dev lo root	Remove latency added above
• host pixelbeat.org	Lookup DNS ip address for name or vice versa
• hostname -i	Lookup local ip address (equivalent to host `hostname`)
• whois pixelbeat.org	Lookup whois info for hostname or ip address
• netstat -tupl	List internet services on a system
• netstat -tup	List active connections to/from system

windows networking (Note samba is the package that provides all this windows specific networking support)

• smbtree	Find windows machines. See also findsmb

nmblookup -A 1.2.3.4	Find the windows (netbios) name associated with ip address
smbclient -L windows_box	List shares on windows machine or samba server
mount -t smbfs -o fmask=666,guest //windows_box/share /mnt/share	Mount a windows share
echo 'message' | smbclient -M windows_box	Send popup to windows machine (off by default in XP sp2)

text manipulation (Note sed uses stdin and stdout. Newer versions support inplace editing with the -i option)

sed 's/string1/string2/g'	Replace string1 with string2
sed 's/\(.*\)1/\12/g'	Modify anystring1 to anystring2
sed '/ *#/d; /^ *$/d'	Remove comments and blank lines
sed ':a; /\\$/N; s/\\\n//; ta'	Concatenate lines with trailing \
sed 's/[\t]*$//'	Remove trailing spaces from lines
sed 's/\([`"$\]\)/\\\1/g'	Escape shell metacharacters active within double quotes
• seq 10 | sed "s/^/ /; s/ *\(.\{7,\}\)/\1/"	Right align numbers
sed -n '1000{p;q}'	Print 1000th line
sed -n '10,20p;20q'	Print lines 10 to 20
sed -n 's/.*<title>\(.*\)<\/title>.*/\1/ip;T;q'	Extract title from HTML web page
sed -i 42d ~/.ssh/known_hosts	Delete a particular line
sort -t. -k1,1n -k2,2n -k3,3n -k4,4n	Sort IPV4 ip addresses
• echo 'Test' | tr '[:lower:]' '[:upper:]'	Case conversion
• tr -dc '[:print:]' < /dev/urandom	Filter non printable characters
• history | wc -l	Count lines

set operations (Note you can export LANG=C for speed. Also these assume no duplicate lines within a file)

sort file1 file2 | uniq	Union of unsorted files
sort file1 file2 | uniq -d	Intersection of unsorted files
sort file1 file1 file2 | uniq -u	Difference of unsorted files
sort file1 file2 | uniq -u	Symmetric Difference of unsorted files
join -t'\0' -a1 -a2 file1 file2	Union of sorted files
join -t'\0' file1 file2	Intersection of sorted files
join -t'\0' -v2 file1 file2	Difference of sorted files
join -t'\0' -v1 -v2 file1 file2	Symmetric Difference of sorted files

math

• echo '(1 + sqrt(5))/2' | bc -l	Quick math (Calculate φ). See also bc

- `echo 'pad=20; min=64; (100*10^6)/((pad+min)*8)' | bc` — More complex (int) e.g. This shows max FastE packet rate
- `echo 'pad=20; min=64; print (100E6)/((pad+min)*8)' | python` — Python handles scientific notation
- `echo 'pad=20; plot [64:1518] (100*10**6)/((pad+x)*8)' | gnuplot -persist` — Plot FastE packet rate vs packet size
- `echo 'obase=16; ibase=10; 64206' | bc` — Base conversion (decimal to hexadecimal)
- `echo $((0x2dec))` — Base conversion (hex to dec) ((shell arithmetic expansion))
- `units -t '100m/9.58s' 'miles/hour'` — Unit conversion (metric to imperial)
- `units -t '500GB' 'GiB'` — Unit conversion (SI to IEC prefixes)
- `units -t '1 googol'` — Definition lookup
- `seq 100 | (tr '\n' +; echo 0) | bc` — Add a column of numbers. See also add and funcpy

calendar

- `cal -3` — Display a calendar
- `cal 9 1752` — Display a calendar for a particular month year
- `date -d fri` — What date is it this friday. See also day
- `[ $(date -d "tomorrow" +%d) = "01" ] || exit` — exit a script unless it's the last day of the month
- `date --date='25 Dec' +%A` — What day does xmas fall on, this year
- `date --date='@2147483647'` — Convert seconds since the epoch (1970-01-01 UTC) to date
- `TZ='America/Los_Angeles' date` — What time is it on west coast of US (use tzselect to find TZ)
- `date --date='TZ="America/Los_Angeles" 09:00 next Fri'` — What's the local time for 9AM next Friday on west coast US
- `echo "mail -s 'get the train' P@draigBrady.com < /dev/null" | at 17:45` — Email reminder
- `echo "DISPLAY=$DISPLAY xmessage cooker" | at "NOW + 30 minutes"` — Popup reminder

locales

- `printf "%'d\n" 1234` — Print number with thousands grouping appropriate to locale
- `BLOCK_SIZE=\'1 ls -l` — get ls to do thousands grouping appropriate to locale
- `echo "I live in `locale territory`"` — Extract info from locale database
- `LANG=en_IE.utf8 locale int_prefix` — Lookup locale info for specific country. See also ccodes
- `locale | cut -d= -f1 | xargs locale -kc | less` — List fields available in locale database

recode (Obsoletes iconv, dos2unix, unix2dos)

- `recode -l | less` — Show available conversions (aliases on each line)

recode windows-1252.. file_to_change.txt	Windows "ansi" to local charset (auto does CRLF conversion)
recode utf-8/CRLF.. file_to_change.txt	Windows utf8 to local charset
recode iso-8859-15..utf8 file_to_change.txt	Latin9 (western europe) to utf8
recode ../b64 < file.txt > file.b64	Base64 encode
recode /qp.. < file.txt > file.qp	Quoted printable decode
recode ..HTML < file.txt > file.html	Text to HTML
• recode -lf windows-1252 | grep euro	Lookup table of characters
• echo -n 0x80 | recode latin-9/x1..dump	Show what a code represents in latin-9 charmap
• echo -n 0x20AC | recode ucs-2/x2..latin-9/x	Show latin-9 encoding
• echo -n 0x20AC | recode ucs-2/x2..utf-8/x	Show utf-8 encoding

CDs

gzip < /dev/cdrom > cdrom.iso.gz	Save copy of data cdrom
mkisofs -V LABEL -r dir | gzip > cdrom.iso.gz	Create cdrom image from contents of dir
mount -o loop cdrom.iso /mnt/dir	Mount the cdrom image at /mnt/dir (read only)
cdrecord -v dev=/dev/cdrom blank=fast	Clear a CDRW
gzip -dc cdrom.iso.gz | cdrecord -v dev=/dev/cdrom -	Burn cdrom image (use dev=ATAPI -scanbus to confirm dev)
cdparanoia -B	Rip audio tracks from CD to wav files in current dir
cdrecord -v dev=/dev/cdrom -audio *.wav	Make audio CD from all wavs in current dir (see also cdrdao)
oggenc --tracknum='track' track.cdda.wav -o 'track.ogg'	Make ogg file from wav file

disk space (See also FSlint)

• ls -lSr	Show files by size, biggest last
• du -s * | sort -k1,1rn | head	Show top disk users in current dir. See also dutop
• df -h	Show free space on mounted filesystems
• df -i	Show free inodes on mounted filesystems
• fdisk -l	Show disks partitions sizes and types (run as root)
• rpm -q -a --qf '%10{SIZE}\t%{NAME}\n' | sort -k1,1n	List all packages by installed size (Bytes) on rpm distros
• dpkg-query -W -f='${Installed-Size;10}\t${Package}\n' | sort -k1,1n	List all packages by installed size (KBytes) on deb distros
• dd bs=1 seek=2TB if=/dev/null of=ext3.test	Create a large test file (taking no space). See also truncate
• > file	truncate data of file or create an empty file

monitoring/debugging

• tail -f /var/log/messages	Monitor messages in a log file

- strace -c ls >/dev/null — Summarise/profile system calls made by command
- strace -f -e open ls >/dev/null — List system calls made by command
- ltrace -f -e getenv ls >/dev/null — List library calls made by command
- lsof -p $$ — List paths that process id has open
- lsof ~ — List processes that have specified path open
- tcpdump not port 22 — Show network traffic except ssh. See also tcpdump_not_me
- ps -e -o pid,args --forest — List processes in a hierarchy
- ps -e -o pcpu,cpu,nice,state,cputime,args --sort pcpu | sed '/^ 0.0 /d' — List processes by % cpu usage
- ps -e -orss=,args= | sort -b -k1,1n | pr -TW$COLUMNS — List processes by mem (KB) usage. See also ps_mem.py
- ps -C firefox-bin -L -o pid,tid,pcpu,state — List all threads for a particular process
- ps -p 1,2 — List info for particular process IDs
- last reboot — Show system reboot history
- free -m — Show amount of (remaining) RAM (-m displays in MB)
- watch -n.1 'cat /proc/interrupts' — Watch changeable data continuously

system information (see also sysinfo) ('#' means root access is required)

- uname -a — Show kernel version and system architecture
- head -n1 /etc/issue — Show name and version of distribution
- cat /proc/partitions — Show all partitions registered on the system
- grep MemTotal /proc/meminfo — Show RAM total seen by the system
- grep "model name" /proc/cpuinfo — Show CPU(s) info
- lspci -tv — Show PCI info
- lsusb -tv — Show USB info
- mount | column -t — List mounted filesystems on the system (and align output)
- grep -F capacity: /proc/acpi/battery/BAT0/info — Show state of cells in laptop battery

dmidecode -q | less — Display SMBIOS/DMI information

smartctl -A /dev/sda | grep Power_On_Hours — How long has this disk (system) been powered on in total

hdparm -i /dev/sda — Show info about disk sda

hdparm -tT /dev/sda — Do a read speed test on disk sda

badblocks -s /dev/sda — Test for unreadable blocks on disk sda

interactive (see also linux keyboard shortcuts)

- readline — Line editor used by bash, python, bc, gnuplot, ...

• screen	Virtual terminals with detach capability, ...
• mc	Powerful file manager that can browse rpm, tar, ftp, ssh, ...
• gnuplot	Interactive/scriptable graphing
• links	Web browser
• xdg-open .	open a file or url with the registered desktop application

miscellaneous

• alias hd='od -Ax -tx1z -v'	Handy hexdump. (usage e.g.: • hd /proc/self/cmdline | less)
• alias realpath='readlink -f'	Canonicalize path. (usage e.g.: • realpath ~/../$USER)
• set | grep $USER	Search current environment
touch -c -t 0304050607 file	Set file timestamp (YYMMDDhhmm)
• python -m SimpleHTTPServer	Serve current directory tree at http://$HOSTNAME:8000/

Filesystem

Change the mode of all files to 666 (—rw-rw-rw) recursively from the current directory

```
find . -type f -exec chmod 666 {} \;
```

Change the mode of all directories to 777 (—rwxrwxrwx) recursively from the current directory

```
find . -type f -exec chmod 777 {} \;
```

CDROMs, DVDs and ISOs

Make an .ISO image file on-disk from a cdrom

```
dd if=/dev/cdrom of=cdrom.iso
```

Mount an ISO image

```
mount -o loop cdrom.iso /mnt/iso
```

Process Management

Find a process by name

```
ps -ef | grep <process name>
```

Kill a process by name

```
killall <process name>
```

Stop (pause) a process

```
kill -SIGSTOP <pid>
```

Stop (pause) a process and all it's children processes

```
kill -SIGSTOP -<pid>
```

To continue a stopped (paused) process use the SIGCONT signal instead.

Shell (Bash)

Spawn a process as a background job

```
$> <executable> &
```

List all jobs in the shell

```
$> jobs
```

Disown a spawned job (so it isn't killed when the shell exits)

```
$> disown %<job number>
```

Networking

Discovery

Ping broadcast

```
ping -b 192.168.1.255
```

DNS domain transfer

```
host -l <domain>
```

Ports

Find all opened ports and by whom

```
netstat -tulpn
```

Dump the arp entries

```
arp -a
```

Find open ports on a remote computer

```
nmap -v <IP or hostname>
```

Remote Desktops

Start a desktop session that can be accessed remotely

```
vncserver
```

Connect to the remote desktop session from your local computer

```
vncviewer <hostname or ip>:1
```

Initiate a reverse desktop connection (usually used to bypass firewalls)
from the client box

```
vncserver -listen <port>
```

from the system that owns the desktop

```
vncconfig -display :1 -connect <hostname or ip>:<port>
```

Edit ~/.vnc/xstartup to modify which desktop and applications auto start on your desktop

Start a desktop from within a desktop

```
xinit ~/.xinitrc -- `which Xnest` :5
```

where ~/.xinitrc contains:

```
#!/bin/bash
exec /usr/kde/3.5/bin/startkde
```

(or to your preferred window manager)

Applications

Firefox

Set the VM (Gentoo Specific)
in a browser tab type '*about:config*'
Set the value *java.default_java_location_others* to */etc/java-config-2/current-system-vm*

Java

List all VMs

```
java-config -L
```

Set the default VM

```
java-config -S <VM name>
```

VIM

Replace all occurances of string1 to string2 in a file

```
%s/string1/string2/g
```

Oracle Server 10g

General

Start the database

```
dbstart
```

Stop the database

```
dbstop
```

Start the TNS listener

```
tnslsnrctl start
```

Stop the TNS listener

```
tnslsnrctl stop
```

Get the status of the TNS listener

```
tnslsnrctl status
```

SQLPLUS

Login using a fully qualified specification

```
sqlplus <user>/<password>@<host>/<SID>
```

DataPump

Export a schema to a file

```
expdp <user>/<password> schemas=<schema name> directory=my_dumps logfile=dump.log
dumpfile=dumpfile.dmp
```

Import a a dump file

```
impdp <user>/<password> schemas=<schema name> directory=my_dumps logfile=dump_import.log
dumpfile=<dump filename>
```

Import a dumpfile while remapping it to another schema

```
impdp <user>/<password> schemas=<schema name> remap_schema=<schema name>:<new schema name>
directory=my_dumps logfile=dump_import.log dumpfile=<dump filename>
```

Retrieve the number of clients connected to an Oracle server

```
sqlplus> select count(*) from v$sessions;
```

Enterprise Manager

Start/Stop/Status the Enterprise Manager

```
emctl start|stop|start dbconsole
```

Browser URL

```
[http://www.example.com http://<host>:1158/em]
```

Linux Cheat Sheet

We are re-doing this page…

Arrow Up: scrolls and edits the command history, press enter to activate.
Shift+pgup: scrolls terminal output up
Shift+pgdown: scrolls terminal output down
CTRL-ALT+DEL reboots the system
Shutdown -h now turns the system off
CTRL C kills the current process
CTRL S Stops the tranfer to the terminal
CTRL Q Resumes the transfer to the terminal
CTRL Z Puts the current process in the background.
Middle Mouse Button Pastes the text that is currently somewhere else.

PWD Shows the current directory
HOSTNAME Shows the host name of the system you are on
WHOAMI Displays your login name
DATE Displays what your machine thinks the date is
WHO Shows who is logged into the machine
RWHO-A Shows all users logged into the server network
FINGER <user name> Shows info on chosen user
LAST Show the last users logged into the machine
UPTIME Shows the systems uptime
PS Shows the current user processes
PS -A Shows all process on the system
UNAME -A Displays all info on your host.
FREE Shows the free memory in KB
DF -H Shows the disk space details
cat/proc/cpuinfo Shows the CPU information
cat/proc/filesystems Shows the file system information in use
cat/etc/printcap Shows if any printers are hooked up
Lsmod Shows the kernel modules loaded
setimore Shows the current user environment
echo $PATH Shows the content on the environment variable path
dmesg Prints the boot messages

Basic Actions

[command] –help – gives syntax for using that command
man [command] – brings up the manual page for the command, if it exists
man [command] > file.txt – dumps the manual page(s) for the command into ‘file.txt’
whatis [command] – gives a short description of the command.
help – gives a list of commands (GNU Bash).
help [command] – gives extra information on the commands listed above. Viewing/editing/creating a text file

vi [filename] – opens VI text editor, if the file doesn’t exist, it’ll be created on saving.
(when inside vi)
- using ‘i’ inserts
- pressing ‘escape’ and then ‘:’ goes back to command mode.
- ‘/searchstring’ searchs for ’searchstring’ using regular expressions.
- ‘:’ followed by ‘w’ writes
- ‘:’ followed by ‘qw’ writes then quits
- ‘:’ followed by ‘q’ quits.
- ‘:’ followed by ‘q!’ quits regardless of whether changes are made.
- ‘:’ followed by ‘z’ undos.
pico [filename] – launches the PICO editor for the filename.
more [filename] – shows one screen’s worth of the file at a time.
less [filename] – similar to more
head [filename] – Shows the first 10 lines of file, or use -n

tail [filename] – Shows the last 10 lines of file, or use -n
cat [filename] | more – works like more, cat concats 2 strings General/System commands

su [user] – changes the login to 'user', or to the root if no 'user' is given.
date – shows the system date
whoami – tells you who you're logged in as
uptime – how long the computer has been running, plus other details
w – shows who's logged on, what they're doing.
df – how much disk space is left.
du – disk usage by your login, it can also total up directories.
uname -mrs – userful info about the system
uname -a – all details about the system Desktop / X server + client

Switchdesk {manager – gnome, Enlightenment, etc} – Switches your desktop What's running

ps – what's running.
ps ax – shows all processes
top – sort of interactive version of ps.
kill [pid] – terminates the named process, which can be name or number or other options.
killall -HUP [command name] – kill a process, running the command specified, by name.
killall -9 [command] – similar to the above
xkill – kills a frozen application in X (gnome,kde etc. desktops), you just click on the frozen app.

File system

ls -la – list all files/directories
dir – simple form of ls
cd [dir] – change directory
cd ~ – go back to the home directory
cdup – similar to using "cd ..", go up one directory.
pwd – print which directory you're in.
./[filename] – run the file if it's executable and in the current directory
rm [filename] – delete a file
rm -R [directory] – delete a directory
mv [oldfilename] [newfilename] – renames the file (or directory)
cp [filename-source] [filename-destination] – copy the file from one place to another
cp -R [dir-source] [dir-destination] – copy a directory an all its subdirectories
mkdir [name] – makes a directory.
cat [sourcefile] >> [destinationfile] – appends sourcefile to the end of destinationfile
df – how much disk space is available, more options available.

- zipping/taring
tar -cvzf mytar.tar.gz sourcefilesordir – creates a new tar file, verbose options on, runs it through gnuzip,f is the filename
tar -xvf mytar.tar.gz destination – extracts a tar file (this example is compressed with gzip), verbosely, f is the filename
gzip fileordir – compresses a file with gzip.
gunzip file.gz – decompresses a file with gzip.
NB gzip only compresses files, it doesn't collect them into a single file like a tarball does.

Searching

locate [filename] – searches the system using an indexed database of files. use updatedb to update the file database
locate [filename] | sort – sorts the files alphabetically
whereis [filename] – locates an application, such as 'whereis bash'
find [filename] – searches the filesystem as with locate, but without a database so its slower.
find /directory -atime +30 -print – searches for files not used in the past 30 days. Setting up links

ln -s target linkname – creates a symbolic link, like a shortcut to the target directory or filename.
ln target linkname – creates the default hard link. Deleting this will delete the targetted file or directory. Network commands

dig domainname – retrieves information about a domain, such as name servers, mx records
whois domainname – whois info on a domain

finger user – gives info about a user, their group status, but can also be used over a network
netstat -ape – lots of info about whos connected to your machine, what processes are doing what with sockets Piping

Piping to another command is straight forward enough:

locate filename | grep /usr/local > searchresults.txt – searches for filename, runs the results through grep to filter everything without /usr/local in it, and then outputs the results to searchresults.txt

| runs one application via another, and can be used multiple times e.g. cat /usr/group | more | grep root | sort
> creates a new file if once doesn't already exist, overwrites the contents of the file if it does exist
>> appends to the end of the file, and creates the file if one doesn't exist.
< sends everything after this to the application, e.g. ./mysql -u bob -p databasename < mysqldump.sql Permissions and directory listing format

groups [username] – shows what groups the user belongs to
id [username] – shows extended information about a user.
finger [user] – give details about a user.
passwd [user] – changes the password for a user, or without the user argument, changes your password.
chsh [user] – changes the shell for a user.
userdel [user] – removes a user from the system, use -r to remove their home directory too.
newgrp [group id] – log into a new group.
useradd -d /home/groupname -g groupname – add a new user with the d being the homedirectory, g the default group they belong to.
groupadd [groupname] – adds a group

Take a look at the users/groups on the system with:

cat /etc/passwd | sort
cat /etc/group | sort

The stuff below is in the man pages also.
The format of passwd is:
username
password denoted by x (use cat /etc/shadow | sort to list the shadow password file)
uid – user identifier number
gid – group identifier number
misc information such as real name
users home directory
shell for the user

The format of group is:
name of group
password denoted by x (use cat /etc/gshadow | sort to list the shadow group file)
gid – group identifier number
list of additional users assigned to the group

Break down of permissions in a directory listing:
-rw-r–r– 1 mainuser devel 9054 Dec 28 12:42 index.html

The first character indicates whether it is a directory or file (d for directory).
After that, the next 3 (rw-) are owner permissions.
The following 3 (r–) are group permissions
The following 3(r–) are permissions for other users.

After that reads the number of files inside the directory if it's a directory (which it isn't so it's 1) this can also be links to the file, the owner of the file, the group the file belongs to, size in bytes, date and time and then the filename.

Chmod and Chown
Owner,group and other permissions can be r,w,x. Translated into their decimal equivalents (actually octal but…)
owner – read=400,write=200,execute=100
group – read=40,write=20,execute=10
other – read=4,write=2,execute=1

So add them up and you've got your user permissions for chmoding:
chmod [mode] fileordirectory – changes the permissions on a file or directory. use -r to recursively change a whole directory and its sub directories.

e.g chmod 755 myfile.txt – changes the permissions on the file to 755 which is : owner read,write,execute; group read,execute; other read,execute.

chown [user:group] fileordirectory – changes the user and group ownership of a file or directory. Use -R to recursively change a whole directory and its sub directories.
chgrp [group] fileordirectory – changes the groupownership of a file or directory. Use -R to recursively change a whole directory and its sub directories.

MySQL

mysqldump – Dumps a table,database or all databases to a SQL file. Use the –opt argument for best results e.g.
mysqldump -u username -p –opt database > file.sql
mysql – The mySQL query manager. To import/export a database to or from a SQL try:
mysql -u username -p database < file_to_go_in.sql
mysql -u username -p database > file_to_go_to.sql

Basic Commands

`man {command}`	Type **man ls** to read the manual for the **ls** command.
`man {command} > {filename}`	Redirect help to a file to download.
`whatis {command}`	Give short description of command. (Not on RAIN?)
`apropos {keyword}`	Search for all Unix commands that match keyword, eg **apropos file**. (Not on RAIN?)

List a directory

`ls {path}`	It's ok to combine attributes, eg **ls -laF** gets a long listing of all files with types.
`ls {path_1} {path_2}`	List both {path_1} and {path_2}.
`ls -l {path}`	Long listing, with date, size and permisions.
`ls -a {path}`	Show all files, including important .dot files that don't otherwise show.
`ls -F {path}`	Show type of each file. "/" = directory, "*****" = executable.
`ls -R {path}`	Recursive listing, with all subdirs.
`ls {path} > {filename}`	Redirect directory to a file.
`ls {path} | more`	Show listing one screen at a time.
`dir {path}`	Useful alias for DOS people, or use with **ncftp**.

Change to directory

`cd {dirname}`	There must be a space between.
`cd ~`	Go back to home directory, useful if you're lost.
`cd ..`	Go back one directory.

`cdup`	Useful alias, like “cd ..”, or use with **ncftp**.

Make a new directory

`mkdir {dirname}`

Remove a directory

`rmdir {dirname}`	Only works if {dirname} is empty.
`rm -r {dirname}`	Remove all files and subdirs. Careful!

Print working directory

`pwd`	Show where you are as full path. Useful if you’re lost or exploring.

Copy a file or directory

`cp {file1} {file2}`	
`cp -r {dir1} {dir2}`	Recursive, copy directory and all subdirs.
`cat {newfile} >> {oldfile}`	Append newfile to end of oldfile.

Move (or rename) a file

`mv {oldfile} {newfile}`	Moving a file and renaming it are the same thing.
`mv {oldname} {newname}`	

Delete a file

`rm {filespec}`	**?** and ***** wildcards work like DOS should. “?” is any character; “*” is any string of characters.
`ls {filespec}` `rm {filespec}`	Good strategy: first list a group to make sure it’s what’s you think… …then delete it all at once.

Download with zmodem

(Use **sx** with xmodem.)

`sz [-a|b] {filename}`	**-a** = ascii, **-b** = binary. Use binary for everything. (It’s the default?)
`sz *.zip`	Handy after downloading with FTP. Go talk to your spouse while it does it’s stuff.

Upload with zmodem

(Use **rx** with xmodem.)

`rz [-a|b] (filename}`	Give **rz** command in Unix, THEN start upload at home. Works fine with multiple files.

View a text file

`more {filename}`	View file one screen at a time.
`less {filename}`	Like **more**, with extra features.
`cat {filename}`	View file, but it scrolls.
`cat {filename} | more`	View file one screen at a time.
`page {filename}`	Very handy with **ncftp**.
`pico {filename}`	Use text editor and don't save.

Edit a text file.

`pico {filename}`	The same editor PINE uses, so you already know it. **vi** and **emacs** are also available.

Create a text file.

`cat > {filename}`	Enter your text (multiple lines with **enter** are ok) and press **control-d** to save.
`pico {filename}`	Create some text and save it.

Compare two files

`diff {file1} {file2}`	Show the differences.
`sdiff {file1} {file2}`	Show files side by side.

Other text commands

`grep '{pattern}' {file}`	Find regular expression in file.
`sort {file1} > {file2}`	Sort file1 and save as file2.
`sort -o {file} {file}`	Replace file with sorted version.
`spell {file}`	Display misspelled words.
`wc {file}`	Count words in file.

Find files on system

`find {filespec}`	Works with wildcards. Handy for snooping.

`find {filespec} > {filename}`	Redirect find list to file. Can be big!

Make an Alias

`alias {name} '{command}'`	Put the command in 'single quotes'. More useful in your **.cshrc** file.

Wildcards and Shortcuts

`*`	Match any string of characters, eg **page*** gets page1, page10, and page.txt.
`?`	Match any single character, eg **page?** gets page1 and page2, but not page10.
`[…]`	Match any characters in a range, eg **page[1-3]** gets page1, page2, and page3.
`~`	Short for your home directory, eg **cd ~** will take you home, and **rm -r ~** will destroy it.
`.`	The current directory.
`..`	One directory up the tree, eg **ls ..**.

Pipes and Redirection

(You **pipe** a command to another command, and **redirect** it to a file.)

`{command} > {file}`	Redirect output to a file, eg **ls > list.txt** writes directory to file.
`{command} >> {file}`	Append output to an existing file, eg **cat update >> archive** adds update to end of archive.
`{command} < {file}`	Get input from a file, eg **sort < file.txt**
`{command} < {file1} > {file2}`	Get input from `file1`, and write to `file2`, eg **sort < old.txt > new.txt** sorts old.txt and saves as new.txt.
`{command} | {command}`	Pipe one command to another, eg **ls | more** gets directory and sends it to **more** to show it one page at a time.

ac - Acension Island
ad - Andorra
ae - United Arab Emirates
aero - Aerospace related industry TLD
af - Afghanistan
ag - Antigua And Barbuda
ai - Anguilla
al - Albania
am - Armenia
an - Netherlands Antilles
ao - Angola
aq - Antarctica
ar - Argentina
as - American Samoa
at - Austria [Global "@"-like TLD]
au - Australia
aw - Aruba
ax - Aland Islands
az - Azerbaijan
ba - Bosnia And Herzegowina
bb - Barbados
bd - Bangladesh
be - Belgium
bf - Burkina Faso
bg - Bulgaria
bh - Bahrain
bi - Burundi
biz - "Bizness" business TLD
bj - Benin
bm - Bermuda
bn - Brunei Darussalam
bo - Bolivia
br - Brazil
bs - Bahamas
bt - Bhutan
bv - Bouvet Island
bw - Botswana
by - Belarus
bz - Belize [Global "BiZ-like" name TLD]
ca - Canada
cc - Cocos (Keeling) Islands
cd - Congo, The Democratic Republic Of
cf - Central African Republic
cg - Congo
ch - Switzerland
ci - Cote d'Ivoire
ck - Cook Islands
cl - Chile
cm - Cameroon
cn - China {.com.cn, .net.cn, .org.cn}
co - Colombia
com- Commercial TLD
coop - Co-operative organization TLD
cr - Costa Rica
cs - Serbia and Montenegro
cu - Cuba
cv - Cape Verde
cx - Christmas Island
cy - Cyprus
cz - Czech Republic
de - Germany
dj - Djibouti
dk - Denmark
dm - Dominica
do - Dominican Republic
dz - Algeria
ec - Ecuador
edu- Educational TLD
ee - Estonia
eg - Egypt
eh - Western Sahara
er - Eritrea
es - Spain
et - Ethiopia
eu - European Union [not available yet]
fi - Finland
fj - Fiji
fk - Falkland Islands (Malvinas)
fm - Micronesia, Federal State of
fo - Faroe Islands
fr - France
~~**fx** - France, Metropolitan~~ *[removed from ISO-3166]*
ga - Gabon
gb - United Kingdom
gd - Grenada
ge - Georgia
gf - French Guiana
gg - Guernsey
gh - Ghana
gi - Gibraltar
gl - Greenland
gm - Gambia
gn - Guinea
gov- U.S. Government TLD
gp - Guadeloupe
gq - Equatorial Guinea
gr - Greece
gs - South Georgia & South Sandwich Islands
gt - Guatemala
gu - Guam
gw - Guinea-bissau
gy - Guyana
hk - Hong Kong
hm - Heard & McDonald Islands
hn - Honduras
hr - Croatia/Hrvatska
ht - Haiti
hu - Hungary {.co.hu, .info.hu, .priv.hu, ...}
id - Indonesia
ie - Ireland
il - Israel {.co.il, .org.il}
im - Isle of Man
in - India
info - Global information TLD
int - International Organization TLD
io - British Indian Ocean Territory
iq - Iraq
ir - Iran
is - Iceland
it - Italy
je - Jersey
jm - Jamaica
jo - Jordan
jp - Japan
ke - Kenya
kg - Kyrgyzstan
kh - Cambodia
ki - Kiribati
km - Comoros
kn - Saint Kitts And Nevis
kp - Korea, Democratic People's Republic of
kr - Korea, Republic of
kw - Kuwait
ky - Cayman Islands
kz - Kazakhstan
la - Lao People's Democratic Republic [L.A.]
lb - Lebanon
lc - Saint Lucia
li - Liechtenstein
lk - Sri Lanka
lr - Liberia
ls - Lesotho
lt - Lithuania
lu - Luxembourg
lv - Latvia
ly - Libyan Arab Jamahiriya
ma - Morocco
mc - Monaco
md - Moldova, Republic Of [Healthcare TLD]
mg - Madagascar
mh - Marshall Islands
mil- U.S. Military TLD
mk - Macedonia, The Former Yugoslav Republic
ml - Mali
mm - Myanmar
mn - Mongolia
mo - Macau
mp - Northern Mariana Islands
mq - Martinique
mr - Mauritania
ms - Montserrat
mt - Malta
mu - Mauritius
museum - Museum TLD
mv - Maldives
mw - Malawi
mx - Mexico {.com.mx}
my - Malaysia
mz - Mozambique
na - Namibia
name - Global "name" TLD
nc - New Caledonia
ne - Niger
net- Global network services TLD
nf - Norfolk Island
ng - Nigeria
ni - Nicaragua
nl - Netherlands
no - Norway
np - Nepal
nr - Nauru
nu - Niue
nz - New Zealand {.co.nz, .net.nz, .org.nz}
om - Oman
org- Global non-profit organization TLD
pa - Panama
pe - Peru
pf - French Polynesia
pg - Papua New Guinea
ph - Philippines
pk - Pakistan
pl - Poland
pm - St. Pierre And Miquelon
pn - Pitcairn Island
pr - Puerto Rico
pro - "Professional" TLD
ps - Palestinian Territory, Occupied
pt - Portugal
pw - Palau
py - Paraguay
qa - Qatar
re - Reunion Island
ro - Romania
ru - Russian Federation
rw - Rwanda
sa - Saudi Arabia
sb - Solomon Islands
sc - Seychelles
sd - Sudan
se - Sweden
sg - Singapore
sh - Saint Helena
si - Slovenia
sj - Svalbard And Jan Mayen Islands
sk - Slovakia/Slovak Republic
sl - Sierra Leone
sm - San Marino
sn - Senegal
so - Somalia
sr - Suriname
st - Sao Tome And Principe ["SiTe" name]
sv - El Salvador
sy - Syrian Arab Republic
sz - Swaziland
tc - Turks And Caicos Islands
td - Chad
tf - French Southern Territories
tg - Togo
th - Thailand
tj - Tajikistan
tk - Tokelau
tm - Turkmenistan
tn - Tunisia
to - Tonga [For names like go.to, fly.to]
tp - East Timor
tr - Turkey {.gen.tr, .com.tr}
tt - Trinidad And Tobago
tv - Tuvalu [Global "TeleVision" TLD]
tw - Taiwan
tz - Tanzania, United Republic Of
ua - Ukraine
ug - Uganda
uk - United Kingdom {.co.uk, .net.uk, .org.uk,.me.uk)
um - U.S. Minor Outlying Islands
us - United States [Must be U.S. Citizen]
uy - Uruguay
uz - Uzbekistan
va - Holy See (Vatican City State)
vc - Saint Vincent And The Grenadines
ve - Venezuela
vg - Virgin Islands (British)
vi - Virgin Islands (U.S.)
vn - Vietnam
vu - Vanuatu [Was once globally available]
wf - Wallis And Futuna Islands
ws - Samoa [Global "WebSite" name TLD]
ye - Yemen
yt - Mayotte
yu - Yugoslavia
za - South Africa
zm - Zambia
zw - Zimbabwe

Internet Top Level Domain(TLD) suffix chart

suso.org

Underlined extensions are globally and publically available through at least one registrar. Some of these may however have namespace restrictions.

Chart created on November 7, 2000 by Suso Banderas, updated March 28, 2005 by Suso Technology Services, Inc. This work is licensed under the Creative Commons Attribution-ShareAlike License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.
For more information, downloads, ordering laminated copies and the original OpenOffice.org file that created this sheet, please visit http://suso.org/infosheets/
Information is from ISO standard #3166, IANA root TLD list, Network Solutions' extention list, Register.com's domain rules and various other registries lists of purchaseable domains.

SOME RIGHTS RESERVED

FOR GNOME:

General Shortcut Keys

Alt + **F1**	Opens the **Applicantions Menu** .
Alt + **F2**	Displays the **Run Application** dialog.
Print Screen	Takes a screenshot.
Alt + **Print Screen**	Takes a screenshot of the window that has focus.
Ctrl + **Alt** + **right arrow**	Switches to the workspace to the right of the current workspace.
Ctrl + **Alt** + **left arrow**	Switches to the workspace to the left of the current workspace.
Ctrl + **Alt** + **up arrow**	Switches to the workspace above the current workspace.
Ctrl + **Alt** + **down arrow**	Switches to the workspace below the current workspace.
Ctrl + **Alt** + **d**	Minimizes all windows, and gives focus to the desktop.
F1	Starts the online help browser, and displays appropriate online Help.

Window Shortcut Keys

Alt + **Tab**	Switches between windows. When you use these shortcut keys, a list of windows that you can select is displayed. Release the keys to select a window.
Alt + **Esc**	Switches between windows in reverse order. Release the keys to select a window.
F10	Opens the first menu on the left side of the menubar.
Alt + **spacebar**	Opens the **Window Menu** .
Arrow keys	Moves the focus between items in a menu.
Return	Chooses a menu item.
Esc	Closes an open menu.
Ctrl + **Alt** + **right arrow**	Switches to the workspace to the right of the current workspace.
Ctrl + **Alt** + **left arrow**	Switches to the workspace to the left of the current workspace.
Ctrl + **Alt** + **up arrow**	Switches to the workspace above the current workspace.
Ctrl + **Alt** + **down arrow**	Switches to the workspace below the current workspace.
Ctrl + **Alt** + **d**	Minimizes all windows, and gives focus to the desktop.

Panel Shortcut Keys

Ctrl + **Alt** + **Tab**	Switches the focus between the panels and the desktop. When you use these shortcut keys, a list of items that you can select is displayed. Release the keys to select an item.
Ctrl + **Alt** + **Esc**	Switches the focus between the panels and the desktop. Release the keys to select an item.
Ctrl + **F10**	Opens the popup menu for the selected panel.
Tab	Switches the focus between objects on a panel.
Return	Chooses the selected panel object or menu item.
Shift + **F10**	Opens the popup menu for the selected panel object.

Ctrl + **Alt** + **Tab**	Switches the focus between the panels and the desktop. When you use these shortcut keys, a list of items that you can select is displayed. Release the keys to select an item.
Arrow keys	Moves the focus between items in a menu. Moves the focus between interface items in an applet also.
Esc	Closes an open menu.
F10	Opens the **Applications** menu from the *Menu Bar* , if the *Menu Bar* is in a panel.

Application Shortcut Keys

Shortcut Keys	***Command***
Ctrl + **N**	New
Ctrl + **X**	Cut
Ctrl + **C**	Copy
Ctrl + **V**	Paste
Ctrl + **Z**	Undo
Ctrl + **S**	Save
Ctrl + **Q**	Quit

FOR KDE:

Alt-F2	**Execute Command**
Ctrl - B	**Add Bookmark**
Ctrl-Esc	**List of running applications.**
Ctrl - W	**Close**
Alt-Tab	**Switch forward among windows**
Ctrl - C	**Copy**
Alt-Shift-Tab	**Switch backward among windows**
Ctrl - End	**End**
Ctrl - F1 to F12	**Switch to Desktop 1 - 12**
Ctrl - F	**Find**
Shift-Ctrl-F1 to F4	**Switch to Desktop 13 to 16**
F3	**Find Next**
Ctrl-Alt-Esc	**Kill Window (Click on the window to action the kill.)**
Ctrl - F3	**Find Prev**
Ctrl-Alt-Delete	**Logout (To complete logout process, Tab & Enter)**

F1	**Help**
F12	**Toggle cursor key mouse emulation**
Ctrl - Home	**Home**
Alt - F1	**Popup KDE launch menu**
Ctrl - Insert	**Insert**
Ctrl-Tab	**Switch forward one desktop**
Ctrl - N	**New**
Ctrl-Shift-Tab	**Switch back one desktop**
Ctrl - Down	**Next Complete Match**
Alt - F4	**Window close**
Down	**Next Item in List**
Alt - F3	**Window open menu**
Ctrl - O	**Open**
Ctrl - V	**Paste**
"Windows" Menu	**Popup Menu Context**
Ctrl - Up	**Previous Complete Match**
Up	**Previous Item in list**
Ctrl - P	**Print**
Page Up	**Prior**
Ctrl - Q	**Quit**
Ctrl-Shift-Z	**Redo**
F5	**Reload**
Ctrl - R	**Replace**
Ctrl - S	**Save**
Ctrl - A	**Select All**
Ctrl - E	**Text Completion**
Ctrl - Z	**Undo**
Shift - F1	**What's this**
Ctrl - Plus (+)	**Zoom In**

Ctrl - Minus (-)	**Zoom Out**