Moxa UC-771x voor beginners

What is linux?

Linux is an operating system. It consists of a "kernel" and various "userland" applications. A Linux system can be easily stripped down or extended. This makes it suitable for desktop, server and embedded applications. All Linux systems are based on the Linux kernel. The Linux kernel is the base of the operating system and contains memory management, drivers and other basic functionality. The Linux kernel can be extended by loading modules. This way drivers or NAT functionality can be added to the kernel.

How to connect

There are two ways to connect to the Moxa:

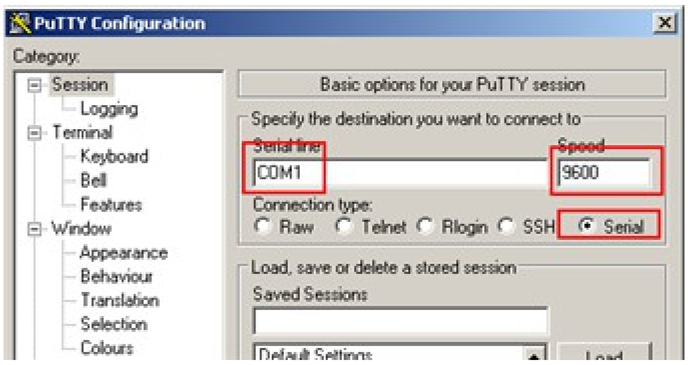
1. Serial:
   - ✔ Shows boot messages
   - ✗ Relatively slow
   - ✗ Direct cable connection
2. Telnet
   - ✔ Easy to connect
   - ✔ Able to use in a remote access session
   - ✔ Fast
   - ✗ Not suitable to solve network/firewall issues

The best program to connect to the moxa is putty (www.putty.nl).

Putty can be installed and used "portable".

Serial

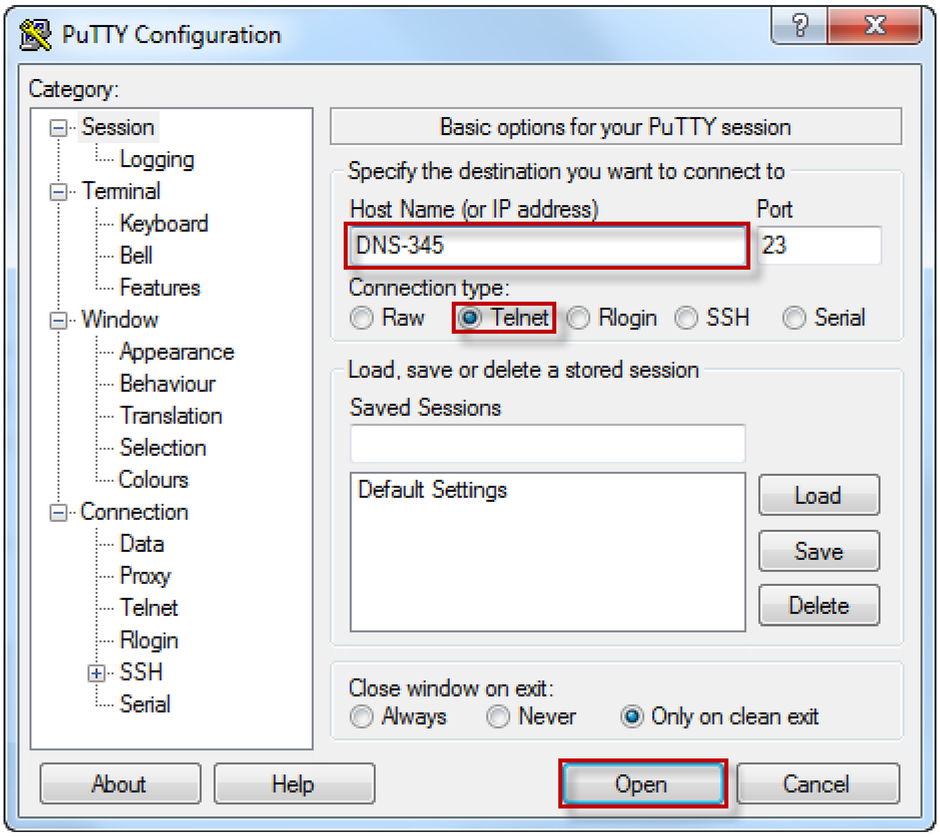
1. Open putty
2. Select on the right side of the radio box "serial"
3. Enter the correct serial port and speed (19200)
4. Click "Open"



Serial line: COM1
Speed: 9600
Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Netwerk

1. Open putty
2. Select the radio box "telnet"
3. Fill in "Host Name" (the IP address of the moxa)
4. Click "Open"


PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): DNS-345

Port: 23

Connection type: Raw [✓ Telnet] Rlogin SSH Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Load / Save / Delete

Close window on exit: Always Never [✓ Only on clean exit]

About / Help / Open / Cancel

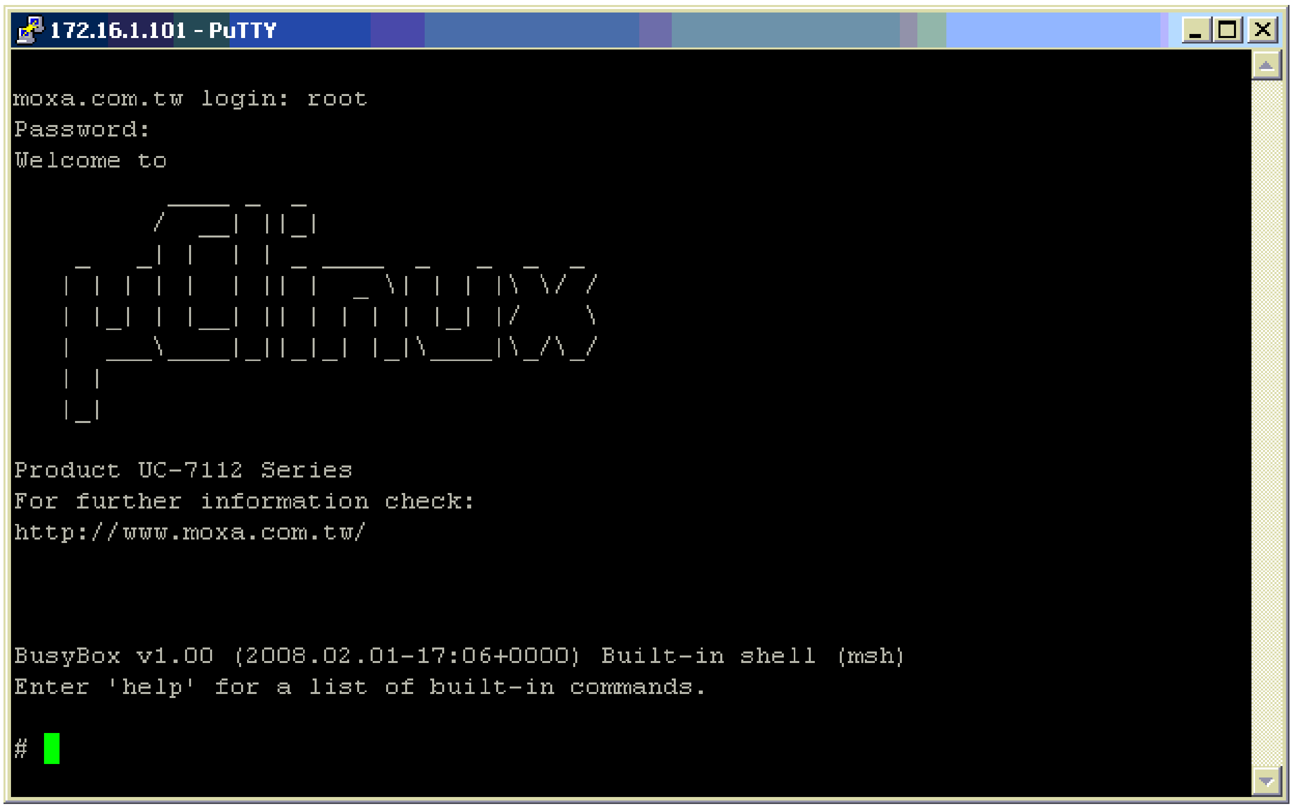
The commandline

After the making of a connection a so-called “login prompt” is displayed.

Default login credentials:

login: root

Password: root



```
moxa.com.tw login: root
Password:
Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

Product UC-7112 Series
For further information check:
http://www.moxa.com.tw/



BusyBox v1.00 (2008.02.01-17:06+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

#
```

Basic commands for navigation and simple file operations are:

ls	List a file or the content of a directory
chmod	Change flags and permissions
cp	Copy files or directories
mv	Move (or rename) files or directories
rm	Remove files or directories
cat	Concatenate files and print their contents
head/tail	output the first/last part of files
cut	remove sections from each line of files
grep	print lines matching a pattern
ps	report a snapshot of the current processes.
kill	terminate a process
vi	Text editor
ifconfig	configure a network interface
iptables	administration tool for IPv4 packet filtering and NAT

General usage tips

Use the TAB-key:

By pressing <TAB> the shell completes your words/commands/file names.

Pressing <TAB> twice show all possible completion options.

Example:

```
#ipt<TAB>
#iptables
#iptables<TAB><TAB>
iptables iptables-restore iptables-save
#iptables
```

Example:

```
#cd /e<TAB>
#cd /etc/<TAB><TAB>
/etc/cron.d/      /etc/dhcpc/
#cd /etc/d<TAB>
#cd /etc/dhcpc/
```

Tab-completion “understands” that the command “***cd***” requires a file/directory name as argument and uses them for completion.

When completing commands, pressing the <TAB> key only shows executable files. (Programs or files with the executable flag set).

Wildcards:

Use the “*****” chracter to indicate multiple files with simular names:

```
# ls net/ipv4/netfilter/ip_*.ko
net/ipv4/netfilter/ip_tables.ko
net/ipv4/netfilter/ip_nat_tftp.ko
net/ipv4/netfilter/ip_nat_snmp_basic.ko
net/ipv4/netfilter/ip_nat_irc.ko
net/ipv4/netfilter/ip_nat_ftp.ko
net/ipv4/netfilter/ip_nat_amanda.ko
net/ipv4/netfilter/ip_nat.ko
#
```

Options:

Commands usually have different options. Options are preceded with a “-” character:

```
# ls /
ramdisk  proc     usr      tmp      lib      dev
sbin     home     var      mnt      etc      bin
# ls -l /
lrwxrwxrwx    1 root     root           12 Feb  1  2008 ramdisk -> /mnt/ramdisk
lrwxrwxrwx    1 root     root            4 Feb  1  2008 sbin -> /bin
dr-xr-xr-x   32 root     root            0 Jan  1  1970 proc
drwxr-xr-x    3 root     root            0 Feb  1  2008 home
drwxr-xr-x    3 root     root            0 Feb  1  2008 usr
drwxr-xr-x    8 root     root         1024 Jan  1  1970 var
lrwxrwxrwx    1 root     root            8 Feb  1  2008 tmp -> /var/tmp
drwxr-xr-x    4 root     root            0 Feb  1  2008 mnt
drwxr-xr-x    3 root     root            0 Feb  1  2008 lib
drwxr-xr-x    4 root     root            0 Jun 12 14:00 etc
drwxr-xr-x    3 root     root            0 Feb  1  2008 dev
drwxr-xr-x    2 root     root            0 Feb  1  2008 bin
```

HELP!:

The option "***--help***" usually shows a short description of the different options and usage of a command.

Pipe:

The "**|**" character (shift+\) pipes the output of one command to another command. This can be used to filter the output of a command:

```
# ps |grep ps
  203 root            RW  ps
  204 root            RW  grep ps
# ps |grep ps|cut -d" " -f 3,4
  203 root
  204 root
```

Redirect output:

It is possible to redirect output of a command to a file. This is done using the ">" character. A single ">" overwrites any existing file whereas a double ">>" appends to it.

```
# ifconfig > ip-configuratie.txt
# cat ip-configuratie.txt
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
# echo test > ip-configuratie.txt
# cat ip-configuratie.txt
test
# ifconfig >> ip-configuratie.txt
# cat ip-configuratie.txt
test
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

Permissions

Long listings show a range of letters at the beginning of each line:

```
drwxr-xr-x    3 root     root            0 Feb  1  2008 dev
```

The first letter indicates whether it concerns a file (-) or directory (d). The following 9 letters indicate the file/directory's permissions. These 9 letters are divided into three groups: User, Group and Others. The letters within these groups represent permission flags and can be either on of off. The order of these letter is always the same.

rwx: Read, Write, eXecute

Since these permission letters can be either on or off, they can be seen as a binary code. Every group of three letters can be translated to one number betweeen 0 and 7. This is done by adding the bit-value of every permission that is set:

r	w	x
4	2	1

The numeric representations of the three permission groups can then be concenated into a three digit numerical representation. For example:

Letters	Numerical
rwxr-xr-x	755
rwxr-x---	750
rw-r--r--	644

Uitleg commandos

ls

ls [-1AacCdeFilnpsTtuwxk] [FILE]...

List directory contents

Options:

```
        -1      List in a single column
        -A      Don't list . and ..
        -a      Don't hide entries starting with .
        -C      List by columns
        -c      With -l: sort by ctime
        -d      List directory entries instead of contents
        -e      List full date and time
        -F      Append indicator (one of */=@|) to entries
        -i      List inode numbers
        -l      Long listing format
        -n      List numeric UIDs and GIDs instead of names
        -p      Append indicator (one of /=@|) to entries
        -s      List the size of each file, in blocks
        -T NUM  Assume tabstop every NUM columns
        -t      With -l: sort by modification time
        -u      With -l: sort by access time
        -w NUM  Assume the terminal is NUM columns wide
        -x      List by lines
```

chmod

chmod [-Rcvf] MODE[,MODE]... FILE...

Each MODE is one or more of the letters ugoa, one of the symbols +-= and one or more of the letters rwxst

Options:

```
        -R      Recurse
```

cp

cp [OPTIONS] SOURCE DEST

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY

Options:

```
        -a      Same as -dpR
        -d,-P   Preserve links
        -H,-L   Dereference all symlinks (default)
        -p      Preserve file attributes if possible
        -f      Force overwrite
        -i      Prompt before overwrite
        -R,-r   Recurse
        -l,-s   Create (sym)links
```

mv

mv [OPTIONS] SOURCE DEST or: mv [OPTIONS] SOURCE... DIRECTORY

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY

Options:

```
        -f      Don't prompt before overwriting
        -i      Interactive, prompt before overwrite
```

rm

rm [OPTIONS] FILE...

Remove (unlink) the FILE(s). Use '--' to indicate that all following arguments are non-options.

Options:

```
        -i      Always prompt before removing
        -f      Never prompt
        -r,-R   Remove directories recursively
```

cat

cat [-u] [FILE]...

Concatenate FILE(s) and print them to stdout

Options:

```
        -u      Use unbuffered i/o (ignored)
```

tail

tail [OPTIONS] [FILE]...

Print last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

Options:

```
        -c N[kbm]       Output the last N bytes
        -n N[kbm]       Print last N lines instead of last 10
        -f              Output data as the file grows
        -q              Never output headers giving file names
        -s SEC          Wait SEC seconds between reads with -f
        -v              Always output headers giving file names
```

If the first character of N (bytes or lines) is a '+', output begins with the Nth item from the start of each file, otherwise, print the last N items in the file. N bytes may be suffixed by k (x1024), b (x512), or m (1024^2).

head

head [OPTIONS] [FILE]...

Print first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.

Options:

```
        -n NUM  Print first NUM lines instead of first 10
        -c NUM  Output the first NUM bytes
        -q      Never output headers giving file names
        -v      Always output headers giving file names
```

cut

cut [OPTIONS] [FILE]...

Print selected fields from each input FILE to standard output

Options:

```
        -b LIST Output only bytes from LIST
        -c LIST Output only characters from LIST
        -d CHAR Use CHAR instead of tab as the field delimiter
        -s      Output only the lines containing delimiter
        -f N    Print only these fields
        -n      Ignored
```

grep

grep [-HhrilLnqvsoweFEABCz] PATTERN [FILE]...

Search for PATTERN in each FILE or standard input

Options:

```
        -H      Prefix output lines with filename where match was found
        -h      Suppress the prefixing filename on output
        -r      Recurse
        -i      Ignore case distinctions
        -l      List names of files that match
        -L      List names of files that do not match
        -n      Print line number with output lines
        -q      Quiet. Return 0 if PATTERN is found, 1 otherwise
        -v      Select non-matching lines
        -s      Suppress file open/read error messages
        -c      Only print count of matching lines
        -o      Show only the part of a line that matches PATTERN
        -m MAX  Match up to MAX times per file
        -w      Match whole words only
        -F      PATTERN is a set of newline-separated strings
        -E      PATTERN is an extended regular expression
        -e PTRN Pattern to match
        -f FILE Read pattern from file
        -A      Print NUM lines of trailing context
        -B      Print NUM lines of leading context
        -C      Print NUM lines of output context
        -z      Input is NUL terminated
```

ps

ps

Report process status

Options:

```
        -o col1,col2=header     Select columns for display
        -T                      Show threads
```

kill

kill [-l] [-SIG] PID...

Send a signal (default: TERM) to given PIDs (-9 forces shutdown)

Options:

```
        -l      List all signal names and numbers
```

ifconfig

ifconfig [-a] interface [address]

Configure a network interface

Options:

```
        [-]broadcast [ADDRESS]] [[-]pointopoint [ADDRESS]]
        [netmask ADDRESS] [dstaddr ADDRESS]
        [outfill NN] [keepalive NN]
        [hw ether|infiniband ADDRESS] [metric NN] [mtu NN]
        [[-]trailers] [[-]arp] [[-]allmulti]
        [multicast] [[-]promisc] [txqueuelen NN] [[-]dynamic]
        [mem_start NN] [io_addr NN] [irq NN]
        [up|down] ...
```

iptables

```
Usage: iptables -[AD] chain rule-specification [options]
       iptables -[RI] chain rulenum rule-specification [options]
       iptables -D chain rulenum [options]
       iptables -[LFZ] [chain] [options]
       iptables -[NX] chain
       iptables -E old-chain-name new-chain-name
       iptables -P chain target [options]
       iptables -h (print this help information)

Commands:
Either long or short options are allowed.
  --append  -A chain            Append to chain
  --delete  -D chain            Delete matching rule from chain
  --delete  -D chain rulenum
                                Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]
                                Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum
                              Replace rule rulenum (1 = first) in chain
  --list    -L [chain]        List the rules in a chain or all chains
  --flush   -F [chain]        Delete all rules in  chain or all chains
  --zero    -Z [chain]        Zero counters in chain or all chains
  --new     -N chain          Create a new user-defined chain
  --delete-chain
            -X [chain]        Delete a user-defined chain
  --policy  -P chain target
                              Change policy on chain to target
  --rename-chain
            -E old-chain new-chain
                              Change chain name, (moving any references)
Options:
  --proto       -p [!] proto    protocol: by number or name, eg. `tcp'
  --source      -s [!] address[/mask]
                                source specification
  --destination -d [!] address[/mask]
                                destination specification
  --in-interface -i [!] input name[+]
                                network interface name ([+] for wildcard)
  --jump        -j target
                                target for rule (may load target extension)
  --goto      -g chain
                              jump to chain with no return
  --match       -m match
                                extended match (may load extension)
  --numeric     -n              numeric output of addresses and ports
  --out-interface -o [!] output name[+]
                                network interface name ([+] for wildcard)
  --table       -t table        table to manipulate (default: `filter')
  --verbose     -v              verbose mode
  --line-numbers                print line numbers when listing
  --exact       -x              expand numbers (display exact values)
[!] --fragment  -f              match second or further fragments only
  --modprobe=<command>          try to insert modules using this command
  --set-counters PKTS BYTES     set the counter during insert/append
[!] --version   -V              print package version.
```

vi

vi [OPTIONS] [FILE]...

Edit FILE

Options:

```
        -R      Read-only - do not write to the file
```

Commonly used commands

Dhcpcd

Route

dnsmasq

ifconfig – configure a network interface

ifconfig is used to display or modify (static) network interface settings. It can be used to change the interface state to on (**up**) or off (**down**), change the IP-address or networkmask of an interface or add virtual interfaces (multiple IP-addresses on one NIC).

NOTE: ifconfig cannot be used to configure a NIC with DHCP. ifconfig will however overwrite any IP-address settings done using DHCP.

Without parameters ifconfig will list the current IP configuration:

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:115641 (112.9 KiB)  TX bytes:167366 (163.4 KiB)
          Interrupt:25
```

The following command sets the IP-address and networkmask of interface “**eth1**”:

```
# ifconfig eth1 192.168.1.1 netmask 255.255.0.0
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:115641 (112.9 KiB)  TX bytes:167366 (163.4 KiB)
          Interrupt:25

eth1      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9F
          inet addr:192.168.1.1  Bcast:192.168.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1412 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:163038 (159.2 KiB)
          Interrupt:27
```

The following command adds a virtual interface with it's own IP-address:

```
# ifconfig eth1:0 192.168.1.2 netmask 255.255.0.0
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:115641 (112.9 KiB)  TX bytes:167366 (163.4 KiB)
          Interrupt:25

eth1      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9F
          inet addr:192.168.1.1  Bcast:192.168.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1412 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:163038 (159.2 KiB)
          Interrupt:27

eth1:0    Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9F
          inet addr:192.168.1.2  Bcast:192.168.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:27
```

The example below turns the interface “**eth1**” off:

```
# ifconfig eth1 down
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:E8:15:8A:9E
          inet addr:172.16.1.101  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:486 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1477 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:115641 (112.9 KiB)  TX bytes:167366 (163.4 KiB)
          Interrupt:25
```

NOTE: when an interface is turned off, all virtual interfaces associated with it will be turned off as well.

iptables – firewall, port-forwarding and NAT-router

iptables the control interface for the “netfilter” modules which can be loaded into the kernel. These modules provide extensive options to filter and redirect network traffic. Together with iptables it's possible to create a firewall, do port-forwarding and configure advanced NAT rules.

NOTE: loading more modules into the linux kernel increases it's memory and CPU usage. In embedded environments like the Moxa it's recommended only to load required modules. So if no firewall or NAT functionality is required, don't load the netfilter kernel modules.

iptables is based on the concept of tables and chains. A chain is a list of rules which are evaluated until a match is found after which the system moves on to the next chain. A table is a collection of chains for a specific mode of operation (NAT or filter). The chains are executed (their rules evaluated) in a specific order. The way the system executes the different chains is illustrated in illustration 1. If a kernel module isn't loaded (like mangle or nat) the chains contained within these tables are skipped.

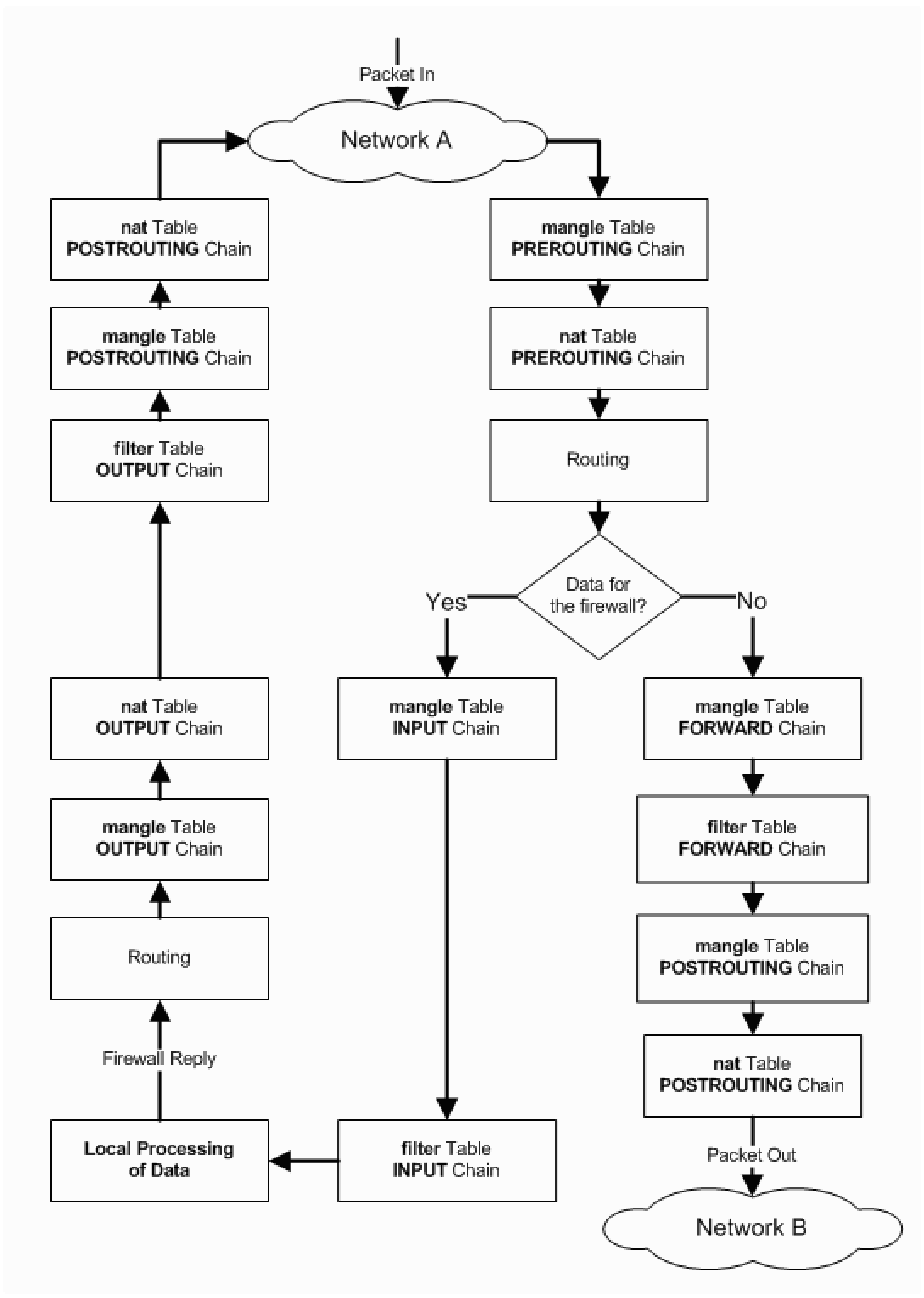

Packet In
Network A
mangle Table PREROUTING Chain
nat Table PREROUTING Chain
Routing
Data for the firewall?
Yes
No
mangle Table INPUT Chain
filter Table INPUT Chain
Local Processing of Data
Firewall Reply
Routing
mangle Table OUTPUT Chain
nat Table OUTPUT Chain
filter Table OUTPUT Chain
mangle Table POSTROUTING Chain
nat Table POSTROUTING Chain
mangle Table FORWARD Chain
filter Table FORWARD Chain
mangle Table POSTROUTING Chain
nat Table POSTROUTING Chain
Packet Out
Network B

Illustration 1: Execution of netfilter tables and chains

NOTE: ip forwarding (or routing) is disabled by default. To enable ip-forwarding the following command should be executed:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Table	Chain	Function
Filter	`FORWARD`	Filters packets to servers accessible by another NIC on the firewall.
	`INPUT`	Filters packets destined to the firewall.
	`OUTPUT`	Filters packets originating from the firewall
Nat	`PREROUTING`	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as destination NAT or DNAT
	`POSTROUTING`	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as source NAT, or SNAT

Table 1: netfilter chains

A chain contains rules and a default target. Every rule consists of match criteria and a target. Once the criteria of a single rule are matched, the system “jumps” to the target for further processing. If all the rules are evaluated without mathing any criteria, the system jumps to the default target. Table 2 shows the most common options to specify match criteria.

iptables command Switch	Description
`-t <-table->`	If you don't specify a table, then the `filter` table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
`-j <target>`	Jump to the specified target chain when the packet matches the current rule.
`-A`	Append rule to end of a chain
`-F`	Flush. Deletes all the rules in the selected table
`-p <protocol-type>`	Match protocol. Types include, icmp, tcp, udp, and all
`-s <ip-address>`	Match source IP address
`-d <ip-address>`	Match destination IP address
`-i <interface-name>`	Match "input" interface on which the packet enters.
`-o <interface-name>`	Match "output" interface on which the packet exits
`--dport <portnr>`	Match destination port

Table 2: Most commonly used match criteria options

target	Desciption	Common Options
`ACCEPT`	• iptables stops further processing.	N/A

	• The packet is handed over to the end application or the operating system for processing	
`DROP`	• iptables stops further processing. • The packet is blocked	N/A
`REJECT`	• Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked	`--reject-with qualifier` The qualifier tells what type of reject message is returned.
`NAT`	• Used to do **destination network address translation** i.e. rewriting the destination IP address of the packet	`--to-destination ipaddress` Tells iptables what the destination IP address should be
`SNAT`	• Used to do **source network address translation** i.e. rewriting the source IP address of the packet • The source IP address is user defined	`--to-source <address>[-<address>][:<port>-<port>]` Specifies the source IP address and ports to be used by SNAT.
`MASQUERADE`	• Used to do Source Network Address Translation. • By default the source IP address is the same as that used by the firewall's interface	`[--to-ports <port>[-<port>]]` Specifies the range of source ports to which the original source port can be mapped.

Table 3: iptable targets

Examples: Saving and loading firewall rules

```
# iptables-save > /etc/firewall
```

Save the current firewall rules to the file **/etc/firewall**. The file **/etc/firewall** will be overwritten.

```
# iptables-restore --no-flush < /etc/ipod-interface
```

Append the contents of the file **/etc/ipod-interface** to any existing firewall rules.

NOTE: The **iptables-restore** command **replaces all** firewall rules by default. Use the option **--no-flush** to append.

Examples: Port-forwarding and MANY:1 NAT

For this example a Moxa has two configured interfaces: an internal interface (**eth0**) with ip-address 192.168.1.40, and an external interface (**eth1**) with ip-address 10.0.0.40. The Moxa will be configured as internet gateway (MANY:1 NAT) and port-forwarding will be configured on the external interface to be able to use the iPhone app.

```
# iptables -t nat -A PREROUTING  -i eth1 -p tcp -d 10.0.0.40 --dport 28401 -j
DNAT --to 192.168.1.50
# iptables -P INPUT DROP
# iptables -A INPUT -i eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j
ACCEPT
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Step by step:

```
# iptables -t nat -A PREROUTING  -i eth1 -p tcp -d 10.0.0.40 --dport 28401 -j
DNAT --to 192.168.1.50
```

Add a rule to the chain **PREROUTING** of the **nat** table which will apply to packets arriving on interface **eth1** using **tcp** protocol and port **28401** and apply **DNAT** to **192.168.1.50**.

```
# iptables -P INPUT DROP
```

Set the default target for chain **INPUT** of the **filter** table (implicitly specified by omitting the **-t** option) in op **DROP**.

```
# iptables -A INPUT   -i eth1        -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

Add a rule to chain **INPUT** of the **filter** tablle which applies to packets that arrived on **eth1** and which has a TCP-port that has the **status ESTABLISHED** or **RELATED** and **ACCEPT** them. This rule enables packets which are sent to the Moxa using a connection that has already been established to pass through to the Moxa itself.

```
# iptables -A FORWARD    -i eth0 -o eth1 -j ACCEPT
```

Add a rule to the chain **FORWARD** of the **filter** table which applies to packets that arrived on **eth0** and will be sent on **eth1**. This rule will allow ip-forwarding for packets originating from the **internal** network with an IP-address on the **external** network as destination. This rule is required to be able to use the Moxa as an internet gateway.

NOTE: This rule doesn't allow any response from the external network to go back through to the internal network!

```
# iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

Add a rule to the chain **FORWARD** of the **filter** table which aplies to packets that arrived on **eth1** and will be sent on **eth0** and have a TCP-port that has the **status ESTABLISHED** or **RELATED** and **ACCEPT** them.

```
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Add a rule to the chain **POSTROUTING** of the **nat** table which applies to packets that will be sent on **eth1** and apply **MASQUERADE**. This will rewrite the “source” address in the packets to the ip-address of the Moxa. Every connection is tracked so that the original source address is retained. When packets return to the moxa they are routed back to the original soruce address. This is often referred to as MANY:1 NAT.

Examples: 1:1NAT internal to external TCP

For this example a Moxa has two internal IP addresses (eth0) 192.168.1.1 and 192.168.1.2 and two external IP addresses(eth1) 10.0.0.1 en 10.0.0.2. There are nodes on the internal network that need to be able to communicate with a node on the external network with IP address 10.0.0.3 using TCP port 502 (Modbus/TCP).

```
# iptables -t nat -A PREROUTING -p tcp -d 192.168.1.2 --dport 502 -j DNAT --to
10.0.0.3
# iptables -A FORWARD -p tcp --dport 502 -d 10.0.0.3 -j ACCEPT
# iptables -t nat -A POSTROUTING -d 10.0.0.3 -j SNAT --to 10.0.0.2
# iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

Step by step:

```
# iptables -t nat -A PREROUTING -p tcp -d 192.168.1.2 --dport 502 -j DNAT --to 
10.0.0.3
```

Add a rule to the chain **PREROUTING** of the **nat** table that applies to packets with destination **192.168.1.2** and destination port **502** and redirects them using **DNAT** to **10.0.0.3**.

```
# iptables -A FORWARD -p tcp --dport 502 -d 10.0.0.3 -j ACCEPT
```

Add a rule to the chain **FORWARD** of the **filter** table which applies to packets with destination **10.0.0.3** and destination port **502** and **ACCEPT**s them.

```
# iptables -t nat -A POSTROUTING -d 10.0.0.3 -j SNAT --to 10.0.0.2
```

Add a rule to the chain **POSTROUTING** of the **nat** table which applies to packets with destination **10.0.0.3** and change the source to **10.0.0.2** using **SNAT**.

```
# iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j 
ACCEPT
```

Add a rule to the chain **FORWARD** of the **filter** table which applies to packets that arrived on **eth0** and will be sent on **eth0** of which the TCP port has status **ESTABLISHED** or **RELATED** and **ACCEPT** them.

Examples: 1:1 NAT

In dit voorbeeld heeft de moxa twee interne IP-adressen (eth0) 192.168.1.1 en 192.168.1.2 en twee externe ip-adressen(eth1) 10.0.0.1 en 10.0.0.2. Verder is er een apparaat op het interne netwerk (192.168.1.3) dat moet kunnen communiceren met een apparaat op het externe netwerk (10.0.0.3) en vice-versa.

```
# iptables -t nat -A PREROUTING  -i eth0 -d 192.168.1.2 -j DNAT --to 10.0.0.3
# iptables -A FORWARD -s 192.168.1.3 -d 10.0.0.3 -j ACCEPT
# iptables -t nat -A POSTROUTING -o eth0 -s 10.0.0.3    -j SNAT --to 192.168.1.2
# iptables -t nat -A PREROUTING  -i eth1 -d 10.0.0.2    -j DNAT --to 192.168.1.3
# iptables -A FORWARD -s 10.0.0.3 -d 192.168.1.3 -j ACCEPT
# iptables -t nat -A POSTROUTING -o eth1 -s 192.168.1.3 -j SNAT --to 10.0.0.2
```

Stap voor stap:

```
# iptables -t nat -A PREROUTING  -i eth0 -d 192.168.1.2 -j DNAT --to 10.0.0.3
```

Voeg een regel toe aan de chain **PREROUTING** van tabel **nat** welke alle pakketten binnengekomen op **eth0** met bestemming **192.168.1.2** doorstuurt naar **10.0.0.3**

```
# iptables -A FORWARD -s 192.168.1.3 -d 10.0.0.3 -j ACCEPT
```

Voeg een regel toe aan de chain **FORWARD** van de tabel **filter** welke alle pakketten die de bron **192.168.1.3** en de bestemming **10.0.0.3** hebben **accepteerd**.

```
# iptables -t nat -A POSTROUTING -o eth0 -s 10.0.0.3    -j SNAT --to 192.168.1.2
```

Voeg een regel toe aan de chain **POSTROUTING** van de tabel **nat** welke van pakketten die verzonden worden op **eth0** (interne netwerk) en bron **10.0.0.3** hebben, de bron veranderd naar **192.168.1.2**.

```
# iptables -t nat -A PREROUTING  -i eth1 -d 10.0.0.2    -j DNAT --to 192.168.1.3
```

Voeg een regel toe aan de chain **PREROUTING** van tabel **nat** welke alle pakketten binnengekomen op **eth1** met bestemming **10.0.0.2** doorstuurt naar **192.168.1.3**.

```
# iptables -A FORWARD -s 10.0.0.3 -d 192.168.1.3 -j ACCEPT
```

Voeg een regel toe aan de chain **FORWARD** van de tabel **filter** welke alle pakketten die de bron **10.0.0.3** en de bestemming **192.168.1.3** hebben **accepteerd**.

```
# iptables -t nat -A POSTROUTING -o eth1 -s 192.168.1.3 -j SNAT --to 10.0.0.2
```

Voeg een regel toe aan de chain **POSTROUTING** van de tabel **nat** welke van pakketten die verzonden worden op **eth1** (externe netwerk) en bron **192.168.1.3** hebben, de bron veranderd naar **10.0.0.2**.

vi voor beginners

vi is een tekst editor en bestaat al sinds 1976. Het is een (naar huidige maatstaven) eenvoudig programma met nagenoeg geen grafische interface. Toch gebruikt 36% van de linux gebruikers dit programma(of zusje vim) nog altijd!

vi werkt radicaal anders dan de grafische applicaties die tegenwoordig gewoongoed zijn. Dit kan zeker voor nieuwe gebruikers erg frustrerend zijn: sterker nog, die krijgen nog geen letter veranderd in een tekstbestand!

Modi

vi kent verschillende modi. De belangrijkste hiervan zijn command modus en insert modus. In command modus kunnen commandos uitgevoerd worden zoals kopieren, plakken, verwijderen, bestand wegschrijven etc. Al deze commandos zijn uit te voeren door toetscombinaties in te drukken. In insert modus kan tekst ingevoerd worden. Om van command modus naar insert modus te gaan drukt men op <i> of <insert>. Om vanuit insert modus naar command modus terug te keren drukt men <esc>.

Voorbeeld: regel verwijderen/knippen, plakken en opslaan:

```
# vi /etc/rc.custom
#!/bin/sh
# Config
MASK_ETH1=255.255.255.0
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
"/etc/rc.custom" line 1 of 75 --1%--
```

De groene rechthoek is de cursor. Door nu op <dd> te drukken (twee keer <d>) wordt de huidige regel geknipt:

```
# Config
MASK_ETH1=255.255.255.0
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
"/etc/rc.custom" [modified] line 1 of 74 --1%--
```

Door nu op <p> te drukken wordt de zojuist geknipte regel onder de huidige regel geplakt:

```
# Config
#!/bin/sh
MASK_ETH1=255.255.255.0
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
"/etc/rc.custom" [modified] line 2 of 74 --2%--
```

Door nu op <:w><enter> te drukken kan het bestand worden weggeschreven:

```
# Config
#!/bin/sh
MASK_ETH1=255.255.255.0
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
:w<enter>
```

```
# Config
#!/bin/sh
MASK_ETH1=255.255.255.0
```

```
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
"/etc/rc.custom" line 2 of 75 --2%--
```

Voorbeeld text toevoegen:

Door op <i> te drukken gaat vi naar “Insert modus”. In insert modus kan men tekst invoeren. vi werkt dan zoals de meeste mensen een text editor gewend zijn. Echter werkt de <del> toets niet zoals verwacht. Onderstaande is het resultaat van <i> en vervolgens: test<del>:

```
tesT#!/bin/sh
# Config
MASK_ETH1=255.255.255.0
IP_ETH1=192.168.2.125
IP_ETH1_prefix=192.168.2.
IP_ETH0_prefix=192.168.1.
"/etc/rc.custom" [modified] line 1 of 75 --1%--
```

Om text te verwijderen in insert modus dient <backspace> gebruikt te worden. Om deze fout ongedaan te maken keert men terug naar command modus door op <esc> te drukken en vervolgens <U> (shift+u) te drukken.

Scripting

Een shell-script is een tekstbestand waar commando's in staan. Als van dit bestand de executable bit aan staat kunnen de commando's in dit bestand uitgevoerd worden. De commando's die in het script staan werken hetzelfde als op de commandline zelf. Shell-scripts zijn dus een eenvoudige manier om dingen te automatiseren die je eerst handmatig hebt uitgeprobeerd. Een goed voorbeeld van zo'n script is /etc/rc.custom.

Hij doet het niet!

“Handige” links en bronnen

http://www.busybox.net/downloads/BusyBox.html (Let op! Nieuwere versie dan op de moxa staat)

http://en.wikibooks.org/wiki/Learning_the_vi_Editor

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables

Vi Cheat Sheet

Modes

Vi has two modes insertion mode and command mode. The editor begins in command mode, where the cursor movement and text deletion and pasting occur. Insertion mode begins upon entering an insertion or change command. [ESC] returns the editor to command mode (where you can quit, for example by typing :q!). Most commands execute as soon as you type them except for "colon" commands which execute when you press the ruturn key.

Quitting

:x	Exit, saving changes
:q	Exit as long as there have been no changes
ZZ	Exit and save changes if any have been made
:q!	Exit and ignore any changes

Inserting Text

i	Insert before cursor
I	Insert before line
a	Append after cursor
A	Append after line
o	Open a new line after current line
O	Open a new line before current line
r	Replace one character
R	Replace many characters

Motion

h	Move left
j	Move down
k	Move up
l	Move right
w	Move to next word
W	Move to next blank delimited word
b	Move to the beginning of the word
B	Move to the beginning of blank delimted word
e	Move to the end of the word
E	Move to the end of Blank delimited word
(	Move a sentence back
)	Move a sentence forward
{	Move a paragraph back
}	Move a paragraph forward
0	Move to the begining of the line
$	Move to the end of the line
1G	Move to the first line of the file
G	Move to the last line of the file
nG	Move to nth line of the file
:n	Move to nth line of the file

Deleting Text

Almost all deletion commands are performed by typing d followed by a motion. For example, dw deletes a word. A few other deletes are:

x	Delete character to the right of cursor
X	Delete character to the left of cursor
D	Delete to the end of the line
dd	Delete current line
:d	Delete current line

Yanking Text

Like deletion, almost all yank commands are performed by typing y followed by a motion. For example, y$ yanks to the end of the line. Two other yank commands are:

yy	Yank the current line
:y	Yank the current line

Changing text

The change command is a deletion command that leaves the editor in insert mode. It is performed by typing c followed by a motion. For wxample cw changes a word. A few other change commands are:

C	Change to the end of the line
cc	Change the whole line

Putting text

p	Put after the position or after the line
P	Put before the poition or before the line

Search for strings

/*string*	Search forward for *string*
?*string*	Search back for *string*
n	Search for next instance of *string*
N	Search for previous instance of *string*