# Linux OS

Tuur Vanhoutte

March 27, 2021

# Contents

# 1 Introductie

## 1.1 Verschil Server & Workstation

### 1.1.1 Server

- Deliver services to (multiple) users
- Focussed: only this and nothing else
- Secure
- No GUI, everything happens through the commandline
- $\Rightarrow$ as small a footprint as possible

### 1.1.2 Workstation

- Use services
- Create documents
- Look for information
- Consume multimedia
- GUI
- $\Rightarrow$ Large footprint

## 1.2 Extra information/resources

- The Linux Documentation Project: `http://tldp.org`
- Pluralsight LPIC-1: Linux Professional Institute Certification: `https://www.pluralsight.com/paths/lpic-1`
- The Arch Linux Wiki is one of the most extensive sources of info about Linux: `https://wiki.archlinux.org`
  - In this module we will use Debian, not Arch, but many things are very similar
- Google

## 1.3 What is Linux?

### 1.3.1 What is an operating system (OS)?

**Definitie 1.1 (Operating System)** *An operating system, or OS, is software that communicates with the hardware and alows other programs to run.*

*It is comprised of system software = the fundamental files your computer needs to function.*

Linux is NOT an operating system: Linux = the **kernel**

### 1.3.2 What is a Kernel?

**Definitie 1.2 (Kernel)** *The kernel is software that is the core of a computer's operating system, with complete control over the system.*

*It is the first program loaded on start-up.*

*It handles. . . :*

- *. . . the rest of the startup*
- *. . . input/output requests from software, translating them into instructions for the CPU*
- *. . . memory*
- *. . . peripherals*

## 1.4 GNU Operating System

**Definitie 1.3 (GNU)** *GNU = GNU's Not Unix (recursive acronym)*

*Founded by Richard Stallman (ex-MIT, founder of the Free Software Foundation), 1984*

*Goal: completely free Operating System*

## 1.5 Linux, the kernel

By Linus Torvalds (Finland), 1991

- Own personal development, not initially intended to distribute
- Interest from other developers, mainly to use with GNU OS
- Meanwhile contributions of over 12000+ developers
- 492 of top-500 supercomputers in the world run Linux
- Basis for Android, Chrome OS

Linux = the kernel

GNU = OS-tools around the kernel

$\Rightarrow$ **GNU/Linux**

### 1.5.1 Distributions

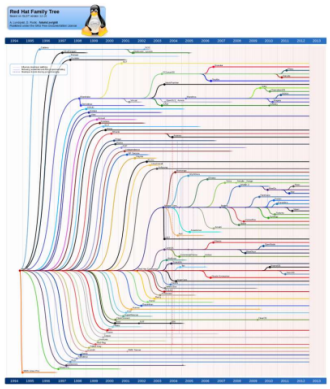
**Definitie 1.4 (Distribution)** *A Linux distribution (or distro for short) is GNU/Linux + extra tools and applications to create a full-fledged OS.*

*That distribution can be easily copied and installed to other computers.*

- RedHat (CentOS)
- Debian (Ubuntu)
- Arch Linux
- Void Linux
- Gentoo
- Pop! OS

Red Hat family tree

Debian family tree



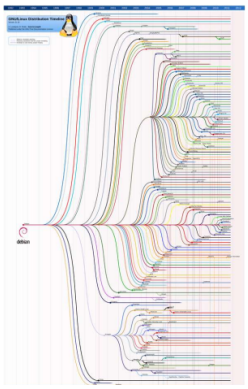


Figure 1: https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

https://en.wikipedia.org/wiki/List_of_Linux_distributions

## 1.6 Open Source

**Definitie 1.5 (Open Source)** *Open source software is software of which the code is licensed to be open to everyone.*

*Anyone can use, change, distribute the software. This allows code to be developed in a public manner.*

**OPEN SOURCE DOES NOT MEAN FREE**

### 1.6.1 Commercial distributions

= Open source, non-free distributions

- SUSE Linux Enterprise Server (SLES)
- SUSE Linux Enterprise Desktop (SLED)
- Red Hat Enterprise Linux (RHEL)
- Oracle Enterprise Linux

Commercial distributions have official support channels.

⇒ You're not paying for the operating system, you're paying for the support.

### 1.6.2 In this course: Debian

- Current version: 10.7
- Forms the basis of many others: Ubuntu, Raspbian, Knoppix, Linux Mint
- Available on many platforms: Intel x86, AMD64, Intel64, ARM, MIPS, Power Systems, ...

# 2 Debian Installation

**See Labs for detailed Installation tutorial**

## 2.1 Networking in Linux (with VMWare)

- VMWare presents ethernet adapter
- During creation of virtual machine: MAC-address is created
- During installation: network configuration through DHCP
  - IPv4-address
  - Default gateway
  - DNS-server
  - Optional: proxy-server

## 2.2 Users in Linux

- Linux is multi-user from the ground up
  - Multiple users can be active at the same time
- 'Administrator'-user is called root
- Each user has a user-ID (uid)
  - root has uid=0
  - uid=0 has all rights
- Each user has a home-directory

## 2.3 Disks, partition, filesystems

- Our VM has 1 disk
  - Presented on the SCSI-bus
  - First disk on SCSI-bus: **sda**
  - Then sdb, sdc, ...
- Disk = concatenation of blocks
- Divide blocks in collections (=partitions)
  - 1st partition: sda1
  - 2nd partition: sda2
  - ...
- 2 types of partitions
  - Primary
  - Extended

### 2.3.1 Partitions

Primary partition

- A filesystem can be created inside this
- Up to 4 primary partitions

Extended Partition

- 'Logical' partitions can be created inside this

Our setup:

- sda1: primary partition
- sda2: extended partition
- sda5: 'logical' partition inside extended partition sda2


Figure 2: Our setup

## 2.4 Partitioning schemes

= a set of rules describing how a disk should be partitioned. A disk partitioning scheme is chosen by desired flexibility, speed, security, and necessary disk space.

### 2.4.1 MBR

We use the MBR Partitioning scheme

**Definitie 2.1 (MBR)** *MBR, or Master Boot Record, is a special type of boot sector at the start of a disk.*

*It contains:*

- *a set of instructions necessary to boot operating systems.*

- *info about how partitions are placed on disk*

**Limitations:**

- Maximum disks of 2TB
  - 32-bit for number of logical sectors
  - Common sector size: 512 bytes
  - $2^{32} \cdot 512$ bytes $= 4294967296 \cdot 512$ bytes $\approx 2$TB
- Maximum amount of primary partitions = 4

BIOS can boot from a disk with MBR partitioning

### 2.4.2 GPT

**Definitie 2.2 (GPT)** *GPT, or GUID Partition Table, is a standard for the layout of partition tables on a disk. It's an alternative to MBR.*

*It uses unique identifiers (GUIDs)*

- BIOS cannot boot from a disk with GPT-partitioning: UEFI required when using GPT
- GPT allows disks larger than 2TB

**Definitie 2.3 (UEFI)** *UEFI, or Unified Extensible Firmware Interface, is a newer firmware interface by Intel (90's) that replaces the BIOS interface by IBM (70's).*

**How does it work?**

- Disk = collection of blocks
- Group of blocks together = sector
- Common sector size: 512 bytes
- Sectors indicated with Logical Block Addresses (LBA)
- MBR in LBA 0
- GPT headers in LBA 1
- Partition tabel right after that

### 2.4.3 Bootstrap procedure

1. Motherboard gets electricity
2. Mini-loader hardcoded in memory
   - BIOS gets loaded
3. Boot media are consulted
4. First boot medium, first sectors are being read $\Rightarrow$
5. MBR contains a bit-more-advanced loader: GRUB
   - GRand Unified Bootloader
6. This loader loads a more advanced loader (GRUB second stage bootloader)
7. The OS is loaded

### 2.4.4 Linux boot process

6 high level steps

- BIOS (Basic Input/Output System) - loads MBR
- MBR (Master Boot Record) - loads GRUB
- GRUB (Grand Unified Bootloader) - loads kernel
- Kernel - executes /sbin/init
- Init - executes runlevel programs
- Runlevel - programs from /etc/rc.d/rcXX.d are started

### 2.4.5 BIOS <> UEFI

- Recent systems use UEFI, not BIOS
- UEFI is required to boot from GPT-disk
- Linux has no trouble working with UEFI

**So why will we use MBR?**

- Virtualisation is the norm
- Virtual machines typically have small disks
- Small disks are MBR partitioned

## 2.5 Filesystems

### 2.5.1 Windows

- FAT (1977)
- FAT32 (1996)
- NTFS (1993)
- ReFS (2012)

### 2.5.2 Linux

- Ext (1992)
- Ext2 (1993)
- Ext3 (2001)
- Ext4 (2008)
- ZFS (2005)
- BtrFS (2007)

### 2.5.3 Swap

= Paging

- Free up physical memory (RAM) by moving pages to slower storage (storage disks instead of RAM)
- Page out = memory page moves to swap
- "Swapiness"
  - = parameter between 0 and 100
  - = how quickly linux will swap
    - 0 = very conservative
    - 100 = very agressive
- Windows uses a swap file (pagefile.sys)
- Linux uses a swap partition

## 2.6 File structure

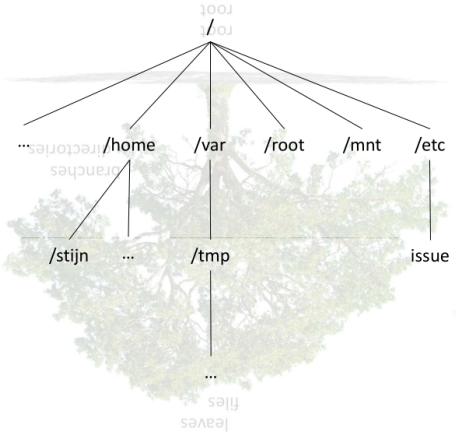



Figure 3: Linux uses a tree structure

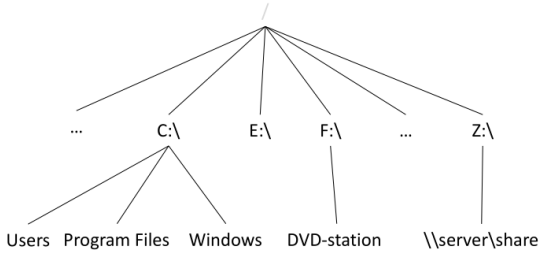



Figure 4: Windows uses a similar structure, but every volume uses a letter.

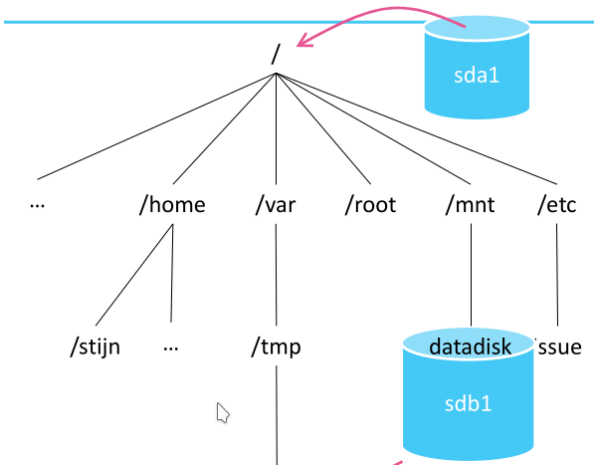



Figure 5: With linux, volumes are 'mounted' to folders somewhere under root /

## 2.7 Configuration

### 2.7.1 Packages

- Tools and applications are build up by files
- All files belonging to 1 application are bundled in a package
- Packages in debian have the .deb extension

Repositories

- Packages are collected in repositories
- Are made available through the internet
- Packages have dependencies

### 2.7.2 Package management

Debian: dpkg & apt (Advanced Package Tool)

- dpkg: Install, remove, give info about .deb packages
  - dpkg -l = lists packages
- apt: Get packages from a repository and install, remove, give info, ...
  - apt update
    - Contact the repositories
    - Get most recent list of packages and versions
  - apt upgrade
    - Of the packages which are more recent in the repositories compared to what is installed: install newest version
  - apt install <xyz>
    - Download package <xyz> from the repository

    * Check the dependencies and download depending packages
    * Install package <xyz> and all corresponding dependencies

Which repositories? See /etc/apt/sources.list for the list of repositories. You can add/remove/change repositories in this file.

### 2.7.3 Useful packages

- open-vm-tools
- vim
- sudo
- tcpdump

Install multiple pacakges in one command: apt install vim sudo tcpdump ntp

## 2.8 Shutdown of VM

- Power button (=ACPI shutdown)
- Shut down operating system only
  - = halt
- Shut down operating system and VM, multiple ways:
  - shutdown -P now
  - init 0
  - poweroff
- Reboot
  - reboot
  - init 6
  - shutdown -r now

## 2.9 Basic network

- No GUI $\Rightarrow$
- Layer 1: Physical (VMWare virtual network)
- Layer 2: Datalink (Ethernet & MAC address)
- Layer 3: Network (IPv4)
- Layer 4: Transport (Transport Control Protocol (TCP), User Datagram Protocol (UDP))
- Layer 5: Application (SSH, HTTP, . . . )

### 2.9.1 Basic networking commands

- arp
- ping
- route
- bmon

## 2.10 Services

- Processes that ‘listen’ on the network
  - TCP or UDP port
- Overview of currently running / listening services: ss command
  - ss -tulpn
  - t: show TCP
  - u: show UDP
  - l: show listening
  - p: show process ID
  - n: no name-resolving

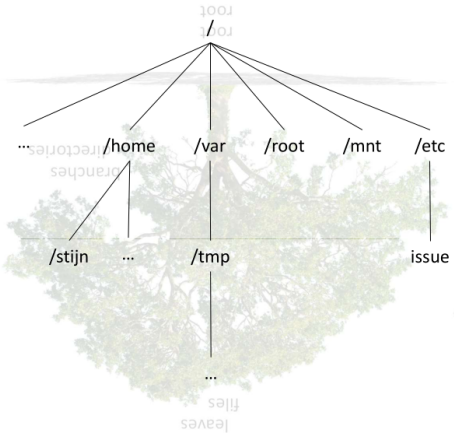
## 2.11 Wooclap Questions

- Why do we talk about GNU/Linux?
- What is a kernel?
- What is the difference between Open Source and free?
- How is the Administrator user called? What is its uid?
- What is MBR?
- What are the limitations of MBR? (Solution?)
- What is swap? What is swappiness?
- What is a package?
- What is a repository?
- What is a dependency?
- What is a package manager?
- What is the difference between ’apt update’ and ’apt upgrade’
- Which protocol makes the link between MAC address & IP address?
- Which command gives you the current ARP-table?
- What are the 5 layers of the TCP/IP network model?
- How do you find the MAC-address of a network interface?
- Put Linux boot process in correct order (6 levels)

- What is a linux distribution?

# 3 File structure

- Tree structure
  - Leaves = files
  - Branches = directories
  - The tree is inverted, root = /
- Everything is a file (even devices, random numbers, and RAM) under 1 root
- This is in contrast to Windows, where every volume is a root.




## 3.1 Intermezzo: single user mode

- Linux (the kernel) is built up as a multi-user system from the beginning
- Standard behaviour = multi-user
- But: also possible to boot in single-user mode
  - No daemons, no multiple logins
  - Sometimes called **Maintenance mode**
- Examples of usage
  - Filesystem repairs
  - Upgrade of distribution
  - Password recovery
  - Adjustments to the root filesystem
  - Forensics after security incident

### 3.1.1 Runlevels

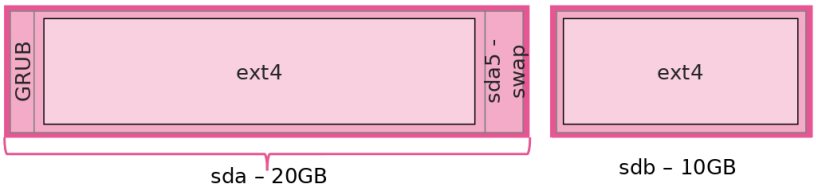
= predefined operating system status

- Is presented with a number
- Linux has 7 runlevels:
  - 0 = system halt (= VM shutdown)
  - 1 = single user
  - 2 = multi-user, no NFS (no network services, not often used)
  - 3 = multi-user, CLI (Command Line Interface)
  - 4 = self-definable
  - 5 = multi-user, GUI (Graphical User Interface, if installed)
  - 6 = reboot

## 3.2 Intermezzo: Add disk

**Add a new disk without shutting down the system**

1. Adjust VM: add disk
2. Detect added disk
3. Partition disk
   - fdisk (for MBR)
   - parted (for GPT)
4. Create filesystem
   - Partition = collection of blocks (sectors)
   - Not usable for the OS ⇒ create filesystem
   - mkfs.ext4 /dev/sdb1




5. Mount filesystem
   - mkdir /mnt/datadisk
   - mount /dev/sdb1 /mnt/datadisk
   - see if it worked: df -h

**For detailed steps: see labs!**

### 3.2.1 What after a reboot?

Use /etc/fstab = a file that contains what needs to be mounted at boot

- Device (/dev/sdXY or UUID)
- Mountpoint (/mnt/folder)
- Type of filesystem (ext4, ntfs, . . . )
- Options

## 3.3 Navigate through the tree

- pwd
  - Print working directory
  - Shows where in the tree you are
- ls
  - Show a list of files in the working directory
  - ls -la : 10 characters at the beginning of each line. The d == directory (see later)
- When you login, you are in your home directory
- / (= the filesystem root) is not the same as /root (the home directory of the root user)
- . = current directory
- .. = the directory one higher

### 3.3.1 Relative vs absolute path

Relative paths:

- cd .. = go to the directory above the current directory
- cat ../etc/issue = go to the etc directory, one directory above the current directory. Open the issue file

Absolute paths:

- cd / = go to the root directory
- cat /etc/issue = go to the etc/ directory under / (root)

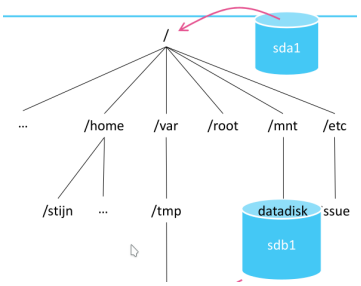
## 3.4 Filesystem Hierarchy Standard (FHS)

- Describes how the filesystem in Linux is build up
- Maintained by the Linux Foundation
- Most recent version: v3.0 (2015)

### 3.4.1 Rules in the standard

- / is the root of the tree structure
- /bin
  - essential binaries (executable files), required for single user mode
- /boot
  - the place on the filesystem where the boot files reside
  - configuration files for GRUB
  - kernels
  - initrd
    - initial RamDisk
    - During boot a temporary root-filesystem is being created in RAM
    - This is used so the kernel can load important modules, so it can then switch to the real root filesystem
    - part of step 3 of the linux boot process (BIOS - MBR - GRUB - kernel - init - runlevel)
- /dev
  - Devices get a place in the filesystem
    - sda
    - rtc
    - random
    - cpu
    - urandom
    - null
  - ls -lah /dev/
- /etc
  - Host-specific sytem-wide configuration files
  - Configuration for this host, readable for the whole system
- /home
  - Each (non-system) user has a home directory
  - except for root $\Rightarrow$ /root
- /mnt
  - (temporarily) 'mounted' filesystems
    - Network shares
    - USB-disk
    - DVD-ROM
    - Extra disks

  - Some distributions use /media for this




- /opt
  - Optional application software packages
  - Our installation ⇒ no applications installed yet = empty (for now)
- /proc
  - Virtual filesystem
  - Provides information about processes and the linux kernel
  - cat /proc/cpuinfo
  - cat /proc/sys/net/ipv4/ip_forward
  - cat /proc/partitions
- /sbin
  - Essential system binaries
  - Only executable by root user
  - fsck, init, route
- /tmp
  - Directory for temporary files
  - Emptied at reboot (with most distributions)
- /usr
  - Read-only user data
  - Constains most user (non-root) utilities and applications
- /var
  - Variable files
  - Files that are expected to change continuously during normal system use
  - Logs, spool files, temporary e-mail files, . . .

## 3.5 Some useful tips

### 3.5.1 History

```
~# history

# shows a list of former commands executed by this user
# spans log-in sessions
# in reality, it shows the contents of the ~/.bash_history file
# if you use another shell like zsh, it's the ~/.zsh_history file
```

CTRL + r:

- Search the command history
- Show commands that match what you're typing
- repeatedly press ctrl+r to scroll through results

### 3.5.2 Bind mount

**Situation**

- /mnt/storage is the normal mountpoint for other filesystems (e.g. SAN)
- Filesystem could not be mounted, but a process already started writing data
- ⇒ this data arrives on the / filesystem under the directory /mnt/storage
- Problem fixed and filesystem can be mounted again ⇒ mounted under /mnt/storage
- ⇒ the already written data is now hidden

**The solution**

- Create /mnt/storage and put some data in it
- Create a 1GB disk, ext4 formatted, mount under /mnt/storage ⇒ data is now hidden
- Use mount -o bind to get data back without unmounting

### 3.5.3 dd

= Command to read or write bytes

```
# Example: overwrite first 2048 bytes of a disk with zeros
~# dd if=/dev/zero of=/dev/sdb count=4 bs=512
# Example: overwrite disk with random data when taking out of service
~# dd if=/dev/random of=/dev/sdb bs=1M
```

## 3.6 Wooclap Questions

- How do you ask the shell in which folder you are currently in?
- What is meant with the term 'runlevel' in Linux
- Describe single user mode with 1 word when you think of its primary use
- What is / are the most common runlevel(s) under linux? (So not all of them!)

- Where can you find the devices under Linux?
- What is the home directory of the root user?
- What command do we use to create a filesystem in a partition?
- What file do you need to edit to have a mounted filesystem available even after reboot?
- Where can you put temporary files in a linux system?
- How can you quickly search through your previously used commands?
- How do you quickly search through previously typed commands?
- What is swap?
- What are the limitations of MBR?
- What can you use a bind mount for?

# 4 Filesystems

## 4.1 Introduction

Books:

- A group of letters together = a word
- A group of words together = a sentence
- A group of sentences together = a book
- A collection of books together = a library
- Books are ordered/sorted according to a certain system
  - Best known: Dewey Decimal System

Computers:

- Work with 0's and 1's
- 1 character in ASCII or ISO-8859-1 = 8bits (1 byte)
- 1 Unicode character in UTF-8: between 8 and 32 bits (4 bytes)
- Gets stored on block devices
  - Hard devices, SSDs, RAMdisk, USB-stick
  - The opposite of block devices = character devices
- System needs to organize this

## 4.2 Blocks

- Disk = blocks
- Collection of blocks = sector (mostly 512 bytes)
- Collection of sectors = partition
- Partition not usable for an OS $\Rightarrow$ filesystem needed

**Definitie 4.1 (Filesystem)** *A filesystem is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk.*

Several choices:

- Ext2/3/4
- BrtFS
- ZFS
- . . .

## 4.3 ext2/3/4

Ext3 and Ext4 have journaling:

### 4.3.1 Journaling

- Keeping track of changes that have not been committed to disk in a sort of 'diary'
- A kind of logbook of previous actions
- Why?
  - Bring filesystem online faster after system crash or power failure

## 4.4 RAID

**Definitie 4.2 (RAID)** *Redundant Array of Independant Disks (RAID) is a data storage virtualisation technology that combines multiple physical disk drive into one ore more logical units.*

*Many purposes:*

- *Data redundancy*
- *Performance Improvement*
- *Both*

### 4.4.1 RAID Controller

- Disks are connected to the controller
- The RAID controller displays the disks as 1 disk to the OS
- Nowadays, we call the RAID Controller the Host Bus Adapter (HBA)
-

### 4.4.2 RAID 0

RAID level 0 uses striping:

**Definitie 4.3 (Striping)** *Data striping is the technique of segmenting logically sequential data (files) so that segments are stored on different physical storage devices*

*Purpose:*

- *Increasing data throughput*

- *Balancing I/O load accross an array of disks*

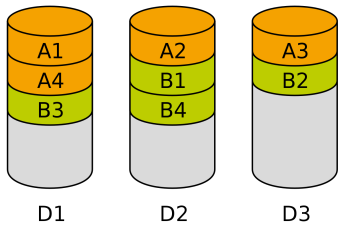



Figure 6: Example: files A and B (4 blocks each) are spread over disks D1-D3

### 4.4.3 RAID 1

RAID level 1 uses mirroring:

**Definitie 4.4 (Mirroring)** *Disk mirroring is the replication of logical disk volumes onto seperate physical disks.*

*Purpose:*

- *Continuous availability: in case of hardware failure, you always have a backup of your data*
- *Increasing read speeds*

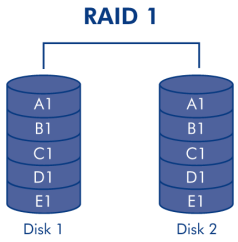



Figure 7: RAID 1

### 4.4.4 RAID 4

- If we have at least 3 disks
- For every block of data:
  - Divide the block in 2 halves: A and B
  - Write A to disk 1
  - Write B to disk 2
  - Write A+B to disk 3
- $\Rightarrow$ RAID 4 is striping (disk 1 & 2) with parity (disk 3)
- Capacity x2
- Read speed x2
- Write speed is limited, because of the need to write all parity data to a single disk

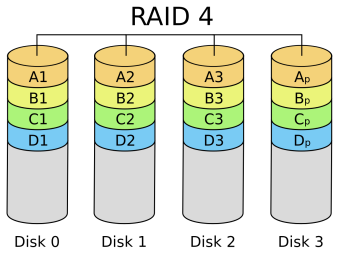



Figure 8: A RAID-4 setup with 4 disks. Disk 3 is the parity disk

### 4.4.5 RAID 5

RAID level 5 like RAID 4, but the parity is distributed.

- This evens out the stress of a dedicated parity disk (RAID 4)
- Write performance is increased

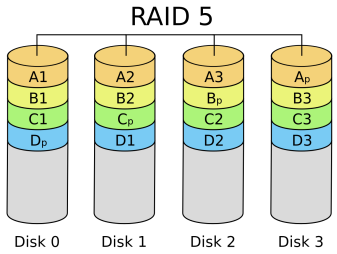
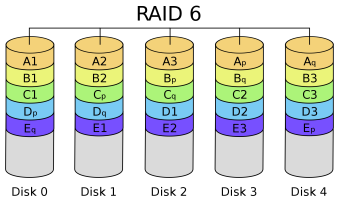



Figure 9: RAID-5: distributed parity with 4 disks

### 4.4.6 RAID 6

RAID level 6 like RAID 5, but with a second parity block.




### 4.4.7 Disk failure

For every RAID level, a certain amount of disks can fail without it becoming a problem:

- RAID 0: no disks can fail: if any disk fails, you lose data
- RAID 1: Every disk except for one can fail
- RAID 5: 1 disk can fail
- RAID 6: 2 disks can fail

### 4.4.8 Compound RAID levels

Combining RAID levels is possible:

- RAID 10 = RAID 1 + RAID 0
- RAID 01 = RAID 0 + RAID 1
- RAID 50 = RAID 5 + RAID 0

## 4.5 OpenZFS

### 4.5.1 ZFS

- Zettabyte File System
- Developed by Sun (2011) $\Rightarrow$ Open source
- Now Oracle (2010) $\Rightarrow$ Not free and closed source

### 4.5.2 Open ZFS

- Fork van ZFS
- 2013
- Option in Ubuntu-installer

Features:

- Long term storage
- Checksum of all data and metadata
- Native RAID levels (0, 1, 5, 6, . . . )
- All data gets written through Copy-On-Write:
  - COW = when a write request is made, the data is copied into a new storage area, and then the original data is modified.
  - Redirect-on-write or ROW: the original storage is never modified. When a write request is made, it is redirected away from the original data into a new storage area.
- Snapshots (read-only and mountable)
- Transparent compression
- Huge storage possibilities: up to 256 quadrillion zettabytes
- 128 bits system

## 4.6 Intermezzo: Kernel modules

- Linux = kernel
- Kernel = modular
- /boot/config-4.9.0-13-amd64: config for this kernel
  - Describes what is inside this kernel
- Not all modules are loaded all the time

### 4.6.1 Commands

```
# Request current list of modules:
~# lsmod

# Load module:
~# modprobe brtfs

# Remove module ("unload"):
~# rmmode brtfs
```

## 4.7 Intermezzo: Snapshots

- Literally: a photograph of your filesystem
- Captures a the state of the filesystem at a certain point in time
- "The possibility to return in time"

### 4.7.1 Do we still need backups if we have snapshots?

YES!

- RAID 1 (mirroring) only protects against disk failure, nothing else
- If someone deletes all data from one disk, the RAID controller will delete all data from the other disk.
- Snapshots can get lost: what if your server fails?
- $\Rightarrow$ backups can be stored safely, on other disks

# 5 File manipulation

## 5.1 Basics

```
# create an empty file called 'test'
~$ touch test

# edit a file
~$ vim test

# remove file
~$ rm rabbot

# move the file to /tmp
~$ mv test /tmp/

# rename the file
~$ mv test rabbit

# Linux doesn't really look at file extensions
# check the file extension:
```

```
~$ file <filename>
~$ file /boot/inird.img-4.9.0-13-amdb64
~$ file /etc/init.d/networking
```

## 5.2 Bundle files

- Tape ARchiver: TAR
  - Created originally to bundle files/directories for storage on tapes
- You can combine tar with gzip: .tar.gz
  - tar cfv bundle.tar *.txt ⇒ not compressed
  - tar czfv bundle.tar.gz ⇒ compressed

```
~$ mkdir bundle
~$ cd bundle
~$ touch 1.txt 2.txt 3.txt
~$ tar cfv bundle.tar *.txt
# c = create a new archive
# f = specify a filename (bundle.tar)
# v = verbose: show what happens
~$ tar --list -f bundle.tar
~$ file bundle.tar

# extracting
~$ tar zxvf bundle.tar.gz
# z = zipped (compressed)
# x = eXtract
# v = verbose
# f = the argument (a file)
```

## 5.3 Links and inodes

- Modern filesytems support links
- This is different from shortcuts in Windows:
- Windows shortcuts are text files that refer to other files

### 5.3.1 Inodes

**Definitie 5.1 (Inode)** *An inode is a data structure on a filesystem on Unix-like operating systems that stores all the information about a file except its name and its actual data*

*Metadata: data about the file*

- *Creation date*
- *Creation author*
- *Access rights*
- *...*

### 5.3.2 Symbolic links

**Definitie 5.2** *A symbolic link (also symlink or soft link) is a term for any file that contains a reference to another file or directory in the form of an absolute or relative path*

*Also called 'softlinks'*

```
~$ ln -s <target> <link-name>
~$ ln -s /etc/issue test-link
# try out the following commands after creating a link:
~$ cat test-link
~$ file test-link
~$ cat /etc/issue
```

### 5.3.3 Hardlinks

- Same file, different name
- A hardlink refers to an inode, while a softlink refers to a file (which refers to an inode)

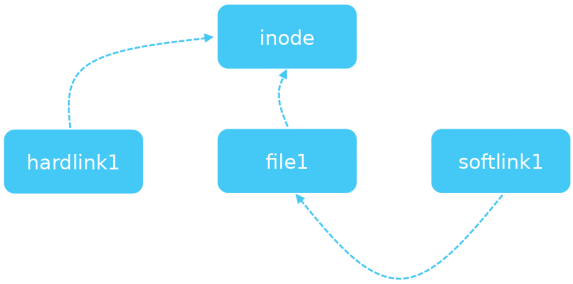



Figure 10: Symlink vs Hardlink

## 5.4 File permissions

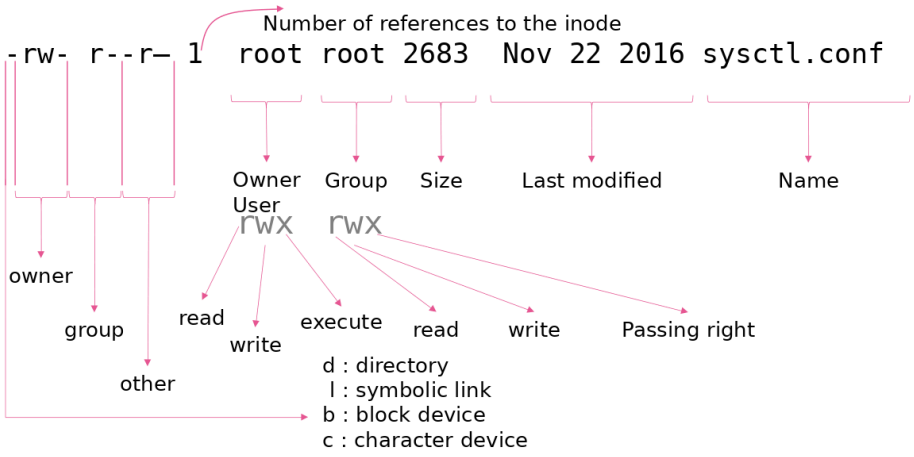



Figure 11: The output of 'ls -l' creates this type of output

1. first character: type of file (directory, symbolic link, block device, character device)

2. next 9 characters: owner rights, group rights, other rights
3. next number: number of references to the inode
4. owner user
5. owner group
6. size of file
7. last modified
8. name of file

```
# change the owner and group of a file or directory
~# chown <user>:<group> <file>
~# chown root:staff file.txt

# change the rights for a file
# chmod: change mode
```

```
0 --- indicates no permissions
1 --x indicates execute permissions
2 -w- indicates write permissions
3 -wx indicates write and execute permissions
4 r-- indicates read permissions
5 r-x indicates read and execute permissions
6 rw- indicates read and write permissions
7 rwx indicates read, write, and execute permissions
```

Figure 12: Octal notation

## 5.5 Overview of basic commands

Usage and details for these commands: see labs

- cat: print contents of file to terminal
- cut: cut (structured) input on a specific place: show a certain column, etc. . .
- grep: display lines for which the pattern matches
- egrep: extended grep, better handling of regular expressions
- find: search for files in a hierarchy of files and directories
- head: show first lines of file
- tail: show last lines of file
- less: show the contents of a text file, interactively
- man: show manual page for specific command
- wc: word count (but also character count, byte counts, newline counts, . . . ) for a file
- date: show or configure system date and time
- cal: show a textual calendar
- sort: sort a file
- uniq: in a sorted output: count double lines or only show unique lines

## 5.6 Wooclap

- What is meant by the term journaling for filesystems?
- Why is journaling used with filesystems?
- Give 2 examples of filesystems under linux that use journaling.
- How can you find out which kernel modules are currently loaded?
- Which command can you use to load a kernel module?
- And which to 'unload' a kernel module?
- How many disks do you need at least to build a RAID10 system? Why?
- What is meant by a 'Copy On Write' filesystem?
- What are the advantages of a CoW filesystem?
- What are snapshots (in the context of storage systems)?
- What are the disadvantages of a CoW filesystem?
- Why do you still need backup when you have RAID1 and have snapshots?
- How can you find out which 'type' is a file? There are no extensions.
- What is an inode?
- What is the difference between a softlink and a hard link?
- At the output of the command ls -la: Which values can the first character of the line have and what do they mean?
- At the output of the command ls -la: Which possible values can the 3 groups of 3 characters have to describe the rights?
- With what command can you 'change' the 'owner' of a file or directory?
- With which command can you 'change' the rights of a file or directory?
- What does number 5 mean when you use it to determine file system permissions?
- What does number 7 mean when you use that to determine file system rights? Explain why.
- Which command do you use to cut structured input at a specific location?
- Which command do you use to display the first 16 lines of a text file?
- Which command can you use to find out all the modified files from the last 24 hours?
- Which command do you use to display the last 12 lines of a text file?
- How can you find out how long it has been since a linux system was rebooted?
- Which command can you use to get an overview of all daemons that are currently active in your system?

# 6 Text editors, Piping, Redirection & Jobs

## 6.1 Text editors

- Emacs (productivity, extensibility)

- Nano (simplicity)
- Vi / Vim (=VI iMproved) (productivity)
- Ne

Our choice: Vim

### 6.1.1 vi vs vi-improved

- Navigating in vi: HJKL (left, down, up, right)
- Navigating in vim: HJKL or arrow keys

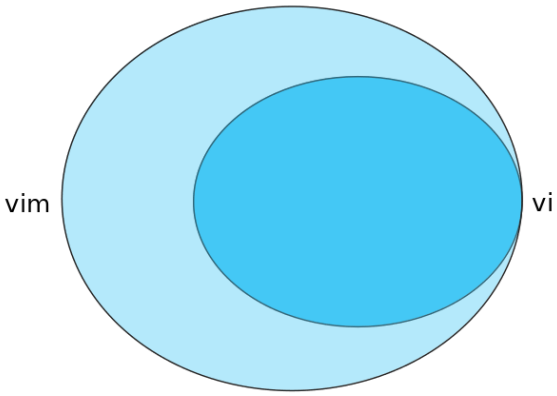



Figure 13: Everything you can do in vi, you can do in vim, and more!

### 6.1.2 First steps in vim

```
# start vim:
$~ vim

# start vim tutorial:
$~ vimtutor
```

- bottom left of window: the current mode
- INSERT = the mode that lets you enter text
- Enter INSERT mode: i
- Leave INSERT mode and go to normal mode: ESC
- Once in normal mode, you can enter commands using ':'
- ESC : q ⇒ quit
- ESC : w ⇒ quit
- ESC : wq ⇒ write and quit

```
# copy a line of text
ESC
# put cursor on the line you want to copy
yy # yank yank: copy a line of text
```

```
p # put: paste the copied line

# copy 2 lines of text and paste it 8 times:
ESC 2yy # 2 yank yank: yank 2 lines
8p      # 8 put: paste the copied lines 8 times
```

### 6.1.3 Search and replace

You can easily search and replace in a text file, even with regex:

```
# search and replace the next instance:
:s/old/new

# all instances: TODO: difference between :s and :%s
:s/old/new/g

# all instances between line x and y:
:x,ys/old/new/g

# all instances in a complete file:
:%s/old/new/g

# all instances in whole file, with confirmation:
:%s/old/new/gc

#undo:
(ESC) u
```

### 6.1.4 Basic editing tricks

```
# starting from line 4, indent the next 7 lines:
(ESC) 7>>

# remove indentation on line 10
(ESC) 10gg
<<

# delete line 3
(ESC) 3gg
dd

# delete the next 4 lines
(ESC) 4dd

# enable and disable syntax highlighting in vim
(ESC) : syntax on
(ESC) : syntax off
```

For more tricks: see labs

## 6.2 Piping

= Use the output from one command as input for the next command

```
# sort the music file alphabetically and count the number of occurences of each unique line
sort music.txt | uniq -c

# count the number of unique lines in music.txt
sort music.txt | uniq | wc -l

# count the number of lines in /etc/locale.gen where nl or NL occurs
grep -i nl /etc/locale.gen | wc -l

# count the number of lines in /var/log/syslog where kernel occurs
cat /var/log/syslog | grep kernel | wc -l

# count the number of lines in /var/log/syslog where kernel does NOT occur
cat /var/log/syslog | grep -v kernel | wc -l

# Show of what days there are logs in /var/log/syslog
# the sixth field is the day field:
cat /var/log/syslog | cut -c 6 | uniq

# show the different sources of log entries in /var/log/syslog
# kernel, client, systemd, ...
cat /var/log/syslog | cut -d' ' -f5 | cut -d'[' -f1 | sort | uniq -c
```

## 6.3 Redirection

Do not send the output of a command to stdout, but to another location, like a textfile

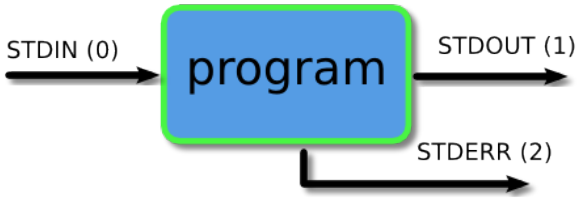
```
# to overwrite a file (or create if it doesn't exist)
ls -la > listing.txt

# to append to a file (or create if it doesn't exist)
ls -la >> listing.txt

# these two commands have the same result
cat > textfile.txt
touch textfile.txt

# redirect the output of 'ls -la' to a file with a custom name:
# example: output_2021-03-03
ls -la > output_$(date  +"%F")
```

### 6.3.1 stdout and stderr

= 2 important output streams

- Normal situation: stdout and stderr appear on the terminal
- Redirection: stdout to a file

- stderr still prints to the terminal
- Redirect stderr: 2> errorfile.txt




```
# redirect the output of a command to out.txt
# and redirect the error of the command to error.txt:
ls -la > out.txt 2> error.txt

# redirect stderr to stdout (&1), and then redirect stdout to a file out.txt:
ls -la > out.txt 2>&1

# redirect both to a file:
ls -la &> out
```

### 6.3.2 stdin

```
# this command prints the amount of lines in a file
wc -l music.txt

# this command does the same
wc -l < music.txt

# this command does the same, but prints the output of wc to out.txt
wc -l < music.txt > out.txt
```

## 6.4 Jobs and process Management

When you execute a command: a process is started

- Every process gets a process ID (PID)
- The init process has PID 0. It starts other processes.
- Every process has a parent
  - $\Rightarrow$ tree structure of processes
  - Get insights into this structure with 'pstree' (part of the 'psmisc' package)
  - Process stops: exit code is passed to the parent

```
# report a snapshot of current processes
ps

```

```
# display a tree of processes
pstree -p
```

### 6.4.1 Exit codes

```
# when a process stops:
# if bash was parent => exit code is passed to bash
# exit code is available in the $? variable:
# read contents of a variable with echo:
echo $?
0

# 0 = ended successfully without errors

# 1 = not ended successfully, there were errors
```

```
# test = verify if you have rights on a file (with -r = read rights)
~$ test -r music.txt

~$ echo $?
0

~$ test -r doesnotexist

# the file doesn't exist, so it exited with code 1:
~$ echo $?
1
```

### 6.4.2 Combining commands

= not the same as piping!

```
# note the difference between one and two ampersands:
# & = execute command 1, then execute command 2
# && = execute command 1, and only execute command 2
#   if command 1 was successful (exit code = 0)

# one ampersand
test -r test.txt & echo "MCT rocks"

# two ampersands
~# test -r test.txt && echo "MCT rocks"

~# test -r doesnotexist & echo "MCT rocks"

# the first command will exit with code 1
# so the second command will not execute
~# test -r doesnotexist && echo "MCT rocks"
```

### 6.4.3 Jobs

A job is a new process originating from the same parent

```
# start a command as job
tail -f /var/log/syslog &
# output appears on stdout, but process runs as a job in the background

# bring a job to the foreground
fg <index>

# stop the job, but do not terminate:
CTRL-Z
```

### 6.4.4 Inter-process Communication

A signal is an asynchronous notification sent to a process or thread within that process to notify that there has been an event

Signal sent to process ⇒ OS interrupts normal execution of that process to deliver the signal

Sending a signal to a process: with the 'kill' command:

```
# not only to kill a process, also to send other signals
kill -s <signal>
```

**Signals**

- SIGHUP - 1 - terminate (hang up)
- SIGINT - 2 - terminal interrupt signal
- SIGKILL - 9 - kill (cannot be caught or ignored)
- SIGTERM - 15 - termination signal

```
# send signal 15 to a process with the entered PID
kill -s 15 <pid>

# send signal 15 to the PID of the tail process
kill -s 15 'pidof tail'

# send signal 9 to a process with the entered PID
kill -s 9 <pid>

# kill the process with name 'tail'
pkill tail
```

## 6.5 Intermezzo: System Load

= a number which represents the load on a computer system

- Completely idle system: system load 0
- Each process which uses a resource or is waiting for a resource: system load + 1

- Gives an indication of how heavy a computer system is loaded
- System load is a snapshot, doesn't say anything
  - System load of 17: is that a problem? No.
  - More interesting: the evolution of the systemload over time

```
# show the system load of the last minute, last 5 minutes and last 15 minutes:
uptime

# show who is logged on and what they are doing
w

# display linux processes
top
# or better:
htop
```

## 6.6 Some useful tips

### 6.6.1 With which unique IP-addresses are there open sockets and how many?

```
netstat -anpt | awk '{print $5}' | sort | uniq -c
```

### 6.6.2 TTY

= Tele TYpewriter = a terminal which is connected with stdin

```
# print the filename of the terminal currently connected to standard input:
~$ tty
```

## 6.7 Answer these questions to test your knowledge

1. What does piping mean?
2. What is the prerequisite for using the command 'uniq'
3. What is the difference between $>$ and $>>$
4. Which are the 2 output streams in linux, and what do they contain?
5. What is 'exit code'
6. Why is the exit code useful
7. What different values can the exit code be?
8. What is the exit code for success?
9. How do you request the exit code?
10. How do you turn a command into a job?
11. How do you pause a job
12. What command shows the list of all running or paused jobs?

13. How do you re-activate a paused job?
14. What does the command screen do?
15. What is meant by the term ‘signal’?
16. How do you send a signal to a process
17. How do you specify which process?
18. Give 2 examples of signals
19. Is a systemload of 23 problematic
20. What process has PID 1?
21. Where do you know this process from?

# 7 Regex, users & firewall

## 7.1 Regex

- Make the webscaper assignment
- Use `www.regexr.com` to test regex

## 7.2 User management

### 7.2.1 Add a user

- Root privilege required!
- There are two common commands to add users:

```
adduser testuser1
# enter password
# name, other details
# /home/testures1
# contents of this directory follows /etc/skel
# group: testuser1
# default shell = /bin/bash

useradd testuser2
# user is created , no password, no shell
```

### 7.2.2 Delete a user

Again with root privileges, and 2 possible commands:

```
deluser testuser1
# user and corresponding group get deleted
# home directory not deleted

userdel testuser2
# user and corresponding group get deleted
# home directory also deleted
```

### 7.2.3 Change password of user

```
# change your own password
passwd
# enter current password
# enter new password x2

# change password of other account (root privilege required)
passwd <username>
# current password not needed, only new password x2
```

### 7.2.4 Create a new group

Root privilege required!

```
addgroup testgroup1
# GID = group ID

groupadd testgroup2
```

### 7.2.5 Assign a user to a group

```
usermod -aG testgroup1 testuser1
# add to group, groupname, username

# overview of groups and group members
cat /etc/group

# of which groups is the current user a member?
~# groups
~# su - testuser1
~$ groups
```

### 7.2.6 Overview of users

/etc/passwd

- Username
- User ID
- Home directory
- Default shell
- Full name
- ...

### 7.2.7 Overview of groups

/etc/group

- Groupname

- Group ID
- Group members

#### 7.2.8 /etc/shadow

- System-wide host-specific configuration file (because it's in /etc/)
- Contains information about password length, expiration, password hash
  - Mostly: sha512-hash
- Assignment:
  - write a one-liner
  - Get all usernames from /etc/shadow and save them in /tmp/usernames
  - Extract from /etc/shadow all usernames and hashes of passwords for the usernames where a hash of the password is known

#### 7.2.9 /etc/gshadow

- Contains an encrypted password for each group, as well as group membership and administrator information
- Group name: the name of the group
- Hashed password:
  - If the value of this field is !, no user is allowed to access the group using the newgrp command.
  - If the value of this field is !! : same things as !, but it also indicates that a password has never been set before
  - If the value is null, only group members can log into the group
- Group administrators: users who can add or remove group members using the gpasswd command
- Group members

#### 7.2.10 sudo

- Root privilege = superuser
- sudo = superuser do
- $ sudo useradd jeff == # useradd jeff

```
# decide who gets sudo rights:

# install the sudo package
apt install sudo
# change the rights
visudo
# view the changes
cat /etc/sudoers
```

There is a sudo group:

- Called sudo, sudoers or sometimes 'real'
- Members of sudo group have sudo rights
- Add user: # usermod -aG sudo <username>

### 7.2.11 Temporarily become another user

```
sudo su - <username>
# this switches the user to <username> and use the environment of that user
# eg. the $PATH

# become root:
~$ sudo su -
# without username => root
```

## 7.3 SSH

Secure SHell

- In the past: telnet (23/TCP)
  - Plain text
  - Not feature-rich
- Better: ssh (22/TCP)
  - Encrypted with public-private key cryptography
  - Feature-rich
  - Connect through the network, login as a user and get access to the shell on the remote system as that user
  - You're sitting at the console of that system, but with 'a very long cable'

### 7.3.1 SSH features

- Log in and get remote shell. Which shell? See /etc/passwd
- Invoke a command on the remote system
- Create a secured, encrypted tunnel
- Transfer files: with scp (secure copy)

### 7.3.2 SCP - Secure Copy

- Transfer files over an encrypted connection (22/TCP)
- scp <what to copy> <where to copy to>
- Also specify the username, the remote system and the location on that system

### 7.3.3 SSH tunnels

Tunnel a port to let traffic through a certain port

```
ssh -L <local-IP>:<local-port>:<remote-ip>:<remote-port> username@ssh-server

# SOCKS-proxy functionality:
ssh -D <local-port> username@ssh_server
```

## 7.4 Full-fledged environment

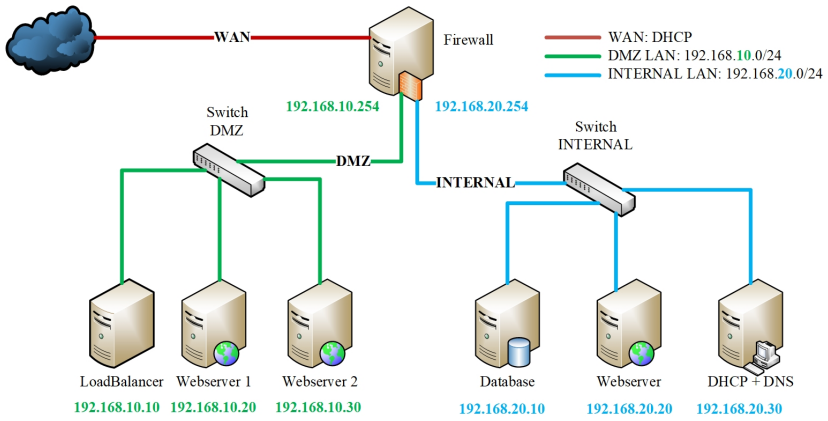



Figure 14: We will create this setup

- 7 VMs
- Two networks: internal and DMZ network for public services
- A firewall
- NAT-ing

## 7.5 Basic networking

To create the setup in the previous section, we need to recap some basic networking:

### 7.5.1 TCP/IP network model


Figure 15: The 5 layers of the TCP/IP model

1. Physical layer: our VMware virtual network
2. Datalink layer: ethernet & MAC-address
3. Network layer: IPv4, IP-address
4. Transport layer:
    - Transport Control Protocol (TCP)
    - User Datagram Protocol (UDP)
5. Application layer: SSH, HTTP, . . .

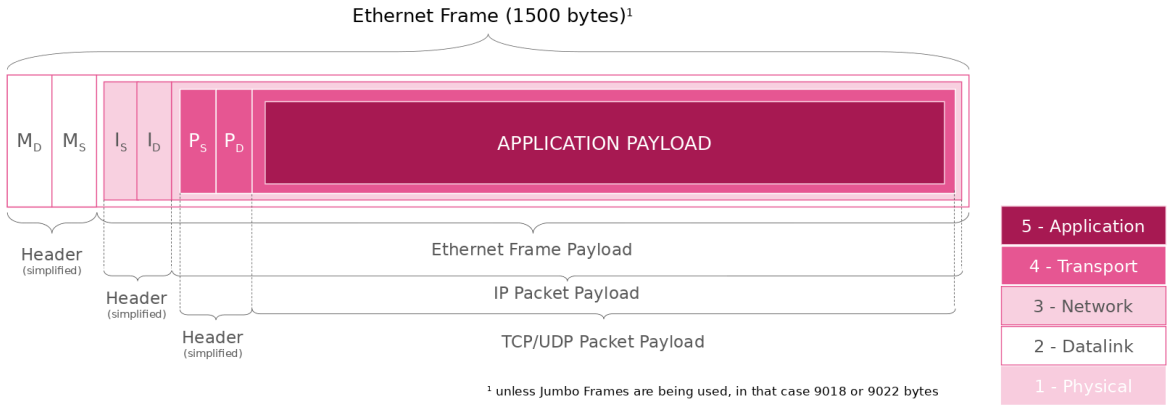



Figure 16: An ethernet frame

### 7.5.2 3 types of firewalls

1. Application Layer Gateway
    - Can see layer 7
    - Can see the application payload
2. Stateful Packet Inspection Firewall

- Stateful = it can keep status, it has memory
- Can't see the application payload

3. Packet Filtering Firewall
   - Simplest type of firewall
   - Can only see information up to Layer 4 (Transport layer)
     - Source & Destination MAC-address
     - Source & Destination IP-address
     - Source & Destination Port number
     - The interface where the packet travels through
   - Can't see the application payload

# 8 Netfilter, iptables, DHCP and DNS

## 8.1 Intermezzo: kernel space vs user space

System memory can be divided in 2 parts: kernel space and user space

### 8.1.1 Kernel space

- Kernel space is that part of memory in which kernel processes are running
- Kernel space memory can not be swapped or deallocated: it is fixed

### 8.1.2 User space

- AKA userland
- That part of the memory where user mode applications are running
- Can be swapped out when needed

## 8.2 Intermezzo: Routing

- Firewall without ruleset = router
- Router with filtering = firewall

Routing = layer 3 (network)

| Destination | Gateway | Interface / link |
|---|---|---|
| 192.168.**Y**.0/24 | 192.168.**X**.254 | ens34 |

Figure 17: What a route looks like (netstat -rn)

Routing = road signs of the network

- Town name = network
- Road sign = route

- Street = interface/link

Router = connected with multiple networks

- Possibly other routers and network further down
- Routing table determines the path to follow
- Create firewall:
  - First create routes
  - Then add rules

## 8.3 Firewalls

### 8.3.1 3 types firewall

- Application Layer Gateway
- Stateful Packet Inspection
- Packet filtering

### 8.3.2 Stateful Packet Inspection Firewall (SPI)

= Packet filter + inspection of state of connection

- Invented / introduced by Check Point in 1994 in FireWall-1
- Packet filter: *stateless*
- Every packet evaluated individually against ruleset
- A lot of packets belong together => make it stateful
- Extra advantage: drop packets which do not belong in the flow

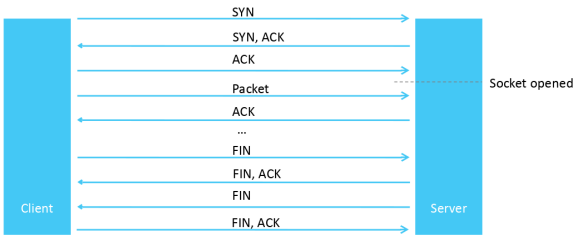



Figure 18

- 3 way handshake between client and server
- Socket opens
- Packets + ACKs
- . . .
- 4 way close

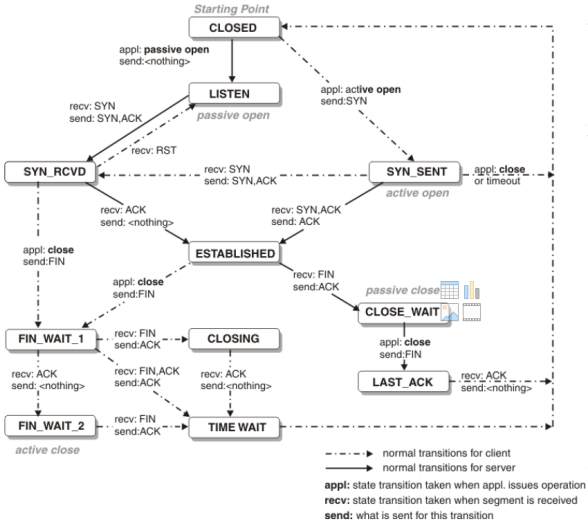



Figure 19: Flowchart of the states

Type of packet does not fit in one of the currently existing flows? ⇒ DROP!

| Source IP | Source Port | Destination IP | Destination Port | Action |
|---|---|---|---|---|
| 194.78.39.18 | any | 2.37.246.12 | 80 | Allow |

Figure 20: Firewall rule

Packet arrives with

- sIP = 194.78.39.18
- sPrt = 20233
- dIP = 2.37.246.12
- dPrt = 80
- Flags = P (push)

⇒ Packet Filter firewall ⇒ ACCEPT

⇒ SPI and no SYN, SYN/ACK, ACK seen before? ⇒ DROP

### 8.3.3 Stateful Packet Inspection vs Packet Filter

Pro:

- More secure
- Evaluate less individual packets (less CPU usage)

Cons:

- Requires more memory (table with state of connections)
  - Decades ago this was a problem because firewalls would only have several MBs of RAM
  - Not a problem anymore because of servers with a lot of RAM

### 8.3.4 SPI in Linux: Connection tracking (conntrack)

The connection tracking system stores information about the state of a connection in a memory structure that contains the source and destination IP addresses, port number pairs, protocol types, state, and timeout.

- conntrack: part of the kernel (it's a kernel module)
- conntrack-tools: user space tools to interact with the in-kernel connection tracking system
- SPI uses the conntrack kernel module, these days it is loaded by default

### 8.3.5 Application Layer Gateway (ALG)

- Can filter based on contents in Layer 5: the Application layer
- Has knowledge of application protocols
- Example:
  - Active FTP
    - Active FTP uses a Control Connection (21/TCP) and a Data Connection (20/TCP), for example:
    - Client: 53621 $\Rightarrow$ Server: 21 (Control Connection: list, get, ...)
    - Server: 20 $\Rightarrow$ Client: 53622 (Data Connection: actual transfer)
  - Passive FTP
  - HTTP: what website may be visited and what websites may not?

Problem: more and more traffic gets encrypted:

- HTTPS, SMTP + SSL, SSH, ...
- One cannot 'look' inside encrypted traffic
- Solution: SSL-inspection
- Purpose: SSL (client $\Rightarrow$ server)
- Problem:
  - SSL1: client makes SSL connection to Firewall
  - SSL2: Firewall makes second SSL connection to server
  - The certificates for the two SSL connections are different
  - Man In The Middle attacks on the Firewall device are possible without the client or server knowing
  - Solution: end-to-end (E2E) encryption

### 8.3.6 Purpose of a firewall

- Selectively allow network traffic
- Network traffic which is welcome, may enter or may pass
- Traffic that is not welcome, is banned

- Firewall uses rules, traffic is evaluated against those rules

## 8.4 Firewall in Linux: iptables

- iptables = package with userspace tools
- Interacts with netfilter (package filtering framework) in kernel space

### 8.4.1 Netfilter

Networking stack = in kernel space

- Driver of network interface card (NIC)
- Ethernet
- CSMA/CD
- Flow-Control
- IEEE802.11 (VLAN)
- ...

These processes in kernel space are provided with a couple API hooks

### 8.4.2 Hooks

**Definitie 8.1** *A hook is a functionality provided by software for users of that software to have their own code called under certain circumstances.*

*Specific actions can be 'hung' on these 'hooks'*

*https://en.wikipedia.org/wiki/Hooking*

Specifically for netfilter hooks:

- Code in the Linux kernel which can catch a specific event of a packet in the network stack

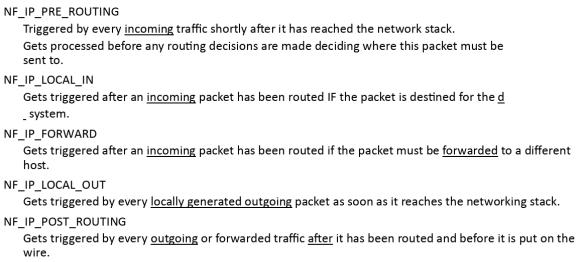



Figure 21: 5 hooks in the kernel networking stack

Example: NF_IP_LOCAL_IN:

- Using iptables a 'rule' can be created specifying which packets may be delivered to a service in this OS, and which can't.
- iptables uses these netfilter hooks in the kernel
- Packet via hook from kernelspace (networking stack, netfilter hook) to userspace (iptables)

- iptables decides what has to happen to the packet and passes decision via the hook back to the kernel

### 8.4.3 iptables - tables

iptables uses tables to order firewall rules

5 built-in tables:

- **Filter table**
  - Most commonly used
  - To decide if a packet is allowed to its desired destination or not
- **NAT table**
  - To do Network Address Translation
  - Source NAT and/or destination NAT
- Mangle table
  - To alter the headers of IP-packets in transit
  - Adjust TTL, TOS, put a label/mark, . . .
- Raw table
  - If you do not want Stateful Packet Inspection
  - Directly evaluate packages instead of using states, do everything manually
- Security table
  - To put SELinux security context marks on packets

### 8.4.4 iptables - chains

Within each table rules are arranged in chains

- These chains represent the netfilter hooks
- Chains determine **when** a rule will be evaluated
- 5 chains in iptables:
  1. PREROUTING: triggered by the NF_IP_PRE_ROUTING hook
  2. INPUT: triggered by the NF_IP_LOCAL_IN hook
  3. FORWARD: triggered by the NF_IP_FORWARD hook
  4. OUTPUT: triggered by the NF_IP_LOCAL_OUT hook
  5. POSTROUTING: triggered by the NF_IP_POST_ROUTING hook
- Chains allow to decide where in the path of ‘packet-delivery’ a firewall has to evaluate a rule

There are 5 built-in tables and 5 available chains. Is every chain available in every table?

- No: security table is of no use in PREROUTING or POSTROUTING

| Tables↓ / Chains → | PREROUTING | INPUT | FORWARD | OUTPUT | POSTROUTING |
|---|---|---|---|---|---|
| raw | ✓ | | | ✓ | |
| mangle | ✓ | ✓ | ✓ | ✓ | ✓ |
| Nat (DNAT) | ✓ | | | ✓ | |
| filter | | ✓ | ✓ | ✓ | |
| security | | ✓ | ✓ | ✓ | |
| nat (SNAT) | | ✓ | | | ✓ |

Figure 22: Tables & Chains

- DNAT = destination NAT = Destination of IP packet gets translated
- Has to be done before packet gets routed ⇒ PREROUTING
- The translated packet is basically a new packet which was crafted by this system and which has to leave this system ⇒ OUTPUT

### 8.4.5 Order of passing through chains

Assumption: routing is correct and firewall allows packet

- Incoming packets destined for local system: PREROUTING ⇒ INPUT
- Incoming packets destined for another system: PREROUTING ⇒ FORWARD ⇒ POSTROUTING
- Outgoing locally created packets: OUTPUT ⇒ POSTROUTING

### 8.4.6 iptables rules

Rules are places in a certain chain of a certain table

- The packet gets matched with each rule from top to bottom
- Each rule has a ‘matching’ component and an ‘action’ component
  - If packet matches with ‘this’, do ‘that’

### 8.4.7 Targets

The ‘action’ component of a rule is the ‘target’

2 categories of targets:

- Terminating targets
  - Do an action that ends all further evaluation and gives control back to the netfilter hook
  - Decision is passed on to hook: DROP or ALLOW
- Non-terminating targets
  - Do an action and continue with the evaluation within the chain
  - Eventually a chain must reach a terminating target and give control back to hook

### 8.4.8 iptables and connection tracking (SPI)

A connection can have a certain state

- **NEW**

- **ESTABLISHED**
- **RELATED**
- INVALID
- UNTRACKED
- SNAT
- DNAT

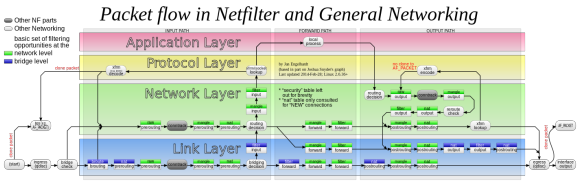



Figure 23: `Bron:https://en.wikipedia.org/wiki/Iptables`

## 8.5 iptables in practice

### 8.5.1 Matches and Targets

A rule is a combination of a ‘match’ and a ‘target’

**Matches**

- A condition that must be met for iptables to process the package
- Some examples:
  - -s (–source): specifies the source of the packet
  - -d (–destination): specifies the destination of the packet
  - -p (–protocol): the protocol (e.g. tcp) which has to be matched
  - -i (–in-interface): specifies the NIC where the packet arrives
  - -o (–out-interface): specifies the NIC where the packet leaves
  - ! stands for negation

**Targets**

- Determine the action which has to be taken when a packet is matched
- **ACCEPT:** allow the packet without further checks
- **DROP:** refuse the packet without sending an answer (without letting the sender know)
- QUEUE: pass the packet on to userspace
- RETURN: give the packet to the next rule in the former chain
- LOG: write a log entry when this rule is matched and continue
- **REJECT:** refuse the packet and let the sender know it was refused

### 8.5.2 Inspect rule, create rules

-A (–append): append a rule at the bottom of the ruleset -D (–delete): remove a rule -L (–list): show a list of all rules

```
~# iptables -L -nv --line-numbers

# Give a list of rules
# Don't do name resolving (-n)
# Do it verbosely (v)
# Show line numbers
```

### 8.5.3 Configure firewall: best practices

- Log in through the console
- Throw away all existing rules (flush)
- Put the default policy on DROP
- Allow management through the local network
- Start 'punching holes'

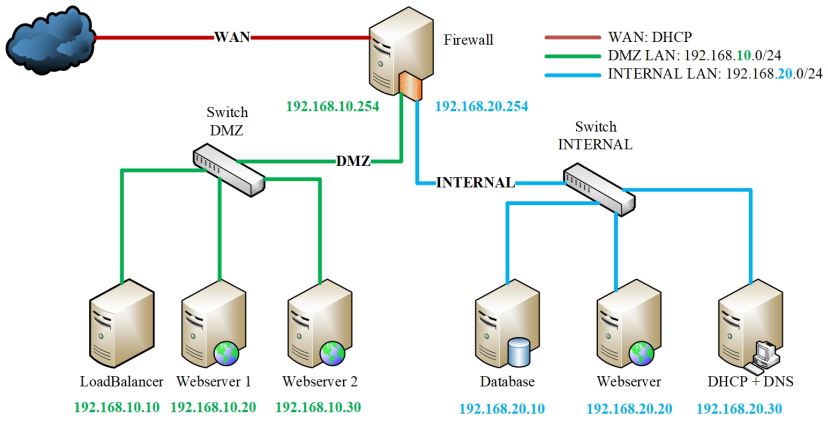
```
# throw away all rules
~# iptables -F
```

```
# Default policy DROP
# by default we don't want to allow anything
~# iptables -P INPUT DROP
~# iptables -P FORWARD DROP
~# iptables -P OUTPUT DROP
```

```
# Allow management through SSH from internal LAN
# Traffic destined for this system => INPUT chain

iptables -A INPUT -s 192.168.X.0/24
        -d 192.168.X.254 -i ens34 -p tcp -m tcp --dport 22 -j ACCEPT
```

# 9 Setting up our firewall in a full-fledged environment

In this chapter, we will set up a firewall in this environment:




## 9.1 Installation: the necessary steps

1. Forwarding: WAN / LAN / DMZ
   - /proc/sys/net/ip_forward (temporary, doesn't survive reboot)
   - /etc/sysctl.conf (persistent)
2. Network interface config
   - /etc/network/interfaces
   - Configure all 3 interfaces

```
# The primary network interface
auto ens33
allow-hotplug ens33
iface ens33 inet dhcp

#extra network interface
auto ens37
allow-hotplug ens37
iface ens37 inet static
        address 192.168.200.254
        netmask 255.255.255.0
        network 192.168.200.0
        broadcast 192.168.200.255
```

Figure 24: Example

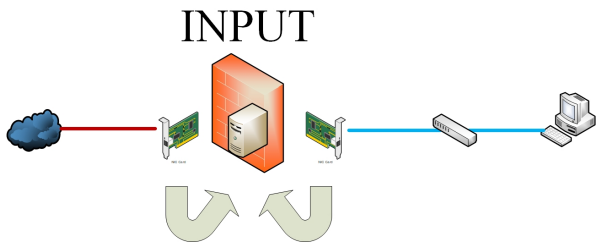
3. Check interface IP configuration
4. Check route information
   - The firewall has 1 default gateway (NAT internet interface / DHCP)
   - The 2 internal networks are directly connected
   - Check if all are reachable with the use of **ping**

## 9.2 iptables chains

- Standard table = filter table
- 3 most important chains: INPUT, OUTPUT, FORWARD

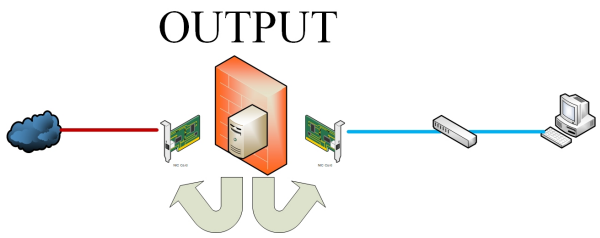
### 9.2.1 INPUT chain

= Traffic from host to firewall itself (to the processes that are running on that machine).



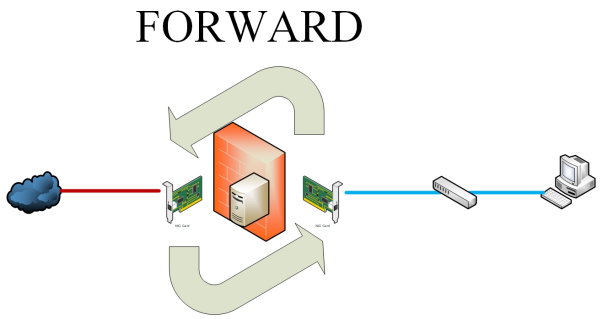

### 9.2.2 OUTPUT chain

= Traffic from firewall to host




### 9.2.3 FORWARD chain

= Traffic passing through firewall from one NIC to another, or from one subnet to another.




## 9.3 Common iptables commands

### 9.3.1 Clear tables

Flush all rules:

```
1 iptables -F
```

### 9.3.2 Close firewall

By default iptables is completely open

```
# close tables
iptables -P INPUT DROP
iptables -P FORWARD DROP

# keep output open
iptables -P OUTPUT ACCEPT

# why accept?
# --> to allow traffic from firewall itself to other host
# security issue? only if firewall gets compromised
```

## 9.4 Allow SSH

- TCP traffic (port 22)
- Traffic from host INTO server

```
iptables -A INPUT -p TCP --dport 22 -s 192.168.20.0/24 -j ACCEPT
# source = 192.168.20.0/24
# accept
# we do not specify the table, so the default is used: FILTER table
```

- Avoid connection from WAN! (to protect against brute-force attacks)
- IMPROVE: limit to correct incoming interface
  - for example: only allow the ens33 interface
- SSH = needed for daily maintenance of server
- What about reply?
  - No problem, because OUTPUT == ACCEPT

## 9.5 Allow TCP and UDP INTO firewall

Our current situation: our firewall blocks all incoming traffic except SSH from the 192.168.20.0/24 subnet.

- Only if traffic was started FROM firewall itself
- $\Rightarrow$ only allow replies, never new connections

```
iptables -A INPUT -p TCP -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -p UDP -m state --state RELATED,ESTABLISHED -j ACCEPT
```

## 9.6 Allow PING or ICMP

Why ping?

- 2 way communication
  - PING comes in (request)
  - Firewall replies (reply)

```
# Allow incoming ICMP connection if a host pings firewall (REQUEST)
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Allow incoming reply if firewall pinged some host (REPLY)
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

Reply or request from firewall to host:

- Already OK, because OUTPUT == ACCEPT

## 9.7 Internet connectivity

### 9.7.1 Who?

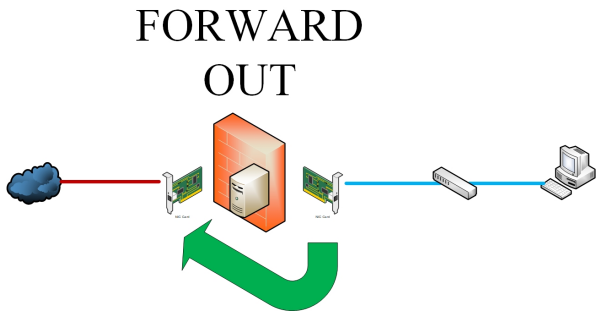
- For internal network through firewall to oustide (NAT)
- For DMZ network through firewall to outside (NAT)
- For internal network through firewall to DMZ network (no NAT, routing)
- One subnet to other subnet ⇒ one NIC to another NIC
- SOURCE NAT = from private local net NAT to elsewhere (public)
- Chain = POSTROUTING (when leaving firewall)

### 9.7.2 How?

We need to configure three things:

1. Allow traffic from internal NIC to WAN NIC ⇒ FORWARD chain
2. Allow traffic from DMZ NIC to WAN NIC ⇒ FORWARD chain
3. Configure an iptable rule for NAT

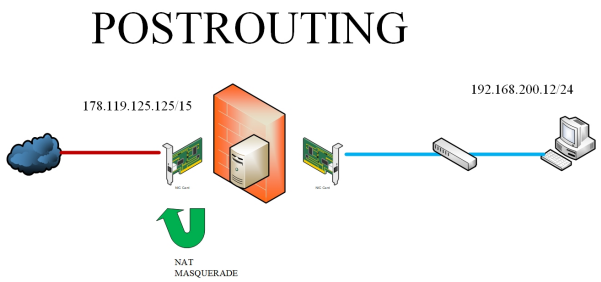
```
# syntax == sudo iptables -A FORWARD -i <in-interface> -o <out-interface> -j ACCEPT
# example:
iptables -A FORWARD -i ens37 -o ens33 -j ACCEPT

# or: it always goes out on WAN NIC:
iptables -A FORWARD -o ens33 -j ACCEPT
```




- iptable rule for NAT

- Not in FILTER table, but in our NAT table
- POSTROUTING chain

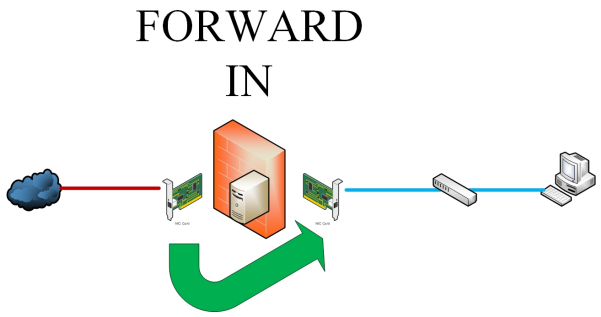
```
# syntax == sudo iptables -t nat -A <chain> -o <out-interface> -j MASQUERADE]
# example:
iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
# -j MASQUERADE will MASQ your internal IP by using external IP
```




### 9.7.3 Allow returning traffic

Traffic that returns and is for internal LAN or for DMZ

```
# allow through firewall, only if related or established traffic
iptables -A FORWARD -i ens33 -m state --state RELATED,ESTABLISHED -j ACCEPT
```



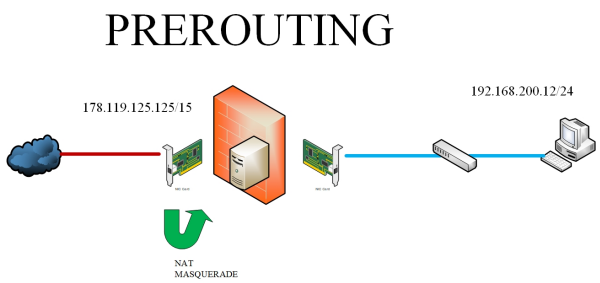

## 9.8 Webserver

Traffic from internet to WAN interface, with destination a webserver in the DMZ network

### 9.8.1 PREROUTING

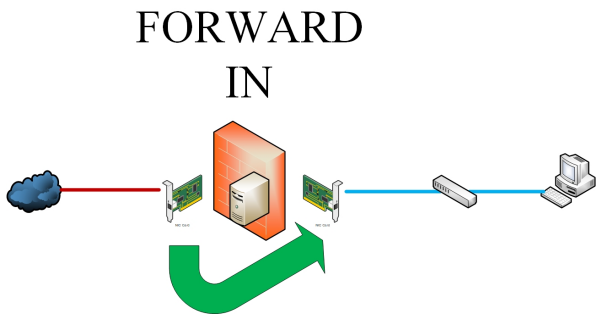
- We need to catch WAN traffic on WAN interface ens33
- TCP port 80
- Prepare it for transport to internal network
  - Change public IP to private (internal) IP
  - = PORT FORWARDING

```
# tell the PREROUTING chain: route all incoming traffic for the WAN interface,
# TCP port 80 to an internal ip and port
iptables -t NAT -A PREROUTING -i ens33 -p tcp --dport 80 -j DNAT
    --to-destination 192.168.10.10:80
```




### 9.8.2 WAN traffic to internal webserver

- Problem: now firewall blocks this
  - Is a 'new' connection, not a reply
  - It only allows related and established connections
- Solution:
  - Open TCP port 80




## 9.9 Don't forget to save!

- All iptables changes are in memory
- Changes disappear on next reboot

```
# command to save the iptables changes
# !! needs the iptables-persistent and netfilter-persistent packages !!
sudo netfilter-persistent save
```