

Linux OS

Tuur Vanhoutte

April 20, 2021

Contents

1	Introduction	1
1.1	Difference Server & Workstation	1
1.1.1	Server	1
1.1.2	Workstation	1
1.2	Extra information/resources	1
1.3	What is Linux?	1
1.3.1	What is an operating system (OS)?	1
1.3.2	What is a Kernel?	2
1.4	GNU Operating System	2
1.5	Linux, the kernel	2
1.5.1	Distributions	2
1.6	Open Source	3
1.6.1	Commercial distributions	3
1.6.2	In this course: Debian	3
2	Debian Installation	4
2.1	Networking in Linux (with VMWare)	4
2.2	Users in Linux	4
2.3	Disks, partition, filesystems	4
2.3.1	Partitions	5
2.4	Partitioning schemes	5
2.4.1	MBR	5
2.4.2	GPT	6
2.4.3	Bootstrap procedure	6
2.4.4	Linux boot process	7
2.4.5	BIOS <> UEFI	7
2.5	Filesystems	7
2.5.1	Windows	7
2.5.2	Linux	7
2.5.3	Swap	8
2.6	File structure	8
2.7	Configuration	9
2.7.1	Packages	9
2.7.2	Package management	9
2.7.3	Useful packages	10
2.8	Shutdown of VM	10
2.9	Basic network	10
2.9.1	Basic networking commands	11
2.10	Services	11
2.11	Wooclap Questions	11
3	File structure	12
3.1	Intermezzo: single user mode	12
3.1.1	Runlevels	13
3.2	Intermezzo: Add disk	13
3.2.1	What after a reboot?	14
3.3	Navigate through the tree	14
3.3.1	Relative vs absolute path	14
3.4	Filesystem Hierarchy Standard (FHS)	14
3.4.1	Rules in the standard	15

3.5	Some useful tips	17
3.5.1	History	17
3.5.2	Bind mount	17
3.5.3	dd	17
3.6	Wooclap Questions	17
4	Filesystems	18
4.1	Introduction	18
4.2	Blocks	18
4.3	ext2/3/4	19
4.3.1	Journaling	19
4.4	RAID	19
4.4.1	RAID Controller	19
4.4.2	RAID 0	19
4.4.3	RAID 1	20
4.4.4	RAID 4	20
4.4.5	RAID 5	21
4.4.6	RAID 6	21
4.4.7	Disk failure	21
4.4.8	Compound RAID levels	22
4.5	OpenZFS	22
4.5.1	ZFS	22
4.5.2	Open ZFS	22
4.6	Intermezzo: Kernel modules	22
4.6.1	Commands	23
4.7	Intermezzo: Snapshots	23
4.7.1	Do we still need backups if we have snapshots?	23
5	File manipulation	23
5.1	Basics	23
5.2	Bundle files	24
5.3	Links and inodes	24
5.3.1	Inodes	24
5.3.2	Symbolic links	25
5.3.3	Hardlinks	25
5.4	File permissions	26
5.5	Overview of basic commands	27
5.6	Wooclap	27
6	Text editors, Piping, Redirection & Jobs	28
6.1	Text editors	28
6.1.1	vi vs vi-improved	28
6.1.2	First steps in vim	29
6.1.3	Search and replace	29
6.1.4	Basic editing tricks	30
6.2	Piping	30
6.3	Redirection	31
6.3.1	stdout and stderr	31
6.3.2	stdin	31
6.4	Jobs and process Management	32
6.4.1	Exit codes	32
6.4.2	Combining commands	33

6.4.3	Jobs	33
6.4.4	Inter-process Communication (IPC)	33
6.5	Intermezzo: System Load	34
6.6	Some useful tips	34
6.6.1	With which unique IP-addresses are there open sockets and how many?	34
6.6.2	TTY	35
6.7	Answer these questions to test your knowledge	35
7	Regex, users & firewall	35
7.1	Regex	35
7.2	User management	36
7.2.1	Add a user	36
7.2.2	Delete a user	36
7.2.3	Change password of user	36
7.2.4	Create a new group	36
7.2.5	Assign a user to a group	37
7.2.6	Overview of users	37
7.2.7	Overview of groups	37
7.2.8	/etc/shadow	37
7.2.9	/etc/gshadow	37
7.2.10	sudo	38
7.2.11	Temporarily become another user	38
7.3	SSH	38
7.3.1	SSH features	39
7.3.2	SCP - Secure Copy	39
7.3.3	SSH tunnels	39
7.4	Full-fledged environment	39
7.5	Basic networking	40
7.5.1	TCP/IP network model	40
7.5.2	3 types of firewalls	41
8	Netfilter, iptables	41
8.1	Intermezzo: kernel space vs user space	41
8.1.1	Kernel space	41
8.1.2	User space	42
8.2	Intermezzo: Routing	42
8.3	Firewalls	42
8.3.1	3 types firewall	42
8.3.2	Stateful Packet Inspection Firewall (SPI)	42
8.3.3	Stateful Packet Inspection vs Packet Filter	44
8.3.4	SPI in Linux: Connection tracking (conntrack)	44
8.3.5	Application Layer Gateway (ALG)	44
8.3.6	Purpose of a firewall	45
8.4	Firewall in Linux: iptables	45
8.4.1	Netfilter	45
8.4.2	Hooks	45
8.4.3	iptables - tables	46
8.4.4	iptables - chains	47
8.4.5	Order of passing through chains	47
8.4.6	iptables rules	47
8.4.7	Targets	48
8.4.8	iptables and connection tracking (SPI)	48

8.5	iptables in practice	48
8.5.1	Matches and Targets	48
8.5.2	Inspect rule, create rules	49
8.5.3	Configure firewall: best practices	49
9	Setting up our firewall in a full-fledged environment	50
9.1	Installation: the necessary steps	50
9.2	iptables chains	51
9.2.1	INPUT chain	51
9.2.2	OUTPUT chain	51
9.2.3	FORWARD chain	51
9.3	Common iptables commands	52
9.3.1	Clear tables	52
9.3.2	Close firewall	52
9.4	Allow SSH	52
9.5	Allow TCP and UDP INTO firewall	52
9.6	Allow PING or ICMP	53
9.7	Internet connectivity	53
9.7.1	Who?	53
9.7.2	How?	53
9.7.3	Allow returning traffic	54
9.8	Webserver	55
9.8.1	PREROUTING	55
9.8.2	WAN traffic to internal webserver	55
9.9	Don't forget to save!	56
10	DHCP	56
10.1	Installing a DHCP server	56
10.1.1	Configuration file	56
10.1.2	Parameters to configure	56
10.1.3	Example configuration	57
10.1.4	Dynamic vs static assignments	57
10.2	Intermezzo: high availability	57
10.2.1	Active-Passive clustering	57
10.2.2	Split-brain	58
10.2.3	STONITH	58
10.2.4	Highly available DHCP server	58
10.3	DNS Server	58
10.3.1	Properties of DNS	58
10.3.2	Structure of a DNS packet	59
10.3.3	DNS Records	59
10.4	DNS in Debian GNU/Linux	59
10.4.1	DNS Client	59
10.4.2	DNS Server	60
10.5	3 useful tips	62
10.5.1	Asynchronous routing and SPI	62
10.5.2	Make brute force on your SSH-daemon harder	62
10.5.3	DansGuardian	62
11	Webserver & Loadbalancer	62
11.1	Apache	62
11.1.1	History	63

11.1.2	Apache modules	63
11.1.3	File structure	64
11.1.4	Multiple websites on 1 server: possible?	64
11.1.5	Basic Authentication	64
11.2	Intermezzo: Loadbalancing algorithms	64
11.2.1	Round-robin (RR)	64
11.2.2	Least connections	65
11.2.3	Source	65
11.2.4	Levels of loadbalancing	66
11.3	HAProxy	66
11.3.1	Access Control Lists (ACLs)	67
11.3.2	Backend	67
11.3.3	Frontend	67
11.3.4	Stickyness	68
11.3.5	Health Checks	68
11.3.6	High availability vs Load balancing	68
11.3.7	Installation haproxy	68
11.3.8	Configuration haproxy	68
11.3.9	Start haproxy	69
11.3.10	Check the operation of the loadbalancer	69
11.3.11	General parameters	69
11.3.12	Health checks: parameters	70
11.4	3 tips	70
11.4.1	Limit the number of simultaneous connections per host	70
11.4.2	Transparent proxy with iptables and squid	70
11.4.3	Conntrack -L	71

1 Introduction

1.1 Difference Server & Workstation

1.1.1 Server

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

1.1.2 Workstation

- Use services
- Create documents
- Look for information
- Consume multimedia
- GUI
- ⇒ 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)?

Definition 1.1 (Operating System) *An operating system, or OS, is software that communicates with the hardware and allows 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?

Definition 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

Definition 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
- Quickly gained 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

⇒ **GNU/Linux**

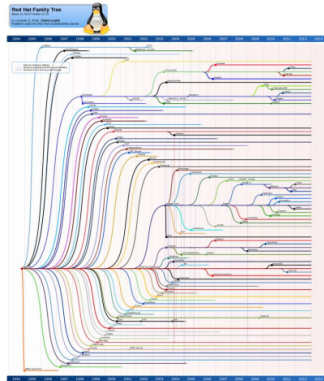
1.5.1 Distributions

Definition 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

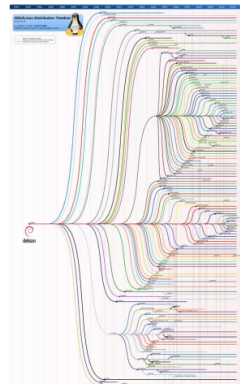


Figure 1: https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

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

1.6 Open Source

Definition 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. The root user's home directory is /root/

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

Definition 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
 - $2^{32} \cdot 512 \text{ bytes} = 4294967296 \cdot 512 \text{ bytes} \approx 2\text{TB}$
- Maximum amount of primary partitions = 4

BIOS can boot from a disk with MBR partitioning

2.4.2 GPT

Definition 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

Definition 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
- Sector = group of blocks together (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 MBR
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** - loads MBR
- **MBR** - loads GRUB
- **GRUB** - loads kernel
- **Kernel** - loads modules and 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
- ⇒ we learn MBR

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

Definition 2.4 *Memory paging or 'Swap' is a memory management scheme by which a computer stores and receives data from a disk to use in RAM. The OS retrieves data from disk in blocks called 'pages'. Paging lets programs exceed the size of available physical memory.*

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

2.6 File structure

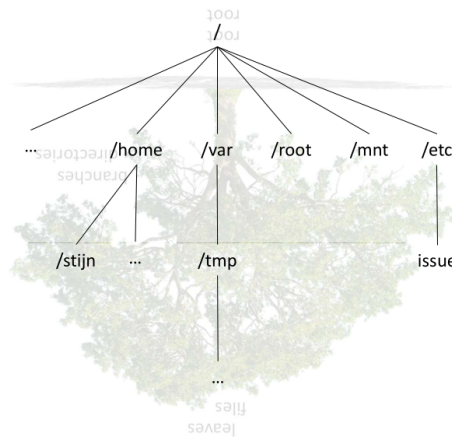


Figure 3: Linux uses a tree structure

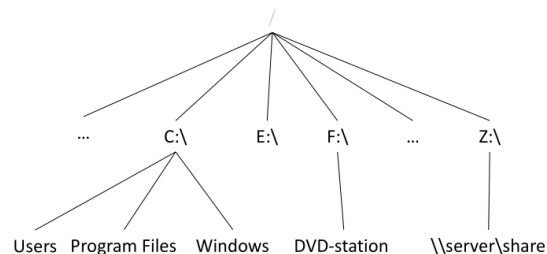


Figure 4: Windows uses a similar structure, but every volume uses a letter.

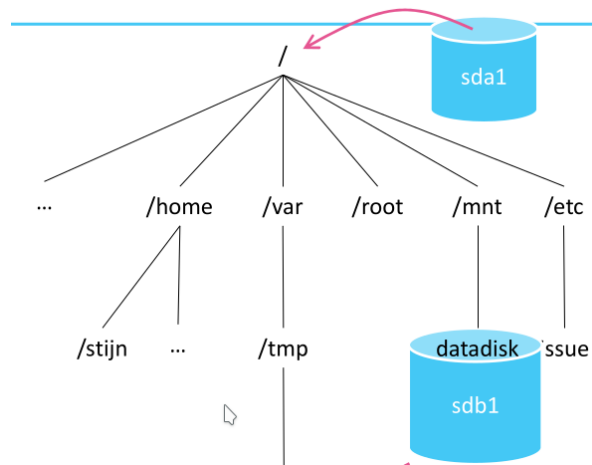


Figure 5: With linux, volumes are 'mounted' to folders somewhere under root /

2.7 Configuration

2.7.1 Packages

- Tools and applications are built up by files
- All files belonging to 1 application are bundled in a package
- Packages in debian have the .deb extension
- Packages are collected in repositories
- Packages have dependencies (= other packages the application depends on to run or install)

Definition 2.5 *Software repositories (repos) are storage locations for software packages and their metadata, which includes the list of dependencies of the package. The repositories are managed and updated by package managers.*

2.7.2 Package management

Debian uses dpkg (Debian Package) & 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 packages in one command:

```
1 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 ⇒
- 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: the ss command:

1

```
ss -tulpn
```

- t: show TCP
- u: show UDP
- l: show listening
- p: show process ID
- n: no name-resolving

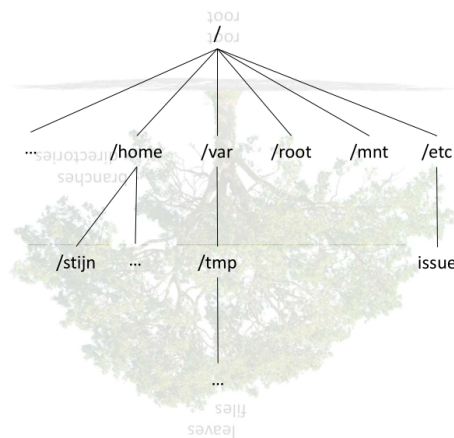
2.11 Wooclap Questions

- 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

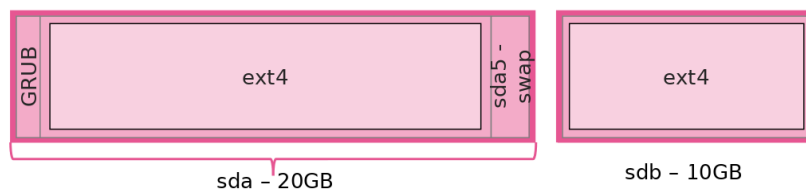
= predefined operating system status

- 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 \Rightarrow 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 (identified by `/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)

Definition 3.1 *The Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions.*

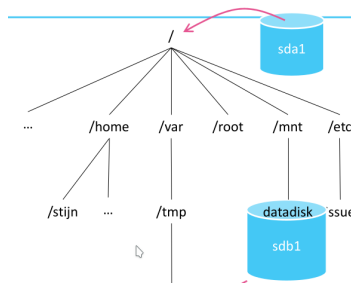
https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

- 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
 - * /dev/sda
 - * /dev/rtc
 - * /dev/random
 - * /dev/cpu
 - * /dev/urandom
 - * /dev/null
 - ls -lah /dev/
- /etc
 - Host-specific system-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 \Rightarrow 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

```
1 ~# history
2
3 # shows a list of former commands executed by this user
4 # spans log-in sessions
5 # in reality, it shows the contents of the ~/.bash_history file
6 # 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

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

Definition 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

The **extended filesystem** or **ext**, was implemented in 1992 as the first filesystem created specifically for the Linux kernel. Ext3 and Ext4 are journaling filesystems.

4.3.1 Journaling

Definition 4.2 *A journaling filesystem is a filesystem that keeps track of changes not yet committed to the disk by recording the goal of these changes in a logbook-like data structure known as a 'journal'.*

In the event of a system crash or power failure, these filesystems can be brought back online faster with a lower likelihood of becoming corrupted.

4.4 RAID

Definition 4.3 (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:

Definition 4.4 (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*

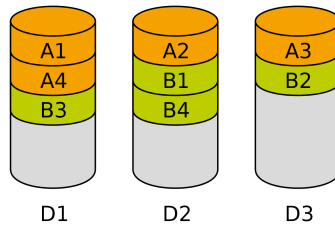


Figure 6: Example: files A and B (4 blocks each) are spread over disks D1-D3

4.4.3 RAID 1

RAID level 1 uses mirroring:

Definition 4.5 (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*

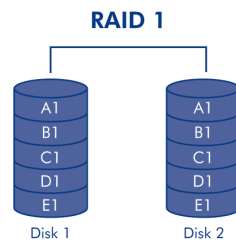


Figure 7: RAID 1

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

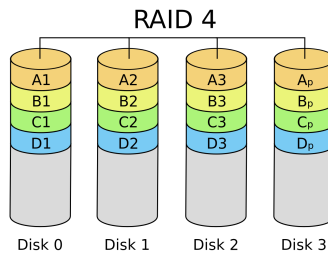


Figure 8: A RAID-4 setup with 4 disks. Disk 3 is the parity disk

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

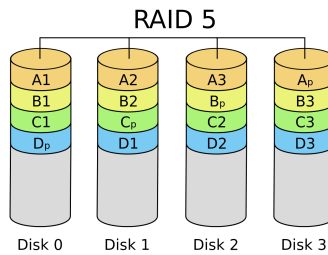
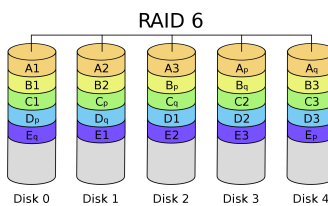


Figure 9: RAID-5: distributed parity with 4 disks

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 losing data:

- 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
- ...

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

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

```
1 # Request current list of modules:
2 ~# lsmod
3
4 # Load module:
5 ~# modprobe brtfs
6
7 # Remove module ("unload"):
8 ~# rmmod brtfs
```

4.7 Intermezzo: Snapshots

- Literally: a photograph of your filesystem
- Captures 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?
- ⇒ backups can be stored safely, on other disks

5 File manipulation

5.1 Basics

```
1 # create an empty file called 'test'
2 ~$ touch test
3
4 # edit a file
5 ~$ vim test
6
7 # remove file
8 ~$ rm rabbot
9
10 # move the file to /tmp
11 ~$ mv test /tmp/
12
13 # rename the file
```

```

14 ~$ mv test rabbit
15
16 # Linux doesn't really look at file extensions
17 # check the file extension:
18 ~$ file <filename>
19 ~$ file /boot/inird.img-4.9.0-13-amdb64
20 ~$ 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

```

1 ~$ mkdir bundle
2 ~$ cd bundle
3 ~$ touch 1.txt 2.txt 3.txt
4 ~$ tar cfv bundle.tar *.txt
5 # c = create a new archive
6 # f = specify a filename (bundle.tar)
7 # v = verbose: show what happens
8 ~$ tar --list -f bundle.tar
9 ~$ file bundle.tar
10
11 # extracting
12 ~$ tar zxvf bundle.tar.gz
13 # z = zipped (compressed)
14 # x = eXtract
15 # v = verbose
16 # f = the argument (a file)

```

5.3 Links and inodes

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

5.3.1 Inodes

Definition 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

Definition 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'.

```

1 ~$ ln -s <target> <link-name>
2 ~$ ln -s /etc/issue test-link
3 # try out the following commands after creating a link:
4 ~$ cat test-link
5 ~$ file test-link
6 ~$ cat /etc/issue

```

5.3.3 Hardlinks

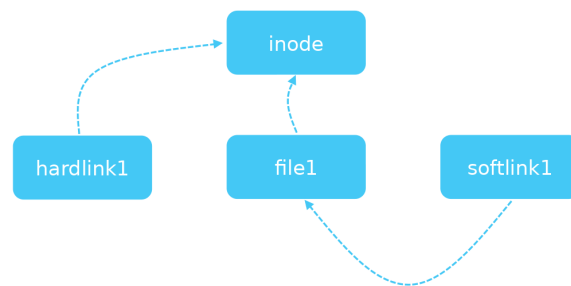


Figure 10: Symlink vs Hardlink

- file1 refers to the same file, but with a different name
- A hardlink refers to an inode, while a softlink refers to a file (which refers to an inode)

5.4 File permissions

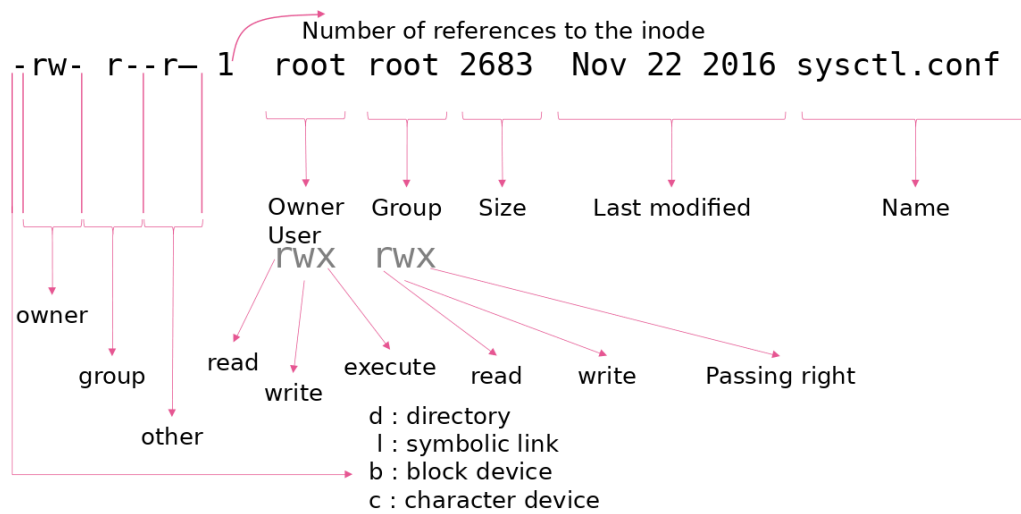


Figure 11: The output of 'ls -l' creates this type of output

From left to right:

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

```

1  # change the owner and group of a file or directory
2  ~# chown <user>:<group> <file>
3  ~# chown root:staff file.txt
4
5  # change the rights for a file
6  # 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

Popular Linux 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 (which stands for left, down, up, right)
- Navigating in vim: HJKL or arrow keys

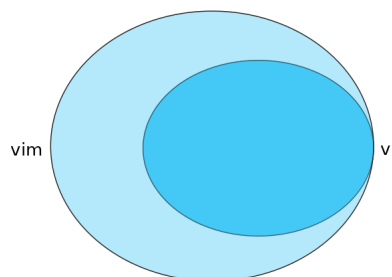


Figure 13: Everything you can do in vi, you can do in vim, and more!

6.1.2 First steps in vim

```
1 # start vim:
2 $~ vim
3
4 # start vim tutorial:
5 $~ 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

```
1 # copy a line of text
2 ESC
3 # put cursor on the line you want to copy
4 yy # yank yank: copy a line of text
5 p # put: paste the copied line
6
7 # copy 2 lines of text and paste it 8 times:
8 ESC 2yy # 2 yank yank: yank 2 lines
9 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:

```
1 # search and replace the next instance:
2 :s/old/new
3
4 # all instances in current line
5 :s/old/new/g
6
7 # all instances between line 10 and 20:
8 :10,20s/old/new/g
9
10 # all instances in a complete file:
11 :%s/old/new/g
12
13 # all instances in whole file, with confirmation:
14 :%s/old/new/gc
15
16 #undo:
17 (ESC) u
```

6.1.4 Basic editing tricks

```
1  # starting from line 4, indent the next 7 lines:
2  (ESC) 4gg
3  7>>
4
5  # remove indentation on line 10
6  (ESC) 10gg
7  <<
8
9  # delete line 3
10 (ESC) 3gg
11 dd
12
13 # delete the next 4 lines
14 (ESC) 4dd
15
16 # enable and disable syntax highlighting in vim
17 (ESC) : syntax on
18 (ESC) : syntax off
```

For more tricks: see labs

6.2 Piping

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

```
1  # sort the music file alphabetically
2  # and count the number of occurrences of each unique line
3  sort music.txt | uniq -c
4
5  # count the number of unique lines in music.txt
6  sort music.txt | uniq | wc -l
7
8  # count the number of lines in /etc/locale.gen where nl or NL occurs
9  grep -i nl /etc/locale.gen | wc -l
10
11 # count the number of lines in /var/log/syslog where kernel occurs
12 cat /var/log/syslog | grep kernel | wc -l
13
14 # count the number of lines in /var/log/syslog where kernel does NOT occur
15 cat /var/log/syslog | grep -v kernel | wc -l
16
17 # Show of what days there are logs in /var/log/syslog
18 # the sixth field is the day field:
19 cat /var/log/syslog | cut -c 6 | uniq
20
21 # show the different sources of log entries in /var/log/syslog
22 # kernel, client, systemd, ...
23 cat /var/log/syslog | cut -d' ' -f5 | cut -d'[' -f1 | sort | uniq -c
```

6.3 Redirection

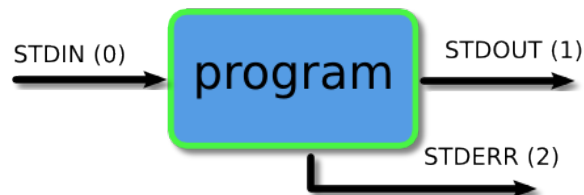
Do not send the output of a command to stdout, but to another location, like a textfile

```
1 # to overwrite a file (or create if it doesn't exist)
2 ls -la > listing.txt
3
4 # to append to a file (or create if it doesn't exist)
5 ls -la >> listing.txt
6
7 # these two commands have the same result
8 cat > textfile.txt
9 touch textfile.txt
10
11 # redirect the output of 'ls -la' to a file with a custom name:
12 # example: output_2021-03-03
13 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



```
1 # redirect the output of a command to out.txt
2 # and redirect the error of the command to error.txt:
3 ls -la > out.txt 2> error.txt
4
5 # redirect stderr to stdout (&1), and then redirect stdout to a file out.txt:
6 ls -la > out.txt 2>&1
7
8 # redirect both to a file:
9 ls -la &> out
```

6.3.2 stdin

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

```

3
4 # this command does the same
5 wc -l < music.txt
6
7 # this command does the same, but prints the output of wc to out.txt
8 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
 - ⇒ 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

```

1 # report a snapshot of current processes
2 ps
3
4 # display a tree of processes
5 pstree -p

```

6.4.1 Exit codes

```

1 # when a process stops:
2 # if bash was parent => exit code is passed to bash
3 # exit code is available in the $? variable:
4 # read contents of a variable with echo:
5 echo $?
6 0
7
8 # 0 = ended successfully without errors
9
10 # 1 = not ended successfully, there were errors

```

```

1 # test = verify if you have rights on a file (with -r = read rights)
2 ~$ test -r music.txt
3
4 ~$ echo $?
5 0
6
7 ~$ test -r doesnotexist
8
9 # the file doesn't exist, so it exited with code 1:
10 ~$ echo $?
11 1

```

6.4.2 Combining commands

= not the same as piping!

```
1 # note the difference between one and two ampersands:
2 # & = execute command 1, then execute command 2
3 # && = execute command 1, and only execute command 2
4 #   if command 1 was successful (exit code = 0)
5
6 # one ampersand
7 test -r test.txt & echo "MCT rocks"
8
9 # two ampersands
10 ~# test -r test.txt && echo "MCT rocks"
11
12 ~# test -r doesnotexist & echo "MCT rocks"
13
14 # the first command will exit with code 1
15 # so the second command will not execute
16 ~# test -r doesnotexist && echo "MCT rocks"
```

6.4.3 Jobs

Definition 6.1 A job is a new process originating from the same parent

```
1 # start a command as job
2 tail -f /var/log/syslog &
3 # output appears on stdout, but process runs as a job in the background
4
5 # bring a job to the foreground
6 fg <index>
7
8 # stop the job, but do not terminate:
9 CTRL-Z
```

6.4.4 Inter-process Communication (IPC)

Definition 6.2 Inter-process communication (IPC) refers to the mechanisms an OS provides to allow processes to manage shared data. Signals are one of those mechanisms.

Definition 6.3 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 \Rightarrow OS interrupts normal execution of that process to deliver the signal

```
1 # Sending a signal to a process: with the `kill' command:
2 # not only to kill a process, also to send other signals
3 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

```

1  # send signal 15 to a process with the entered PID
2  kill -s 15 <pid>
3
4  # send signal 15 to the PID of the tail process
5  kill -s 15 'pidof tail'
6
7  # send signal 9 to a process with the entered PID
8  kill -s 9 <pid>
9
10 # kill the process with name 'tail'
11 pkill tail

```

6.5 Intermezzo: System Load

Definition 6.4 *In Unix, the system load is a measure of the amount of computational work that a computer system performs.*

The load average represents the average system load over a period of time. It conventionally appears in the form of three numbers which represent the system load during the last one, five and fifteen minutes.

- Completely idle system: system load 0
- Each process which uses a resource or is waiting for a resource increases the system load
- System load is a snapshot, doesn't say anything
 - System load of 17: is that a problem? No.
 - More interesting: the evolution of the system load over time \Rightarrow load average

```

1  # show the system load of the last minute, last 5 minutes and last 15 minutes:
2  uptime
3
4  # show who is logged on and what they are doing
5  w
6
7  # display linux processes
8  top
9  # or better:
10 htop

```

6.6 Some useful tips

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

```

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

```


6.6.2 TTY

= Tele TYpewriter = a terminal which is connected with stdin

```
1 # print the filename of the terminal currently connected to standard input:
2 ~$ 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

Definition 7.1 A *Regular Expression (regex)* is a sequence of characters that specifies a search pattern. These patterns can be used in many tools like *sed*, *vim*, *grep*, *python*, ... to search and/or replace characters.

Use www.regexr.com to learn and test regex.

7.2 User management

7.2.1 Add a user

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

```
1 adduser testuser1
2 # enter password
3 # name, other details
4 # /home/testures1
5 # contents of this directory follows /etc/skel
6 # group: testuser1
7 # default shell = /bin/bash
8
9 useradd testuser2
10 # user is created , no password, no shell
```

7.2.2 Delete a user

Again with root privileges, and 2 possible commands:

```
1 deluser testuser1
2 # user and corresponding group get deleted
3 # home directory not deleted
4
5 userdel testuser2
6 # user and corresponding group get deleted
7 # home directory also deleted
```

7.2.3 Change password of user

```
1 # change your own password
2 passwd
3 # enter current password
4 # enter new password x2
5
6 # change password of other account (root privilege required)
7 passwd <username>
8 # current password not needed, only new password x2
```

7.2.4 Create a new group

Root privilege required!

```
1 addgroup testgroup1
2 # GID = group ID
3
4 groupadd testgroup2
```

7.2.5 Assign a user to a group

```
1 usermod -aG testgroup1 testuser1
2 # add to group, groupname, username
3
4 # overview of groups and group members
5 cat /etc/group
6
7 # of which groups is the current user a member?
8 ~# groups
9 ~# su - testuser1
10 ~$ groups
```

7.2.6 Overview of users

/etc/passwd contains a row for every user, with:

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

7.2.7 Overview of groups

/etc/group contains a row for every group, with:

- 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
 - Most common: sha512-hash

7.2.9 /etc/gshadow

Contains a row for each group, with:

- the group's encrypted password, as well as group membership and administrator information
- the group name
- the hashed group 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
- a list of group administrators: users who can add or remove group members using the gpasswd command
- a list of group members

7.2.10 sudo

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

```

1  # decide who gets sudo rights:
2
3  # install the sudo package
4  apt install sudo
5  # change the rights
6  visudo
7  # view the changes
8  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

```

1  sudo su - <username>
2  # this switches the user to <username> and use the environment of that user
3  # eg. the $PATH
4
5  # become root:
6  ~$ sudo su -
7  # 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

```

1 ssh -L <local-IP>:<local-port>:<remote-ip>:<remote-port> username@ssh-server
2
3 # SOCKS-proxy functionality:
4 ssh -D <local-port> username@ssh_server

```

7.4 Full-fledged environment

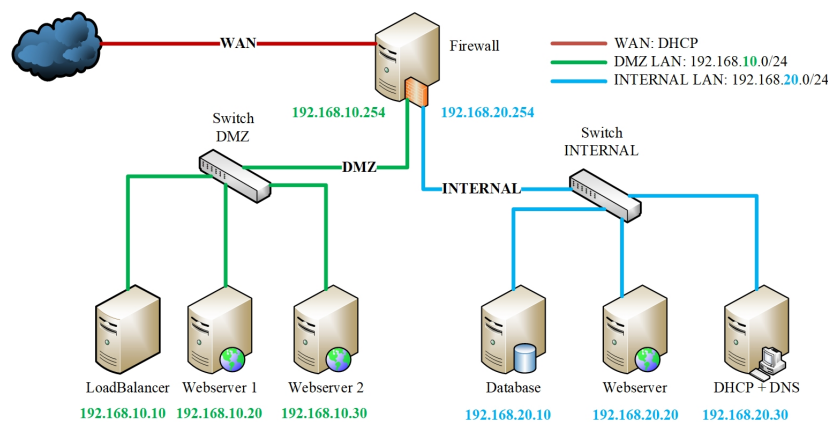


Figure 14: We will create this setup

- 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

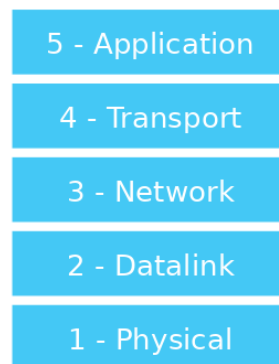


Figure 15: The 5 layers of the TCP/IP model

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, ...

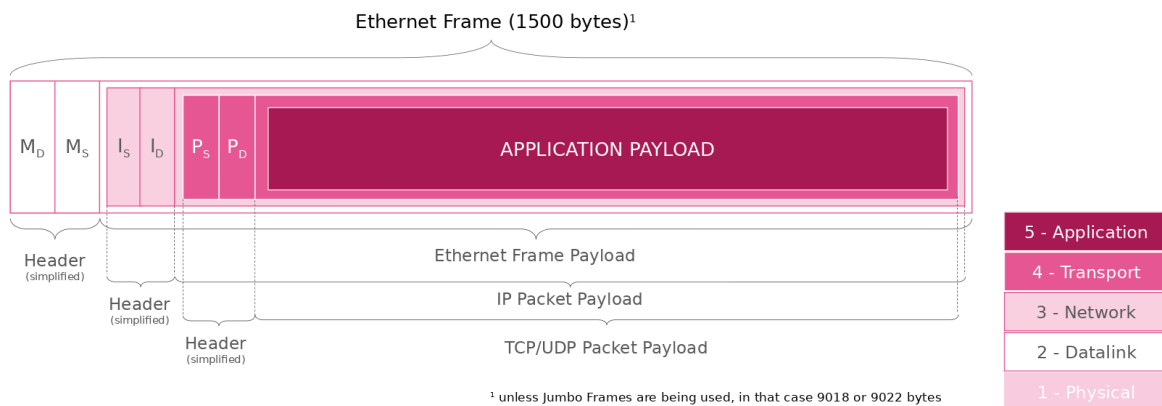


Figure 16: An ethernet frame

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

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

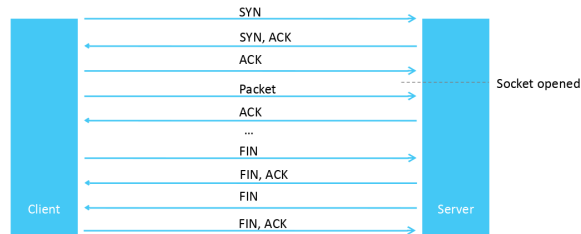


Figure 18

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

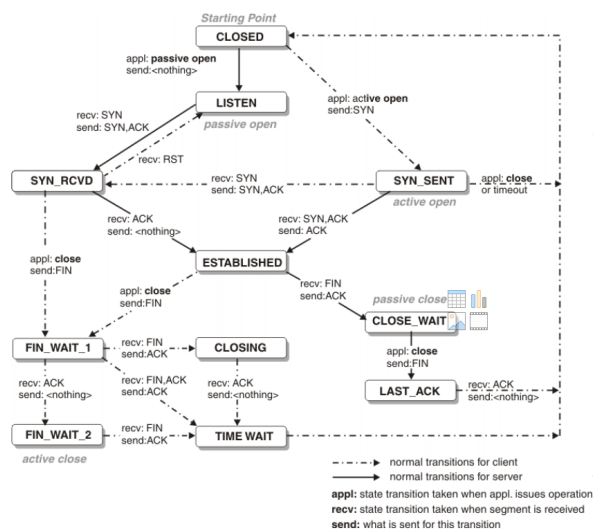


Figure 19: Flowchart of the states

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
- examples:
 - Active FTP
 - * Active FTP uses a Control Connection (21/TCP) and a Data Connection (20/TCP), for example:
 - * Client: 53621 ⇒ Server: 21 (Control Connection: list, get, ...)
 - * Server: 20 ⇒ 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
- 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

8.4.1 Netfilter

Definition 8.1 *Netfilter is a framework provided by the Linux kernel that offers various functions and operations for packet filtering, network address translation (NAT) and port translation.*

*This framework resides in kernel space, and is controllable by userspace applications like **iptables**.*

The entire network stack resides 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

Definition 8.2 *A hook is 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

```

NF_IP_PRE_ROUTING
    Triggered by every incoming traffic shortly after it has reached the network stack.
    Gets processed before any routing decisions are made deciding where this packet must be
    sent to.
NF_IP_LOCAL_IN
    Gets triggered after an incoming packet has been routed IF the packet is destined for the d
    _system.
NF_IP_FORWARD
    Gets triggered after an incoming packet has been routed if the packet must be forwarded to a different
    host.
NF_IP_LOCAL_OUT
    Gets triggered by every locally generated outgoing packet as soon as it reaches the networking stack.
NF_IP_POST_ROUTING
    Gets triggered by every outgoing or forwarded traffic after it has been routed and before it is put on the
    wire.

```

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 and their 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

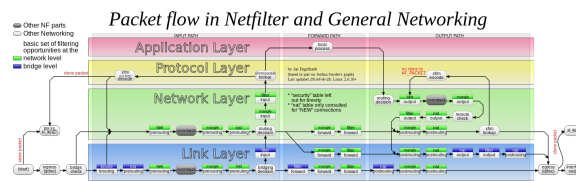


Figure 23: Bron:<https://en.wikipedia.org/wiki/Iptables>

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

```

1 ~# iptables -L -nv --line-numbers
2
3 # Give a list of rules
4 # Don't do name resolving (-n)
5 # Do it verbosely (v)
6 # 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'

```

1 # throw away all rules
2 ~# iptables -F

```

```

1 # Default policy DROP
2 # by default we don't want to allow anything
3 ~# iptables -P INPUT DROP
4 ~# iptables -P FORWARD DROP
5 ~# iptables -P OUTPUT DROP

```

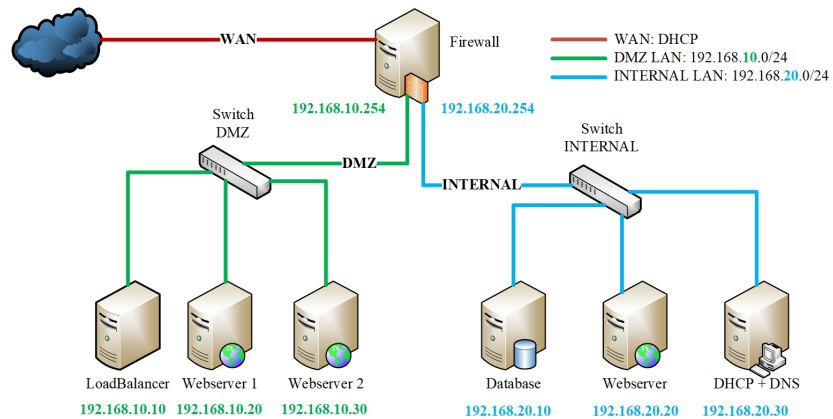
```

1 # Allow management through SSH from internal LAN
2 # Traffic destined for this system => INPUT chain
3
4 iptables -A INPUT -s 192.168.X.0/24
5         -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/ipv4_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

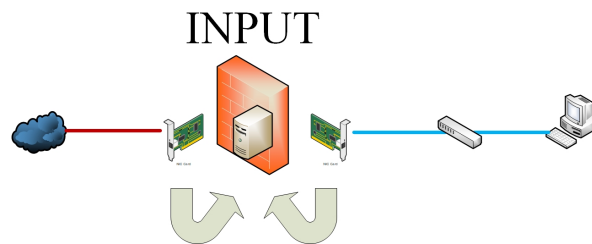
- The firewall has 1 default gateway (NAT internet interface / DHCP)
- The 2 internal networks are directly connected
- Check if all are reachable with the use of **ping**

9.2 iptables chains

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

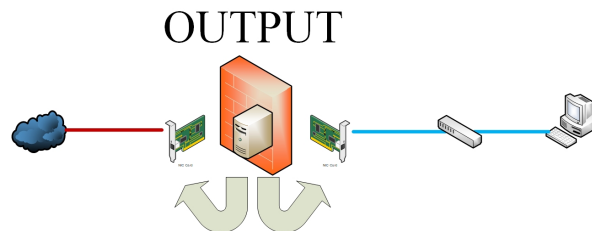
9.2.1 INPUT chain

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



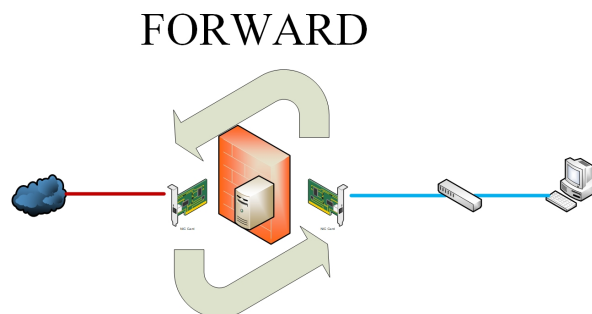
9.2.2 OUTPUT chain

= 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

```
1 # close tables
2 iptables -P INPUT DROP
3 iptables -P FORWARD DROP
4
5 # keep output open
6 iptables -P OUTPUT ACCEPT
7
8 # why accept?
9 # --> to allow traffic from firewall itself to other host
10 # security issue? only if firewall gets compromised
```

9.4 Allow SSH

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

```
1 iptables -A INPUT -p TCP --dport 22 -s 192.168.20.0/24 -j ACCEPT
2 # source = 192.168.20.0/24
3 # accept
4 # 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
- ⇒ only allow replies, never new connections

```

1 iptables -A INPUT -p TCP -m state --state RELATED,ESTABLISHED -j ACCEPT
2 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)

```

1 # Allow incoming ICMP connection if a host pings firewall (REQUEST)
2 iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
3
4 # Allow incoming reply if firewall pinged some host (REPLY)
5 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 outside (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:

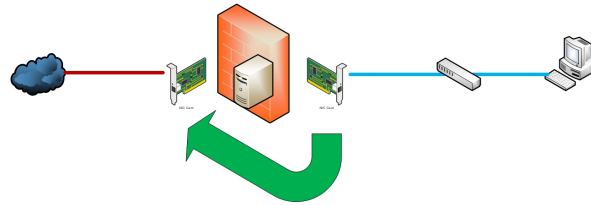
1. Allow traffic from internal NIC to WAN NIC ⇒ FORWARD chain
2. Allow traffic from DMZ NIC to WAN NIC ⇒ FORWARD chain
3. Configure an iptable rule for NAT

```

1 # syntax == sudo iptables -A FORWARD -i <in-interface> -o <out-interface> -j ACCEPT
2 # example:
3 iptables -A FORWARD -i ens37 -o ens33 -j ACCEPT
4
5 # or: it always goes out on WAN NIC:
6 iptables -A FORWARD -o ens33 -j ACCEPT

```

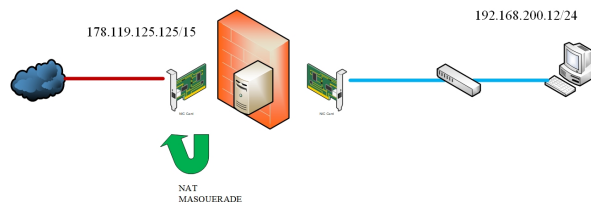
FORWARD OUT



- iptable rule for NAT
- Not in FILTER table, but in our NAT table
- POSTROUTING chain

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

POSTROUTING

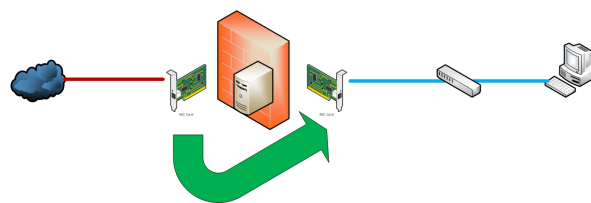


9.7.3 Allow returning traffic

Traffic that returns and is for internal LAN or for DMZ

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

FORWARD IN



9.8 Webserver

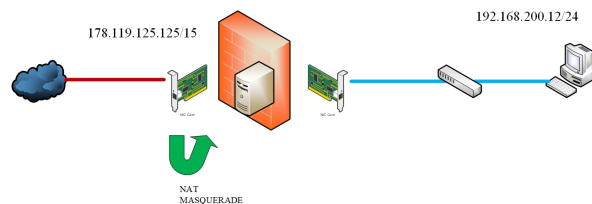
Traffic from internet to WAN interface, with destination a webserver in the DMZ network

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

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

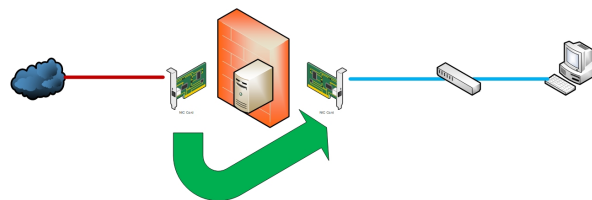
PREROUTING



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

FORWARD IN



9.9 Don't forget to save!

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

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

10 DHCP

- Works with UDP
- Ports 67 and 68
- Broadcast: because the client does not have an IP address yet
- DHCP servers are limited to their own subnet (layer 2)

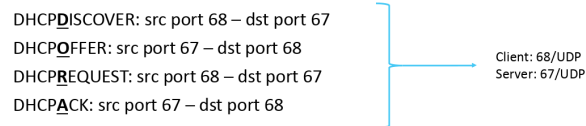


Figure 25: Server = 67, Client = 68

10.1 Installing a DHCP server

- Package = isc-dhcp-server
- ISC = Internet Systems Consortium
- <https://www.isc.org/mission/>

```
1 ~$ sudo apt install isc-dhcp-server
```

10.1.1 Configuration file

- System-wide host-specific configuration file ⇒ somewhere under /etc/
- /etc/default/isc-dhcp-server
- In this file you need to explicitly allow dhcp to run

10.1.2 Parameters to configure

- Listen on which NIC?
- domain-name
- domain-name-servers
- subnet
 - Network address and netmask

- dhcp-range
- default gateway

10.1.3 Example configuration

/etc/default/isc-dhcp-server:

```
1 INTERFACESv4="ens34"
```

/etc/dhcp/dhcpd.conf:

```
1 subnet 192.168.20.0 netmask 255.255.255.0 {
2     range 192.168.20.50 192.168.20.100;
3     option routers 192.168.20.254;
4     option domain-name "mctinternal.be";
5     option domain-name-servers 192.168.20.30;
6 }
```

10.1.4 Dynamic vs static assignments

Dynamic:

- Available IP from the range gets assigned

Static:

- Host with MAC-address X:Y:Z always has to receive IP address a.b.c.d
- Note: a.b.c.d lies outside the range, but inside the subnet!

```
1 host nameOfTheHost {
2     hardware ethernet c8:f7:33:22:c6:4e;
3     fixed-address 192.168.20.200;
4 }
```

10.2 Intermezzo: high availability

- Purpose: increase the availability of a service
- How? Build in redundancy
- Redundancy = overcapacity
- In other words: create overcapacity and ensure that it does not oppose each other

10.2.1 Active-Passive clustering

- One server is active and serves clients
- The other server is passive and only intervenes when the active one fails
- The passive server periodically syncs with the changes from the active server
- Or every transaction is synced to both servers at the same time
- ⇒ 'Referee' required

10.2.2 Split-brain

With most types of clustering, there is a possible problem. Take the following situation:

- Active still works
- Passive still works
- But they don't know it from each other
- \Rightarrow passive thinks active is gone and becomes active
- Problem: 2 active servers \Rightarrow many problems

10.2.3 STONITH

- A way of 'fencing' with clusters
- = Shoot The Other Node In The Head
- <https://en.wikipedia.org/wiki/STONITH>
- You provide a mechanism to 'kill' one node once the other node takes over
- Common fencing mechanisms:
 - Shutting down the UPS to the server
 - Hardware-wise shut down the network

10.2.4 Highly available DHCP server

- 2 DHCP servers in network \Rightarrow Many possible problems:
 - If they have incompatible configurations
 - If they have the exact same configurations

Solution: a High Availability (HA) DHCP server

- With a HA DHCP server, it is possible to 'share' DHCP configuration among multiple servers
- Let the HA DHCP server fix its own problems

Easier solution:

- 2 DHCP servers with the same configuration
- Except the subnet range: make sure the subnet range do not overlap

10.3 DNS Server

10.3.1 Properties of DNS

- 53/UDP (normal use)
- 53/TCP (zone transfer, or as a fallback for UDP)
- Resource Records (RR)
- FQDN, with a trailing dot at the end
 - For example: 'www.example.com.'

– <http://www.dns-sd.org/TrailingDotsInDomainNames.html>

- 2 types of DNS queries:
 - **Iterative queries:** the DNS server will not go and fetch the complete answer for your query but will give back a referral to other DNS server's, which might have the answer.
 - **Recursive queries:** queries in which the DNS server, who received your query, will do all the job of fetching the answer and giving it back to you.

10.3.2 Structure of a DNS packet

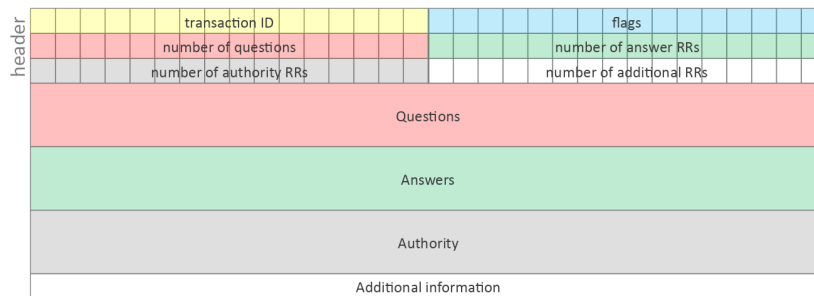


Figure 26: Structure of a DNS packet

10.3.3 DNS Records

- Convert hostnames into IP-address = **A-record**
- Convert IP address into hostname = **PTR-record**
- Which server handles e-mail for this domain = **MX-record**
- Anti-SPAM = **SPF-record** (using a specifically formatted TXT-record)
- Which server handles DNS for the domain = **NS-record**
- Alias of another A-record = **CNAME**

10.4 DNS in Debian GNU/Linux

10.4.1 DNS Client

- Using the 'dig' command, you can query DNS servers
- There are many parameters to filter/change the query:
 - dig ns www.nmct.be ⇒ query only NS-records
 - +norecurse ⇒ don't use recursive queries
 - You can query a specific DNS server: dig @ns1.example.com www.example.com

```

; <<> Dig 9.10.3-P4-Debian <<> www.nmct.be
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 19571
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: 0, MBZ: 0005, udp: 4096
;; QUESTION SECTION:
;www.nmct.be.                IN      A

;; ANSWER SECTION:
www.nmct.be.                5       IN      CNAME   nmct.azurewebsites.net.
nmct.azurewebsites.net. 5       IN      CNAME   waws-prod-db3-087.vip.azurewebsites.windows.net.
waws-prod-db3-087.vip.azurewebsites.windows.net. 5 IN CNAME waws-prod-db3-087.cloudapp.net.
waws-prod-db3-087.cloudapp.net. 5 IN     A      13.74.252.44

;; Query time: 23 msec
;; SERVER: 192.168.254.2#53(192.168.254.2)
;; WHEN: Mon Mar 19 10:34:40 CET 2018
;; MSG SIZE rcvd: 191

```

Figure 27: dig www.nmct.be

```

; <<> Dig 9.10.3-P4-Debian <<> ns www.nmct.be
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 46436
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: 0, MBZ: 0005, udp: 4096
;; QUESTION SECTION:
;www.nmct.be.                IN      NS

;; ANSWER SECTION:
www.nmct.be.                5       IN      CNAME   nmct.azurewebsites.net.
nmct.azurewebsites.net. 5       IN      CNAME   waws-prod-db3-087.vip.azurewebsites.windows.net.
waws-prod-db3-087.vip.azurewebsites.windows.net. 5 IN CNAME waws-prod-db3-087.cloudapp.net.

;; AUTHORITY SECTION:
cloudapp.net.                5       IN      SOA      prd1.azure-dns-cloud.net.
msnhs1.microsoft.com.cloudapp.net. 2113772632 900 300 604800 60

;; Query time: 36 msec
;; SERVER: 192.168.254.2#53(192.168.254.2)
;; WHEN: Mon Mar 19 10:42:38 CET 2018
;; MSG SIZE rcvd: 252

```

Figure 28: dig ns www.nmct.be ⇒ query NS-records

```

; <<> Dig 9.10.3-P4-Debian <<> +norecurse @192.168.20.254 www.mct.be
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: REFUSED, id: 33502
;; flags: qr ad; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; Query time: 3 msec
;; SERVER: 192.168.20.254#53(192.168.20.254)
;; WHEN: Mon Mar 19 10:49:22 CET 2018
;; MSG SIZE rcvd: 12

```

Figure 29: dig +norecurse @192.168.20.254 www.mct.be

```

; <<> Dig 9.10.3-P4-Debian <<> @ns1.howest.be www.howest.be
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 20209
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: 0, udp: 4096
;; QUESTION SECTION:
;www.howest.be.              IN      A

;; ANSWER SECTION:
www.howest.be.              3600    IN      A      185.162.30.243

;; Query time: 12 msec
;; SERVER: 193.191.136.220#53(193.191.136.220)
;; WHEN: Mon Mar 19 10:53:24 CET 2018
;; MSG SIZE rcvd: 58

```

Figure 30: dig @ns1.howest.be www.howest.be ⇒ query a specific DNS server

10.4.2 DNS Server

The standard, most popular DNS server is **Bind**

```

1 # install the necessary bind packages
2 sudo apt install bind9 bind9utils bind9-doc
3
4 # this file holds information on root name servers

```

```

5  # needed to initialize cache of internet domain name servers
6  cat /etc/bind/db.root
7
8  # activate the bind
9  /etc/default/bind9
10     OPTIONS="-4 -u bind"
11
12  # configuration
13  /etc/bind/named.conf.local
14     zone "example.com" {
15         type master;
16         file "/etc/bind/zones/example.com";
17     };
18
19     zone "20.168.192.in-addr.arpa" {
20         type master;
21         file "/etc/bind/zones/reverse/rev.20.168.192";
22     };

```

```

1  # create a zones folder, and a zones/reverse folder
2  sudo mkdir -p /etc/bind/zones/reverse
3
4  # contents of a zone file:
5  /etc/bind/zones/example.com
6
7  ;
8  ; BIND data for example.com
9  ;
10 $TTL 3h
11 @      IN      SOA      ns1.example.com. admin.example.com. (
12                                1          ; SERIAL
13                                3h          ; Refresh
14                                1h          ; Retry
15                                1w          ; Expire
16                                1h )       ; Minimum
17 ;
18 @      IN      NS       ns1.example.com.
19 @      IN      NS       ns2.example.com.
20
21 example.com.      IN      A          192.168.20.30
22 ns1               IN      A          192.168.20.30
23 ns2               IN      A          192.168.20.31
24 www              IN      CNAME      example.com.
25 ftp              IN      CNAME      example.com.

```

```

1  # contents of a reverse zone file:
2  /etc/bind/zones/reverse/rev.20.168.192
3
4  ;
5  ; BIND reverse file for 20.168.192.in-addr.arpa
6  ;
7  $TTL      604800
8  @      IN      SOA      ns1.example.com. admin.example.com. (
9                                  1          ; Serial

```

```

9             3h      ; Refresh
10            1h      ; Retry
11            1w      ; Expire
12            1h )    ; Minimum
13      ;
14      @      IN      NS      ns1.example.com.
15      @      IN      NS      ns2.example.com.
16
17      38      IN      PTR     example.com.

```

10.5 3 useful tips

10.5.1 Asynchronous routing and SPI

Be careful with asynchronous routing and Stateful Packet Inspection

- SYN arrives through side A
- SYN, ACK leaves through side B
- ACK arrives through side A

10.5.2 Make brute force on your SSH-daemon harder

```

1 iptables -A INPUT -s 192.168.20.0/24 -d 192.168.20.201/32
2   -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
3
4 iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22
5   -m state --state NEW -m recent --update --seconds 60 --hitcount 4
6   --name DEFAULT --mask 255.255.255.255 --rsource -j DROP

```

10.5.3 DansGuardian

- Software designed to control which websites users can access
- It also includes virus filtering and usage monitoring features
- <https://en.wikipedia.org/wiki/DansGuardian>

11 Webserver & Loadbalancer

11.1 Apache

Apache is not just a webserver: Apache is a gigantic software project with:

- 28 categories
- 185 individual projects
- One of which is 'HTTP server' (aka httpd)
- <https://www.apache.org/index.html#projects-list>
- <http://httpd.apache.org/>

- Apache has its own license

11.1.1 History

- 1989
 - Tim Berners-Lee writes a proposal for an ‘information management system’
 - Together with Belgian Robert Cailliau
 - HyperText Transfer Protocol (HTTP)
 - Founder of websites on the internet
- Early 1990s
 - World Wide Web is gradually emerging
- 1995: many dissatisfied with the quality of the existing web servers
 - Student projects, no real commercial web servers yet
 - An alliance of programmers decides to create a web server themselves

Because Apache was a gathering of work by many, there were a lot of ‘holes’ in the code

- Many ‘patches’ had to be released to close those ‘holes’
- Hence the name = ‘A patchy’ webserver

11.1.2 Apache modules

Apache has a modular design

- Extra functionality wanted \Rightarrow load module
 - mod_proxy
 - mod_rewrite
 - mod_ssl
 - ...
 - <https://httpd.apache.org/docs/2.4/mod/>

```
1 # get list of currently loaded modules
2 apache2ctl -M
```

Which modules are enabled on your system?

```
1 /etc/apache2/mods-enabled/
2 # this is full of symbolic links
3 # don't change these links, modules are enabled/disabled with commands
4 a2enmod
5 a2dismod
```

11.1.3 File structure

System-wide host-specific configuration files: /etc/apache2/

- Configuration
 - conf-available
 - conf-enabled
- Modules
 - mods-available
 - mods-enabled
- Sites
 - sites-available
 - sites-enabled

DocumentRoot = /var/www/html

11.1.4 Multiple websites on 1 server: possible?

- Using VirtualHosts
- Every VirtualHost has a DocumentRoot
 - = The folder containing the website's files

11.1.5 Basic Authentication

Using 2 files:

- .htaccess
- .htpasswd

11.2 Intermezzo: Loadbalancing algorithms

3 important load-balancing algorithms:

- Round-robin
- Least connections
- Source

11.2.1 Round-robin (RR)

= Alternate between each backend server

Example with 3 servers:

- 1st request goes to server 1
- 2nd request goes to server 2
- 3rd request goes to server 3
- 4th request goes to server 1

- 5th request goes to server 2
- 6th request goes to server 3
- ...

Advantages:

- Proportional distribution
- Very simple implementation

Drawbacks:

- Does not take capacity of backend server into account
- Does not take backend server load into account

Weighted Round-robin

- Solution for 'Does not take capacity of backend server into account'
- Every backend server gets a weight
- Weight represents capacity
- Higher weight = backend server can process more connections
- No solution for 'Does not take backend server load into account'

11.2.2 Least connections

Not all connections are equal:

- Short visits
- Long-term visits
- Visits with little server load
- Visits with a lot of taxes
- ...

Backend server with the least number of active connections receives the new request

- Ensures a more balanced distribution
- Partial solution for 'Does not take backend server load into account'

Weighted Least connections

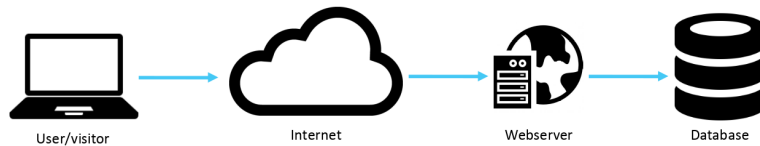
- Every backend server gets a weight
- Weight represents capacity
- Higher weight = backend server can process more connections

11.2.3 Source

- Choice of backend server is determined by hash of source IP
- Ensures that a user always ends up on the same server

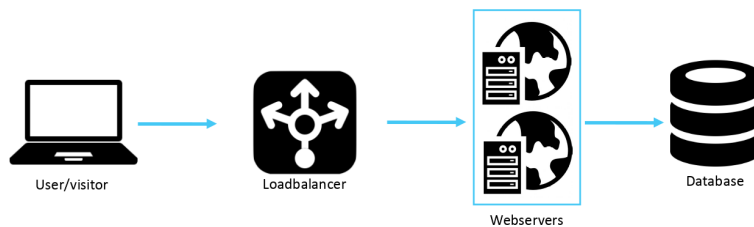
11.2.4 Levels of loadbalancing

- No loadbalancing



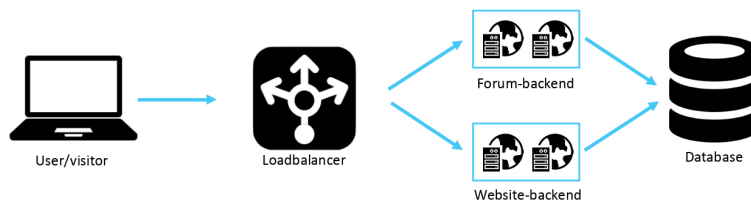
- Layer 4 loadbalancing

- Simplest form of loadbalancing
- Deals with the delivery of messages with no regard to the content
- IP+port combination (Transport layer)



- Layer 7 loadbalancing

- Application layer
- Deals with the actual content of each message (looks at URL, cookies, ...)
- <https://www.nginx.com/resources/glossary/layer-7-load-balancing/>



11.3 HAProxy

HAProxy is free, open-source software that provides a high availability load balancer and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers

Used by:

- GitHub
- Imgur
- Instagram
- Twitter

- ...

Configuration manual: <http://www.haproxy.org/download/2.0/doc/configuration.txt>

11.3.1 Access Control Lists (ACLs)

The ACLs test a certain condition, and link an action to that condition. Examples:

- pattern matching (regex)
- number of connections
- ...

```
1 # create an ACL that matches when the path of a request starts with /blog
2 # Example: http://yourdomain.com/blog/blog-entry-1
3 # the acl's name is url_blog
4 acl url_blog path_beg /blog
```

You can send these actions to a backend

11.3.2 Backend

= Term for 'the underlying web servers'

In our case: 2 web servers with apache httpd:

- 192.168.X.20
- 192.168.X.30

Backend, in its simplest form, consists of:

- Chosen loadbalancing algorithm
- List of web servers and port numbers

Example configuration:

```
1 backend web-backend
2     balance roundrobin
3     server web1 web1.yourdomain.com:80 check
4     server web2 web2.yourdomain.com:80 check
5
6 backend blog-backend
7     balance roundrobin
8     mode http
9     server blog1 blog1.yourdomain.com:80 check
10    server blog1 blog1.yourdomain.com:80 check
```

11.3.3 Frontend

Specifies how requests should be forwarded to backends

- Frontend consists in its simplest form of:
 - Collection of IP addresses and port numbers
 - ACLs

- ‘use_backend’ rules (determines which backend to use, based on ACLs)

11.3.4 Stickyness

Some applications require that a user always connect to the same backend server.

- This is called ‘sticky sessions’
- Parameter appsession in haproxy is used for this

11.3.5 Health Checks

- Haproxy must know which backend server can handle all requests
- Simplest way: try 3-way open to the TCP port of each of the backend servers
- More advanced way: request web page of backend server, calculate hash, compare hash with hash of "OK" page
 - Specific health check page that checks whether the application is still OK

Server is alive ⇒ is still participating in the load balancing

Server is dead? ⇒ is automatically removed from load balancing

11.3.6 High availability vs Load balancing

High Availability

- The goal is to **maximize availability**, usually by setting up different systems that can fulfill the same function

Load Balancing

- The goal is to **distribute the load** between different systems that fulfill the same function

Combination of HA and LB is possible

11.3.7 Installation haproxy

```
1 # install the haproxy package
2 sudo apt install haproxy
3
4 # enable haproxy
5 sudo vim /etc/default/haproxy
6
7 # show haproxy version info
8 sudo haproxy -v
```

11.3.8 Configuration haproxy

```
1 # this file contains configuration for frontend, backend and ACLs
2 sudo vim /etc/haproxy/haproxy.cfg
```

```

1 frontend resume_site_front
2     bind 192.168.X.10:80
3     stats uri /haproxy?stats
4     default_backend resume_site_back
5
6 backend resume_site_back
7     balance roundrobin
8     server webserver1 192.168.X.20:80 check
9     server webserver2 192.168.X.30:80 check

```

11.3.9 Start haproxy

```

1 sudo systemctl restart haproxy

```

Navigate to <http://192.168.X.10/haproxy?stats>

Security issues: do not enable this for remote connections (information disclosure)

11.3.10 Check the operation of the loadbalancer

1. On both web servers:

```

1     # use this command to 'sniff' traffic
2     tcpdump -flni any port 80

```

2. Browse to your public IP

On both web servers, you can see some continuous movement

- But we only requested our webpage once?
- This is because we do a check (see configuration) to check if the servers are alive

11.3.11 General parameters

httplog:

- Enable logging of HTTP request, session state and timers
- By default: the log output format is very poor: it only contains the source and destination addresses and the instance name
- By specifying 'option httplog', each log line turns into a much richer format

httpclose:

- Enable or disable passive HTTP connection closing

abortonclose:

- Enable or disable early dropping of aborted requests pending in queues.

httpchk:

- Enable HTTP protocol to check on the servers health
- Default: layer 4 health check

- 'option httpchk' => layer 7 health check
- When "option httpchk" is specified, a complete HTTP request is sent once the TCP connection is established, and responses 2xx and 3xx are considered valid, while all other ones indicate a server failure, including the lack of any response.

11.3.12 Health checks: parameters

inter:

- Interval (in seconds) between 2 consecutive health checks
- Default: 2 seconds

fastinter:

- Interval (in seconds) between 2 consecutive health checks when state of backend server is 'transitionally up' or 'transitionally down'
- Default: value of inter

downinter:

- Interval when backend server is down
- Default: value of inter

fall:

- Number of consecutive failed health checks before a backend server gets status DOWN
- Default: 3

rise:

- Number of consecutive succesful health checks before a backens server gets status UP
- Default: 2

11.4 3 tips

11.4.1 Limit the number of simultaneous connections per host

```
1 iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 15
2   --connlimit-mask 32 -j REJECT --reject-with tcp-reset
```

11.4.2 Transparent proxy with iptables and squid

Iptables: forward to a squid proxy

```
1 iptables -t nat -A PREROUTING -i ens34 -s 192.168.45.0/24
2   -p tcp --dport 80 -j REDIRECT --to-port 3128
3
4 # Squid listens on 3128/TCP
```

11.4.3 Conntrack -L

Conntrack = userspace tool to interact with kernelspace connection tracking module

```
1 # give a list of current connections being `tracked`  
2 conntrack -L
```