Linux OS

Tuur Vanhoutte

March 3, 2021

Contents

1 Introductie

1.1 Verschil Server & Workstation

1.1.1 Server

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

1.1.2 Workstation

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

1.2 Extra information/resources

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

1.3 What is Linux?

1.3.1 What is an operating system (OS)?

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

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

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

1.3.2 What is a Kernel?

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

It is the first program loaded on start-up.

It handles. . . :

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

1.4 GNU Operating System

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

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

Goal: completely free Operating System

1.5 Linux, the kernel

By Linus Torvalds (Finland), 1991

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

Linux = the kernel

GNU = OS-tools around the kernel

$\Rightarrow$ **GNU/Linux**

1.5.1 Distributions

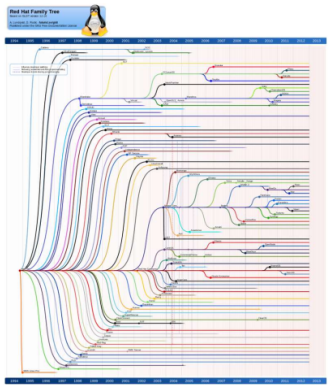
Definitie 1.4 (Distribution) *A Linux distribution (or distro for short) is GNU/Linux + extra tools and applications to create a full-fledged OS.*

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

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

Red Hat family tree

Debian family tree



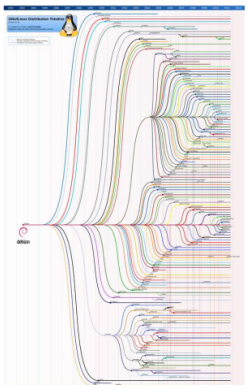


Figure 1: https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

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

1.6 Open Source

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

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

OPEN SOURCE DOES NOT MEAN FREE

1.6.1 Commercial distributions

= Open source, non-free distributions

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

Commercial distributions have official support channels.

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

1.6.2 In this course: Debian

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

2 Debian Installation

See Labs for detailed Installation tutorial

2.1 Networking in Linux (with VMWare)

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

2.2 Users in Linux

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

2.3 Disks, partition, filesystems

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

2.3.1 Partitions

Primary partition

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

Extended Partition

- 'Logical' partitions can be created inside this

Our setup:

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

Figure 2: Our setup

2.4 MBR <> GPT

2.4.1 MBR

We use the MBR Partitioning scheme

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

It contains:

- *a set of instructions necessary to boot operating systems.*
- *info about how partitions are placed on disk*

Limitations:

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

BIOS can boot from a disk with MBR partitioning

2.4.2 GPT

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

It uses unique identifiers (GUIDs)

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

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

How does it work?

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

2.4.3 Bootstrap procedure

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

2.4.4 Linux boot process

6 high level steps

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

2.4.5 BIOS <> UEFI

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

So why will we use MBR?

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

2.5 Filesystems

2.5.1 Windows

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

2.5.2 Linux

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

2.5.3 Swap

= Paging

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

2.6 File structure

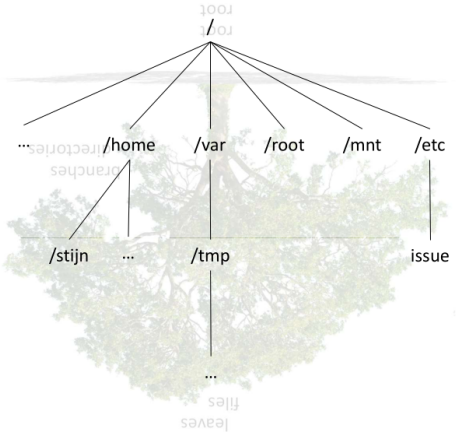


Figure 3: Linux uses a tree structure

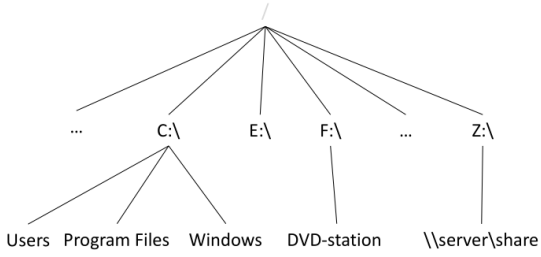


Figure 4: Windows uses a similar structure, but every volume uses a letter.

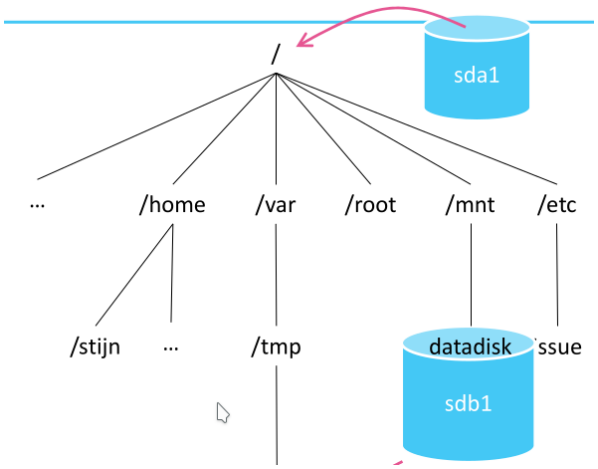


Figure 5: With linux, volumes are 'mounted' to folders somewhere under root /

2.7 Configuration

2.7.1 Packages

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

Repositories

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

2.7.2 Package management

Debian: dpkg & apt (Advanced Package Tool)

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

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

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

2.7.3 Useful packages

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

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

2.8 Shutdown of VM

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

2.9 Basic network

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

2.9.1 Basic networking commands

- arp
- ping
- route
- bmon

2.10 Services

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

2.11 Wooclap Questions

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

- What is a linux distribution?

3 File structure

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

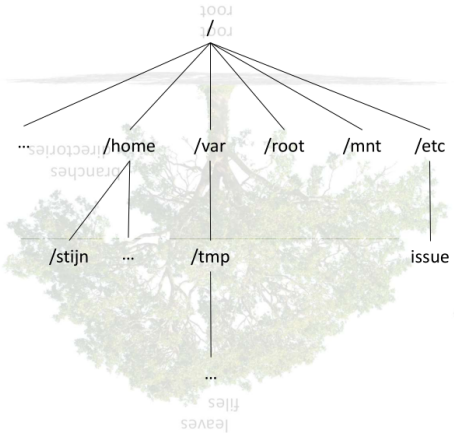


Figure 6

3.1 Intermezzo: single user mode

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

3.1.1 Runlevels

= predefined operating system status

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

3.2 Intermezzo: Add disk

Add a new disk without shutting down the system

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

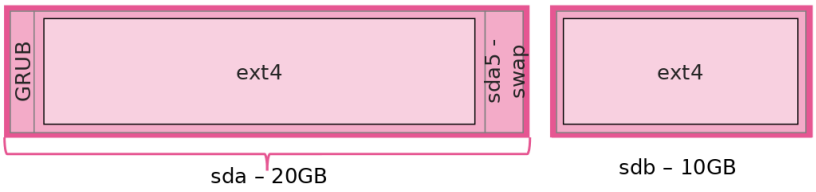


Figure 7

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

For detailed steps: see labs!

3.2.1 What after a reboot?

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

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

3.3 Navigate through the tree

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

3.3.1 Relative vs absolute path

Relative paths:

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

Absolute paths:

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

3.4 Filesystem Hierarchy Standard (FHS)

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

3.4.1 Rules in the standard

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

  - Some distributions use /media for this

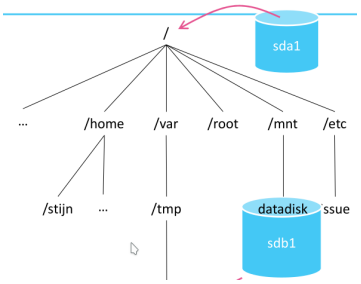


Figure 8

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

3.5 Some useful tips

3.5.1 History

```
~# history

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

CTRL + r:

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

3.5.2 Bind mount

Situation

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

The solution

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

3.5.3 dd

= Command to read or write bytes

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

3.6 Wooclap Questions

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

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

4 Filesystems

4.1 Introduction

Books:

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

Computers:

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

4.2 Blocks

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

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

Several choices:

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

4.3 ext2/3/4

Ext3 and Ext4 have journaling:

4.3.1 Journaling

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

4.4 RAID

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

Many purposes:

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

4.4.1 RAID Controller

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

4.4.2 RAID 0

RAID level 0 uses striping:

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

Purpose:

- *Increasing data throughput*

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

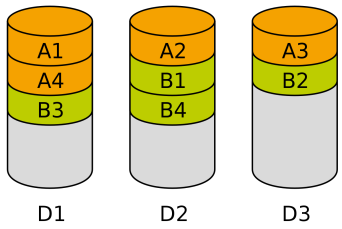


Figure 9: Example: files A and B (4 blocks each) are spread over disks D1-D3

4.4.3 RAID 1

RAID level 1 uses mirroring:

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

Purpose:

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

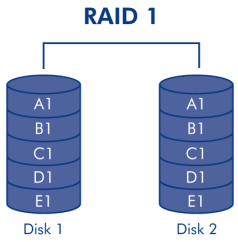


Figure 10: RAID 1

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

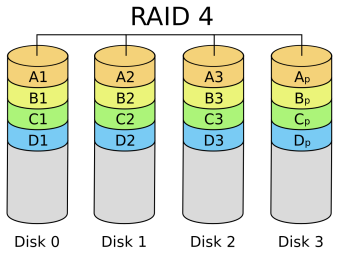


Figure 11: A RAID-4 setup with 4 disks. Disk 3 is the parity disk

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

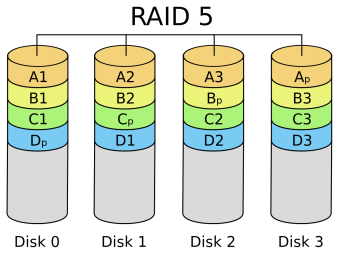


Figure 12: RAID-5: distributed parity with 4 disks

4.4.6 RAID 6

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

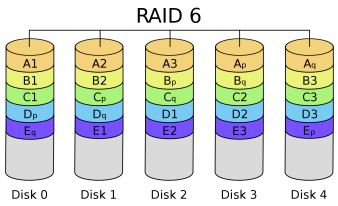


Figure 13

4.4.7 Disk failure

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

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

4.4.8 Compound RAID levels

Combining RAID levels is possible:

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

4.5 OpenZFS

4.5.1 ZFS

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

4.5.2 Open ZFS

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

Features:

- Long term storage
- Checksum of all data and metadata
- Native RAID levels (0, 1, 5, 6, . . .)
- All data gets written through Copy-On-Write
- Snapshots (read-only and mountable)
- Transparent compression
- Huge storage possibilities
- 128 bits system
- up to 256 quadrillion zettabytes

4.6 Intermezzo: Kernel modules

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

4.6.1 Commands

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

# Load module:
~# modprobe brtfs

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

4.7 Intermezzo: Snapshots

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

4.7.1 Do we still need backups if we have snapshots?

YES!

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

5 File manipulation

5.1 Basics

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

# edit a file
~$ vim test

# remove file
~$ rm rabbot

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

# rename the file
~$ mv test rabbit

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

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

5.2 Bundle files

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

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

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

5.3 Links and inodes

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

5.3.1 Inodes

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

Metadata: data about the file

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

5.3.2 Symbolic links

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

Also called 'softlinks'

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

5.3.3 Hardlinks

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

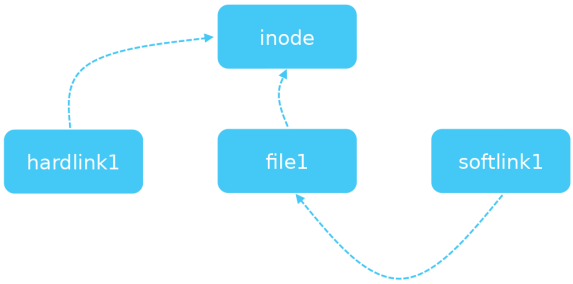


Figure 14: Symlink vs Hardlink

5.4 File permissions

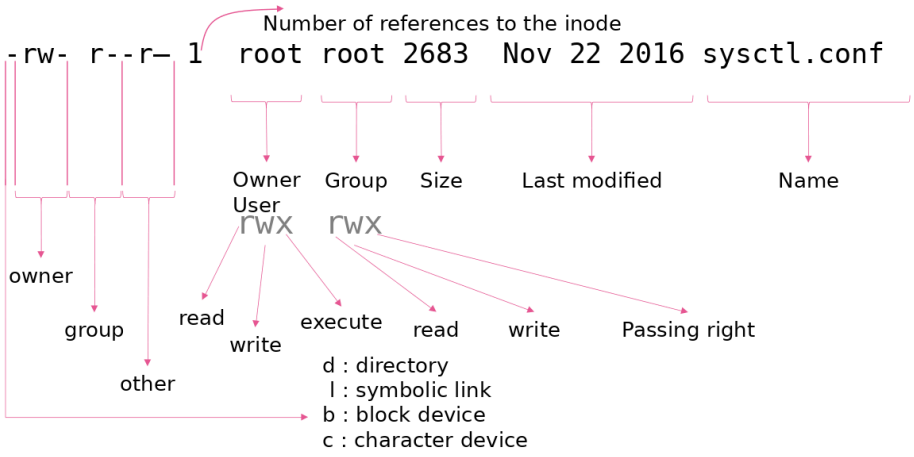


Figure 15: The output of 'ls -l' creates this type of output

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

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

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

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

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

Figure 16: Octal notation

5.5 Overview of basic commands

Usage and details for these commands: see labs

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

5.6 Wooclap

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

6 Text editors, Piping, Redirection & Jobs

6.1 Text editors

- Emacs (productivity, extensibility)

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

Our choice: Vim

6.1.1 vi vs vi-improved

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

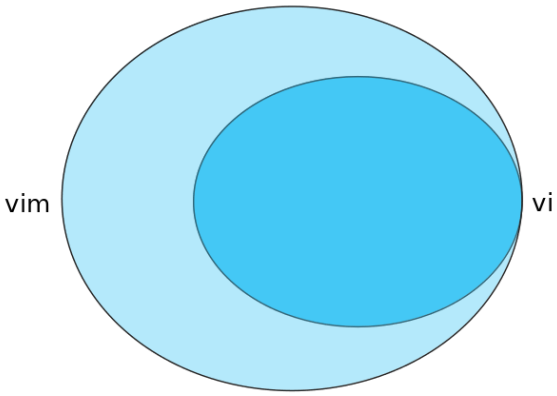


Figure 17: Everything you can do in vi, you can do in vim, and more!

6.1.2 First steps in vim

```
# start vim:
$~ vim

# start vim tutorial:
$~ vimtutor
```

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

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

```
p # put: paste the copied line

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

6.1.3 Search and replace

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

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

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

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

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

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

#undo:
(ESC) u
```

6.1.4 Basic editing tricks

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

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

# delete line 3
(ESC) 3gg
dd

# delete the next 4 lines
(ESC) 4dd

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

For more tricks: see labs

6.2 Piping

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

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

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

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

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

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

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

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

6.3 Redirection

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

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

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

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

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

6.3.1 stdout and stderr

= 2 important output streams

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

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

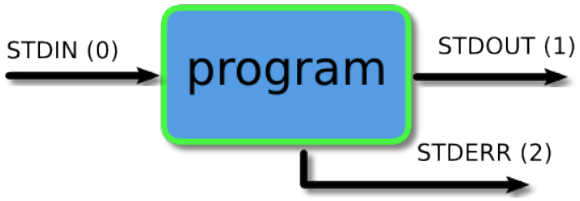


Figure 18

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

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

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

6.3.2 stdin

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

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

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

6.4 Jobs and process Management

When you execute a command: a process is started

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

```
# report a snapshot of current processes
ps

```

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

6.4.1 Exit codes

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

# 0 = ended successfully without errors

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

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

~$ echo $?
0

~$ test -r doesnotexist

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

6.4.2 Combining commands

= not the same as piping!

```
# note the difference between & (run in background) and && (combine command)

# one ampersand: run something in the background
test -r test.txt & echo "MCT rocks"

# two ampersands: combine commands
# if the first command exits 0, the second command will work
~# test -r test.txt && echo "MCT rocks"

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

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

6.4.3 Jobs

A job is a new process originating from the same parent

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

# bring a job to the foreground
fg <index>

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

6.4.4 Inter-process Communication

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

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

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

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

Signals

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

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

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

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

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

6.5 Intermezzo: System Load

= a number which represents the load on a computer system

- Completely idle system: system load 0
- Each process which uses a resource or is waiting for a resource: system load + 1
- Gives an indication of how heavy a computer system is loaded
- System load is a snapshot, doesn't say anything

- System load of 17: is that a problem? No.
- More interesting: the evolution of the systemload over time

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

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

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

6.6 Some useful tips

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

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

6.6.2 TTY

= Tele TYpewriter = a terminal which is connected with stdin

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