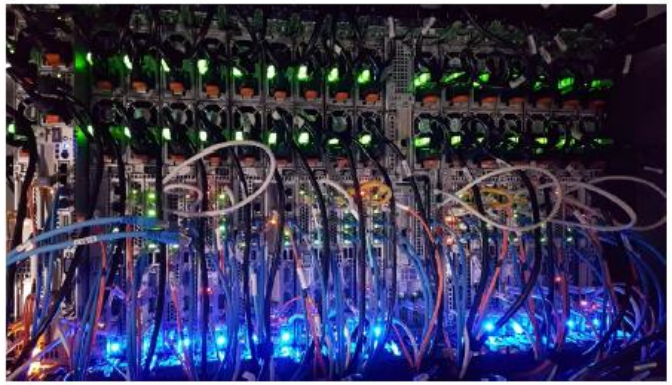
# Linux Fundamentals

# What is Linux?

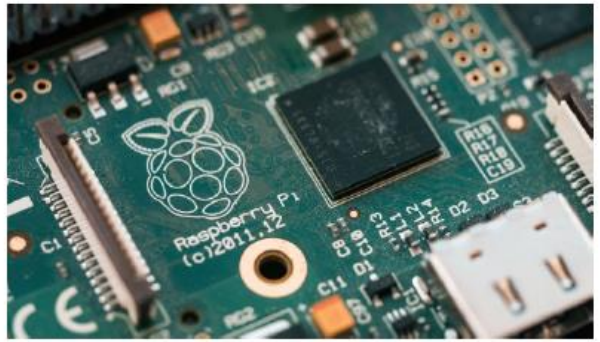
## Use Cases



Servers
(Web, Email, File, Database, Game,...)

Data Centers
(Cloud Computing)

Internet of Things (IoT)



Micro-Computers

Cyber-Security

# What is Linux?

- **Linux** is the kernel of a family of open-source Unix-like operating systems (OS)
- **OS**: A system software that manages computer hardware and provides various services for computer programs
- **Unix:** A powerful, multitasking, multi-user OS that serves as the basis for many modern OS
- **Kernel**: The **kernel** is the core part of an OS
  - It is responsible for managing the hardware, including the CPU, memory, and other devices

# What is the GNU Project?

- GNU stands for **G**NU's **N**ot **U**nix:
  - **The goal:** To create a free and open-source operating system
  - Initiated in 1983
  - Most GNU software is released under the **GNU General Public License** (GPL), promoting the freedom to use, modify, and distribute the code
  - The Linux kernel, created in 1993 by Linus Torvalds, is released under the GPL
- GNU provides many of the additional tools & utilities that we want to use
- Linux provides the kernel
- Together: **GNU/Linux**

# Linux Distributions

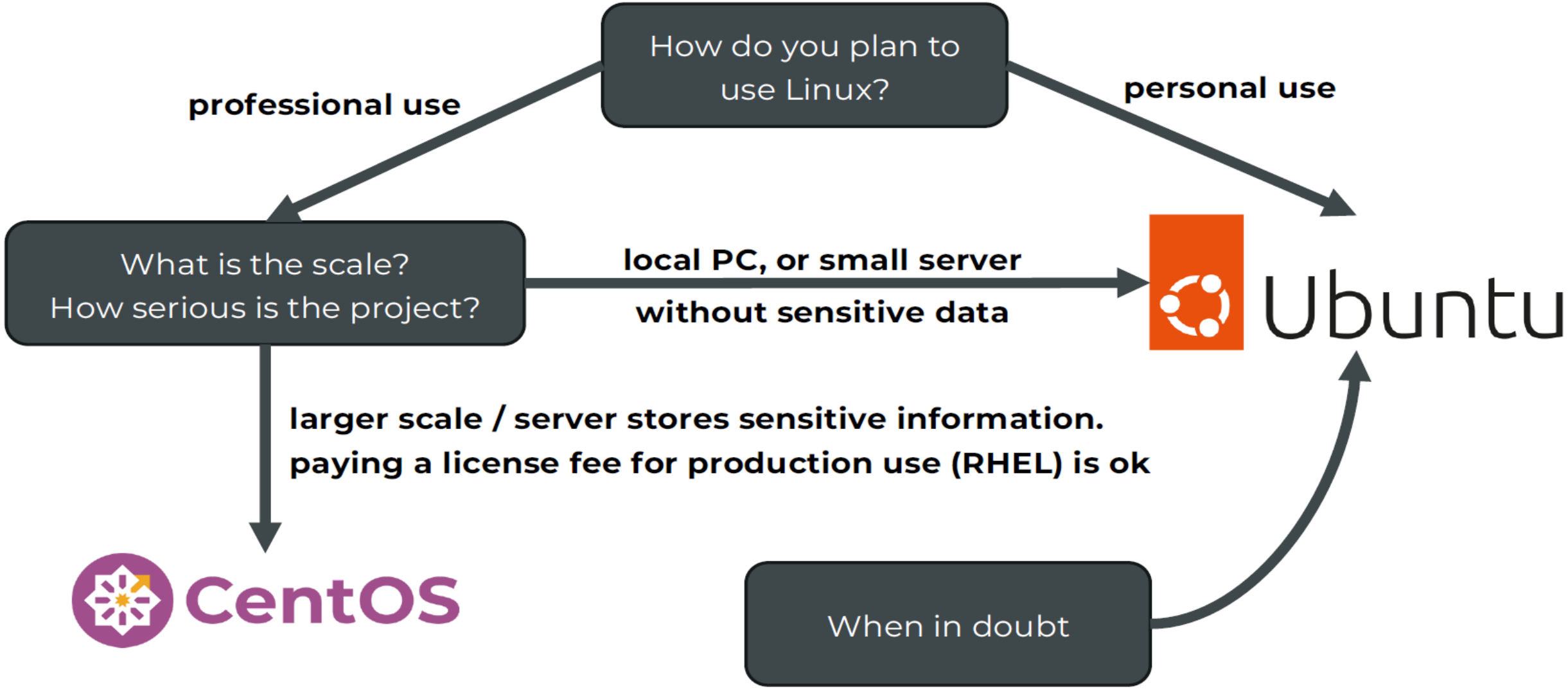
- There are various OS built upon GNU/Linux, called Linux distributions (distros)
- Each distribution then adds additional, specific software:
- This allows to cater to the needs of a specific user group
- Most Linux distributions are free

# Linux Distributions

- **There are 2 popular distributions:**
  - **Ubuntu:**
    - Part of the Debian family
    - Designed to be user-friendly
  - **CentOS Stream:**
    - Part of the Red Hat family
    - It's free to use, and allows us to familiarize ourselves with the Red Hat family
- **We will later have an overview about additional Linux distributions!**

# Which Linux Distribution Should You Choose?

# Which Linux Distribution Should You Choose?



# How To Run Linux

# How to run Linux

- **We could install Linux as a main operating system:**
  - You could then reboot and choose to boot into Linux (depending on how you install it: "dual-boot")
- **But for learning Linux:**
  - It's better to run Linux virtualized
  - This means we create a virtual computer that runs on your computer
  - And on this virtual computer, we'll run Linux

# How to run Linux

- **The advantages:**
  - If we destroy your system - nothing to worry about
  - We can just reinstall it
  - And we can easily switch between different Linux systems...
  - ... or even run them at the same time!

# What is VirtualBox?

- VirtualBox is a **virtualization software** that allows you to create and run virtual machines (VMs) on your computer
  - Developed by Oracle Corporation
  - Free & open-source software
- **Installation**:
  - https://www.virtualbox.org/wiki/Downloads

# How To Install Ubuntu

# Installing Ubuntu in VirtualBox

- **Ubuntu:**
  - You also need to download the **Ubuntu Desktop image**
  - This image contains the whole operating system - including an installer
  - We can then create a virtual machine, and let VirtualBox install Ubuntu for us
  - https://ubuntu.com/
  - **Important:**
    - **We will be using the latest LTS version of Ubuntu in this course**

# VirtualBox: Configuring Ubuntu

# Configuring Ubuntu

- We need to install some drivers to be able to work smoothly with our virtual machine
- **For this:**
  - We need to update our system
  - We need to install tools required to compile the drivers
  - And then we need to install the drivers

# VirtualBox: How to install CentOS Stream

Jerry Bame | AC-IT Consulting

# Installing CentOS Stream in VirtualBox

- **CentOS Stream:**
  - We need to manually create an empty virtual machine
  - And then install CentOS Stream into it
- **For this:**
  - **We need to go to the website of CentOS:**
  - https://www.centos.org/
  - And download the latest version of CentOS Stream from there!

# VirtualBox: How to configure CentOS Stream

- We need to install some drivers to be able to work smoothly with
- our virtual machine
- **For this:**
  - We need to update our system
  - We need to install additional ways to fetch additional software
  - We need to update our system (one more time)
  - We need to install tools required to compile the drivers
  - And then we need to install the drivers

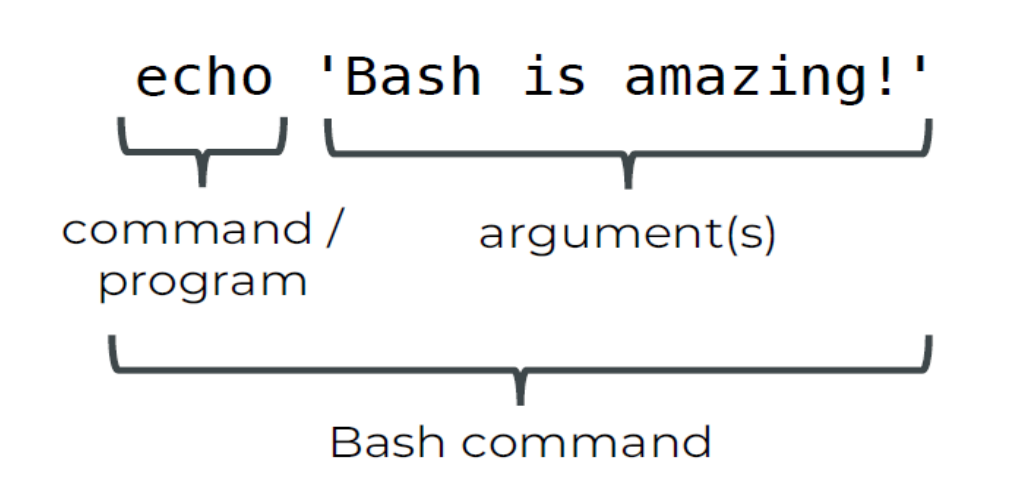
# Let’s confirm that you’re using Bash

- Let's first confirm that you're running a Bash
- **We can do this with the following command:**
- echo "${BASH_VERSION}"
- **If it doesn't output anything (or just an empty line):**
- Be sure to launch Bash with the following command:
- bash
- For now, you will need to do this any time you open a new window / tab in your terminal

# A first command in the Terminal

# The command echo

- echo allows us to output text in the terminal
- **We can test this:**
- echo 'Bash is amazing!'
- **For now:**
- The string that we want to print should be wrapped in **single quotes**



# The command echo

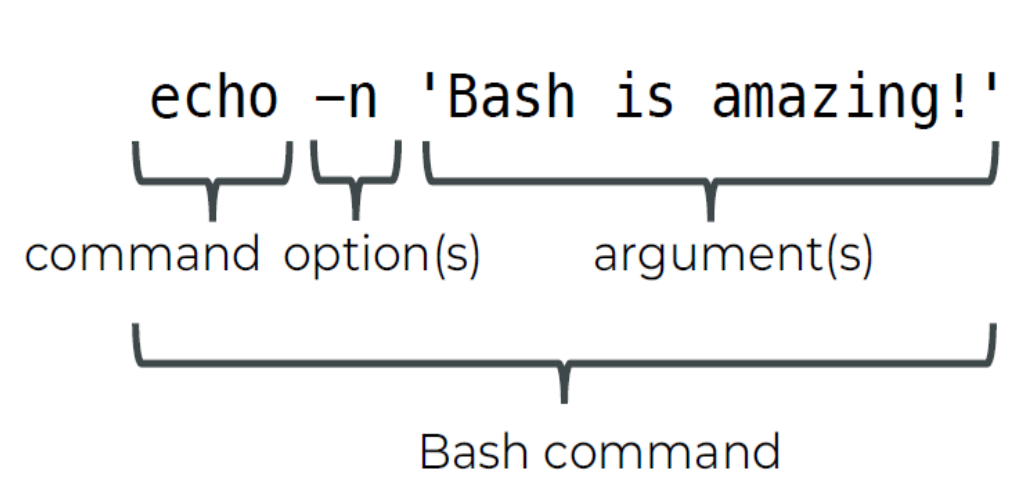
- **By default:**
  - echo will by default output a line break at the end
  - We can disable this with the option: -n
  - **Example:**

    echo -n 'Bash is amazing!'

- **We can also use different options:**
  - The option -e enables backslash escapes
  - **Example:**

    echo -e 'Line 1\nLine 2'

  - Here, \n will now be converted into a line break



# Combining Options

- Quite often, we want to combine multiple options
- **Usually, all of those are equivalent:**
  - echo -e -n 'Hello\nworld'
  - echo -en 'Hello\nworld'
  - echo -ne 'Hello\nworld'

# Navigating the filesystem

# The pwd command

- **In our terminal:**
  - We're always in a directory
  - This directory is called the ***working directory***
  - It is the folder that the shell is currently operating at
  - Commands can access this folder and act relative to this directory
- **To find out our current directory:**
  - pwd
  - (**p**rint **w**orking **d**irectory)

# Changing the directory:cd

- If we want to change the directory, we can use the cd command
- cd stands for: "change directory"
- **Examples:**
- **cd [directory_name]:**
- Moves into the directory specified by [directory_name]
- **Example:**
- **cd Desktop**
- **cd /**
- **cd ..:**
- Move into the parent directory
- **cd ~ or cd:**
- Move to the user's home directory
- **cd ~/Desktop:**
- Move to the user's desktop

# Listing contents of a folder

**To list the contents of the current working directory:**

- ls
- **We can also add additional options:**
- **Syntax:** ls [option...] [path]
- **-a:** List all entries, including hidden files starting with .
- We will learn more about hidden files later
- **-r:** Reverse order while sorting
- **-t:** Sort by modification time, newest first
- **--color:**
- Enables colorful output
- --color={always,never,auto}

**Example:**

- ls -ta --color ~/Desktop
- ls -ta --color=auto ~/Desktop

# Absolute vs. Relative Paths

**Absolute paths:**

- Start with a "/"
- They define the complete path to a file
- Thus, they work everywhere - no matter our current working directory

**Example:**

- /home/bmawein/Desktop
- ~/Desktop

**Relative paths:**

- Are being resolved according to our current working directory

**Example:**

- ./Desktop
- Desktop
- ../Desktop

# Executing Multiple Commands

- We can also execute multiple commands
- **For this, we can just add a semicolon between them:**

  command1; command2;
- **For example:**

  echo -n 'Hello '; echo 'World'

  echo -n 'Hello '; echo -n 'World '; echo  'of Linux Learning!'

# How to get Help in Linux

- **--help / -h:**
- For a lot of commands, we can just add a -h or a --help:
- ls --help
- We will then be shown a list of possible options and arguments

**man:**

- If this doesn't work, we can check the built-in manuals
- man ls

**Important:**

- man pages (documentation) must be installed on your system

**Otherwise:**

- Many tools have extensive online documentation
- Communities such as **Stack Overflow**, or **Reddit's Linux** communities can also be great for help

# First Steps with Linux

Jerry Bame | AC-IT Consulting

# User Management

**In Linux, users can be categorized into three general categories:**

**System accounts:**

- They are responsible for running background tasks on your system (such as: webserver, database,...)
- They don't have a home directory

**Regular users**:

- They have access to their own files and directories
- They cannot perform administrative tasks or access other user's files without permission

# User Management

**Superuser (root):**

- The superuser (root) has unrestricted access to the entire system (including files in the home directories of regular users)
- Can add / remove users, install software
- Can change the configuration of the system

# Elevating Privileges: sudo

- If we want to temporarily elevate our privileges, we can put a **sudo** in front of our command

**Example:**

- sudo ls /root
- Usually, we could not access this directory
- Only the root user can access it
- **sudo** elevates our privileges

**But be careful:**

- Always make sure you understand what a command does
- Especially when **sudo** is involved

**Example:**

- **Important: DO NOT EXECUTE!**
- sudo rm -rf /etc

# Package Management on Linux

Jerry Bame | AC-IT Consulting

# Package Management on Linux

**Package management:**

- Most Linux distributions offer a centralized way to install software
- This process is called package management
- This is an enormous benefit of many Linux systems as it helps us to keep our system up to date

**How does it work?**

- Our system connects to centralized repositories
- They provide a list of available packages (including available versions, and their dependencies)
- This list can then be used to install updates, or install additional tools

# How to Install Software in Ubuntu (apt)

- How to update software with apt
- On Debian-based distributions (such as Ubuntu), we can use the tool apt to keep our system up to date
- **It provides several commands for us:**

**apt update:**

- Refreshes the list of available packages
- We should run this before doing anything else with apt

# How to Install Software in Ubuntu (apt)

**apt upgrade:**

- Runs a small upgrade of our system
- **Small means:** Upgrades existing packages (and when using apt, also allows the installation of additional dependencies)

**apt full-upgrade or apt dist-upgrade:**

- Runs a large upgrade of our system
- **Large means:** Upgrades existing packages, and removes / installs additional packages (dependencies)

# How to install / remove software

- **To install / remove software:**

**apt install [package]:**

- This will install an additional package on our system
- **Example: apt install cowsay**

**apt remove [package]:**

- Removes a package from our system

**apt auto remove:**

- Removes packages that are no longer needed
- You can run this if there're any errors during an **upgrade** or **full-upgrade**

# CentOS: How to update software with dnf

- On RHEL-related distributions (such as CentOS Stream), we can use the tool dnf to keep our system up to date

**With just one command, we can keep our system up to date:**

- **dnf upgrade or dnf update:**
- Fetches the latest version of the package list and upgrades our system

**Important:**

- dnf always keeps the local package list up to date
- We don't have to manually refresh it (in contrast to Ubuntu)

# CentOS: How to install / remove software

**To install / remove software:**

- **dnf install [package]:**
- This will install an additional package on our system
- **Example:**
- dnf install epel-release
- dnf install cowsay
- **dnf remove [package]:**
- Removes a package from our system

# CentOS: How to install / remove software

**Important:**

- For now, please install **epel-release** on your system
- This adds additional package lists to be able to install additional tools on your system
- Command: **dnf update**; **dnf install epel-release**; **dnf update**;
- We should also run: **crb enable**
- We will have a more detailed look at **epel-release** later

# Heads-up: dnf

**Important:**

- There're additional commands for **dnf**
- We will have a more in-depth look at package management on CentOS later in this course

**Also, a quick heads-up:**

- **yum** used to be a package manager on CentOS previously
- Nowadays, it is replaced by **dnf**
- **But yum commands still works perfectly:**
- **yum install –y cowsay**

# MacOS: Package Management & Installing Bash

- On Mac, we can use Homebrew to install additional packages
- **We can find it here:**
- https://brew.sh/
- This gives us a package manager, that we can then use to install additional packages

**Once we've installed Homebrew:**

- **To update the package definitions:**
- brew update
- **To install a package:**
- brew install bash
- **And to install available updates:**
- brew upgrade

# Launching Bash on MacOS

- **Once we've installed Bash on a Mac:**
- We can launch Bash in Version 5.x+ by launching a terminal, and executing the following command:
- bash
- **We can confirm that we're running a recent version of Bash:**
- echo "${BASH_VERSION}"
- **To launch the original (3.x) version of Bash:**
- /bin/bash

# Important Points

- Now, you know quite a bit of the Linux operating system
- This is now a great foundation we can keep building on top
- **You can:**
  - Launch a terminal
  - Keep your system up to date
  - Install additional software
  - And you have even explored the first few commands in the Terminal

# Bash CLI

# Bash CLI

Overview

- We will have a look how to manage files through the command line
- We will see how we can redirect the output of programs into files / use files as input for programs
- We will see how we can use pipes to combine tools
- We will see how the Bash environment works and how this influences the way programs are executed
- And we will configure your prompt to achieve a more beautiful terminal

# File Management in Bash

# File Management in Bash: touch and mkdir

touch

- typical use case: create an empty file (or multiple files)
- **more precisely:**
  - to modify the timestamp of a file
  - if the file already exists, only its timestamp will be modified
  - otherwise, a new (and empty) file is created

mkdir (**m**a**k**e **dir**ectory)

- create a new directory

# File Management in Bash: mv and cp

mv (**m**o**v**e)

- move an existing file to another location
- can also be used to rename an existing file

mv file4.txt /home/bmawein/linuxtraining/test4.txt

cp (**c**o**p**y)

- to copy an existing file

cp file1.txt /home/bmawein/linuxtraining/

- cp -R: copies a whole folder

cp -R awstraining/ /home/bmawein/linuxtraining/

# How to delete files and folders

rm (remove)

touch file{1..4}.txt

rm file1.txt

- to remove a file (or multiple files at once)
- for deleting a directory, you need to use the option –r
- works for empty and non-empty directories

rmdir (remove directory)

- to delete an empty directory

mkdir dir{1..4}

rmdir dir2 dir3 dir4

- To delete a directory with files:

rm -rf dir1