Introduction to Linux: **Overview**

Philipp Fruck & Pius Walter

Introduction

About us

- Pius & Philipp
- Studied CS back in 2019
- First Linux lecture in 2025
  - We still update/rework the lecture every year
  - We need your feedback!

Introduction

About us

- Pius & Philipp
- Studied CS back in 2019
- First Linux lecture in 2025
  - We still update/rework the lecture every year
  - We need your feedback!

About you

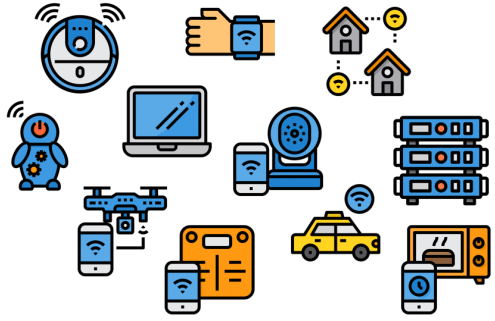
Quickly introduce yourself:

- Who are you?
- Which operating systems did you use previously
  - Windows, macOS, Linux? Or only mobile devices?

Organizational

- 11 lectures, each 1.5 hours
- Mostly hands-on
  - You'll need your own Linux system
- Introduction into Linux system components
  - Installation
  - Basic & advanced terminal commands
  - Building applications and containers
- Practical lab work submissions

What is Linux?



Icons by flaticon.com

What is Linux?



https://img.devrant.com/devrant/ran
1578772_VbG6J.jpg

- In 1983, Richard Stallman started the GNU Project (**G**NU's **N**ot **U**nix)
  - Open source compiler (gcc), system library (glibc) and coreutils
  - But: Lacking a kernel

What is Linux?



https://img.devrant.com/devrant/ran
1578772_VbG6J.jpg

- In 1983, Richard Stallman started the GNU Project (**G**NU's **N**ot **U**nix)
  - Open source compiler (gcc), system library (glibc) and coreutils
  - But: Lacking a kernel
- In 1991, Linus Torvalds developed his own kernel
  - Started as a hobby project
  - Similar to proprietary Unix, but open source

What is Linux?



https://img.devrant.com/devrant/ran
1578772_VbG6J.jpg

- In 1983, Richard Stallman started the GNU Project (**G**NU's **N**ot **U**nix)
  - Open source compiler (gcc), system library (glibc) and coreutils
  - But: Lacking a kernel
- In 1991, Linus Torvalds developed his own kernel
  - Started as a hobby project
  - Similar to proprietary Unix, but open source
- The components were combined to create a fully open source operating system

What is Linux?



https://img.devrant.com/devrant/ran
1578772_VbG6J.jpg

- In 1983, Richard Stallman started the GNU Project (**G**NU's **N**ot **U**nix)
  - Open source compiler (gcc), system library (glibc) and coreutils
  - But: Lacking a kernel
- In 1991, Linus Torvalds developed his own kernel
  - Started as a hobby project
  - Similar to proprietary Unix, but open source
- The components were combined to create a fully open source operating system
- Could run programs originally developed for Unix
  - Fast adoption from developers

What is Linux?



https://img.devrant.com/devrant/ran
1578772_VbG6J.jpg

- In 1983, Richard Stallman started the GNU Project (**G**NU's **N**ot **U**nix)
  - Open source compiler (gcc), system library (glibc) and coreutils
  - But: Lacking a kernel
- In 1991, Linus Torvalds developed his own kernel
  - Started as a hobby project
  - Similar to proprietary Unix, but open source
- The components were combined to create a fully open source operating system
- Could run programs originally developed for Unix
  - Fast adoption from developers

When we say "Linux", we are referring to an entire operating system

- There are opinions that a Linux based system should be called "GNU/Linux"
- Nowadays: Memes about "systemd/Linux"

Advantages of Linux

- Runs (almost) everywhere
  - From low-power to high performance computing (https://top500.org)
- Open Source
  - All source code is publicly available
  - Can be analyzed and modified to your liking
  - Great for academia
- Customizable/Configurable
- Free of charge*

Components of a Linux system

Component	Example	Description
Bootloader	grub, systemd-boot	Starts the system
Kernel	Linux	Interfaces with hardware
Init System	systemd / openrc	Starts and manages system services
Display Server	X11, Wayland	Renders the graphical user interface (GUI)
Display Manager	GDM, LightDM	Graphical login screen
Desktop Environment	Gnome, KDE, XFCE	Defines how your GUI looks and behaves
GUI toolkit/framework	GTK, QT, Electron	Allows building GUI apps with different look and feel
Security Framework	SELinux, AppArmor	Optional, enhanced security regulation

Components of a Linux system

Component	Example	Description
Bootloader	grub, systemd-boot	Starts the system
Kernel	Linux	Interfaces with hardware
Init System	systemd / openrc	Starts and manages system services
Display Server	X11, Wayland	Renders the graphical user interface (GUI)
Display Manager	GDM, LightDM	Graphical login screen
Desktop Environment	Gnome, KDE, XFCE	Defines how your GUI looks and behaves
GUI toolkit/framework	GTK, QT, Electron	Allows building GUI apps with different look and feel
Security Framework	SELinux, AppArmor	Optional, enhanced security regulation

---

Most distros do not ship/configure a security framework by default

- AppArmor is shipped/enabled by default on Debian/Ubuntu
- SELinux is shipped/enabled by default on the RedHat distro family (including Fedora)
- SUSE provides configuration for both AppArmor and SELinux

Linux Desktop Environments

A **Desktop Environment (DE)** defines the look & feel of your Linux system

Includes: panels/menus, settings, file manager, system tools

Linux Desktop Environments

A **Desktop Environment (DE)** defines the look & feel of your Linux system

Includes: panels/menus, settings, file manager, system tools

Desktop Environment	Characteristics	Target audience
GNOME	Modern, minimal, keyboard-friendly	Users who like a clean workflow
KDE Plasma	Highly customizable, Windows-like, many settings	Power users, tinkerers
COSMIC	Gnome-inspired, written in Rust, supports tiling	Power users who like simplicity
XFCE	Lightweight, classic interface	Older hardware, performance-focused
LXQt / LXDE	Extremely lightweight	Very resource-constrained systems
Cinnamon	Traditional desktop (Windows-like)	Linux Mint users, beginners
MATE	Fork of old GNOME 2, lightweight	Users who want a stable, classic desktop
Budgie	Modern, elegant, GNOME-based	Users who like simplicity + polish
Pantheon	macOS-like, minimalistic	Users who like macOS feel (elementary OS)

Linux Desktop Environments

A **Desktop Environment (DE)** defines the look & feel of your Linux system

Includes: panels/menus, settings, file manager, system tools

Desktop Environment	Characteristics	Target audience
GNOME	Modern, minimal, keyboard-friendly	Users who like a clean workflow
KDE Plasma	Highly customizable, Windows-like, many settings	Power users, tinkerers
COSMIC	Gnome-inspired, written in Rust, supports tiling	Power users who like simplicity
XFCE	Lightweight, classic interface	Older hardware, performance-focused
LXQt / LXDE	Extremely lightweight	Very resource-constrained systems
Cinnamon	Traditional desktop (Windows-like)	Linux Mint users, beginners
MATE	Fork of old GNOME 2, lightweight	Users who want a stable, classic desktop
Budgie	Modern, elegant, GNOME-based	Users who like simplicity + polish
Pantheon	macOS-like, minimalistic	Users who like macOS feel (elementary OS)

> Choice of DE = personal preference -> try several to find your favorite!

> DE is **not tied to the distro** -> you can install others later

What is a Linux Distribution?

- Windows, macOS, etc. only have a single OS with different Versions
  - Windows 10, Windows 11, ...

What is a Linux Distribution?

- Windows, macOS, etc. only have a single OS with different Versions
  - Windows 10, Windows 11, ...
- Linux has a much greater variety of system components
  - Different Desktops, different init systems, different apps

What is a Linux Distribution?

- Windows, macOS, etc. only have a single OS with different Versions
  - Windows 10, Windows 11, ...
- Linux has a much greater variety of system components
  - Different Desktops, different init systems, different apps
- A Linux Distro bundles certain components together:
  - Different kernel versions, different desktops, different package repositories
  - Distros support various release cycles and models

What is a Linux Distribution?

- Windows, macOS, etc. only have a single OS with different Versions
  - Windows 10, Windows 11, ...
- Linux has a much greater variety of system components
  - Different Desktops, different init systems, different apps
- A Linux Distro bundles certain components together:
  - Different kernel versions, different desktops, different package repositories
  - Distros support various release cycles and models
- Distros are opinionated
  - Software selection based on certain preferences
  - Some distros only ship open source software components (Debian, Fedora)
  - Some distros compile everything from source (Gentoo)
  - Different out-of-the-box security configuration (SELinux, AppArmor, Firewall frontends)
  - Different package managers and package formats
  - Desktop vs Server focus
  - etc ...

Which Linux Distros are there?

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

Which Linux Distros are there?

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

A couple to point out:

- **Slackware** is one of the oldest distros, but nowadays almost obsolete
- **Debian** is a very stable (mostly server) distro which focuses on free software (community driven)
- **Ubuntu** is a newcomer friendly distro based on **Debian**, owned by Canonical
- **Linux Mint** is likely the most recommended newcomer distro based on **Debian** *or* **Ubuntu**

Which Linux Distros are there?

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

A couple to point out:

- **Slackware** is one of the oldest distros, but nowadays almost obsolete
- **Debian** is a very stable (mostly server) distro which focuses on free software (community driven)
- **Ubuntu** is a newcomer friendly distro based on **Debian**, owned by Canonical
- **Linux Mint** is likely the most recommended newcomer distro based on **Debian** *or* **Ubuntu**

- **Arch Linux**, community driven, focuses on customization and has bleeding edge software
- **Gentoo** is a source based distro -> software is compiled locally
- **NixOS** is a declarative configurable distro

Which Linux Distros are there?

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

A couple to point out:

- **Slackware** is one of the oldest distros, but nowadays almost obsolete
- **Debian** is a very stable (mostly server) distro which focuses on free software (community driven)
- **Ubuntu** is a newcomer friendly distro based on **Debian**, owned by Canonical
- **Linux Mint** is likely the most recommended newcomer distro based on **Debian** *or* **Ubuntu**

- **Arch Linux**, community driven, focuses on customization and has bleeding edge software
- **Gentoo** is a source based distro -> software is compiled locally
- **NixOS** is a declarative configurable distro

- **Fedora** is a community driven distro that focuses on modern software and security
- **RedHat** and **SUSE** offer *Enterprise Linux* (paid)
  - **Alma** and **Rocky** are community editions of **RedHat Enterprise Linux** (RHEL)
  - **SUSE Linux Enterprise Server** (SLES) has free community editions **Leap** and **Tumbleweed**
  - **Fedora** used as **RHEL** upstream -> Community + RedHat driven
  - **Ubuntu LTS** is considered *Enterprise Linux* as well

Which Linux Distros are there?

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

A couple to point out:

- **Slackware** is one of the oldest distros, but nowadays almost obsolete
- **Debian** is a very stable (mostly server) distro which focuses on free software (community driven)
- **Ubuntu** is a newcomer friendly distro based on **Debian**, owned by Canonical
- **Linux Mint** is likely the most recommended newcomer distro based on **Debian** *or* **Ubuntu**

- **Arch Linux**, community driven, focuses on customization and has bleeding edge software
- **Gentoo** is a source based distro -> software is compiled locally
- **NixOS** is a declarative configurable distro

- **Fedora** is a community driven distro that focuses on modern software and security
- **RedHat** and **SUSE** offer *Enterprise Linux* (paid)
  - **Alma** and **Rocky** are community editions of **RedHat Enterprise Linux** (RHEL)
  - **SUSE Linux Enterprise Server** (SLES) has free community editions **Leap** and **Tumbleweed**
  - **Fedora** used as **RHEL** upstream -> Community + RedHat driven
  - **Ubuntu LTS** is considered *Enterprise Linux* as well

- **Kali**/**Parrot OS** are targeted towards pentesting/security auditing (no daily-driving)
- **Alpine** is a minimal Linux distro focusing on minimal overhead (e.g., resource-constraint hardware/containers)

Which Linux Distros are there?



Linux distro release cycles

Distro	Release model	Cadence & support lifecycle
Slackware	Fixed / irregular	No fixed cadence; long-lived releases
Debian	Fixed	~2 yr; ~5 yr support
Ubuntu	Fixed	Each April and October; 9 mo support
Ubuntu LTS	Fixed	~2 yr; ~5 yr support (10 yr when paid)
Linux Mint	Fixed	~2 yr; follows Ubuntu LTS
Fedora	Fixed	~6 mo; ~1 yr support
Arch Linux	Rolling	Continuous; no fixed lifecycle
Gentoo	Rolling	Continuous; no fixed lifecycle
NixOS	Fixed / rolling channels	~6 mo; stable channels maintained
RHEL	Fixed / enterprise	Major ~3–4 yr; 10 yr lifecycle (multiple minor versions)
SUSE SLES	Fixed	Major ~3 yr; minor ~1 yr; ~5 yr support (LTS extends further)
SUSE Leap	Fixed	~2 yr; ~2 yr support
SUSE Tumbleweed	Rolling	Continuous; no fixed lifecycle
Alpine	Fixed + rolling edge	~2–3 releases/yr; defined support period

Linux distro release cycles

Distro	Release model	Cadence & support lifecycle
Slackware	Fixed / irregular	No fixed cadence; long-lived releases
Debian	Fixed	~2 yr; ~5 yr support
Ubuntu	Fixed	Each April and October; 9 mo support
Ubuntu LTS	Fixed	~2 yr; ~5 yr support (10 yr when paid)
Linux Mint	Fixed	~2 yr; follows Ubuntu LTS
Fedora	Fixed	~6 mo; ~1 yr support
Arch Linux	Rolling	Continuous; no fixed lifecycle
Gentoo	Rolling	Continuous; no fixed lifecycle
NixOS	Fixed / rolling channels	~6 mo; stable channels maintained
RHEL	Fixed / enterprise	Major ~3–4 yr; 10 yr lifecycle (multiple minor versions)
SUSE SLES	Fixed	Major ~3 yr; minor ~1 yr; ~5 yr support (LTS extends further)
SUSE Leap	Fixed	~2 yr; ~2 yr support
SUSE Tumbleweed	Rolling	Continuous; no fixed lifecycle
Alpine	Fixed + rolling edge	~2–3 releases/yr; defined support period

No, this will not be an exam question ;)

Installing Software

Windows

Traditional

- `.exe` or `.msi` as **installer**
  - Manually downloaded from some website
- Contains required application files or downloads them from the internet
- Each application ships all dependencies by itself -> no sharing

Installing Software

Windows

Traditional

- `.exe` or `.msi` as **installer**
  - Manually downloaded from some website
- Contains required application files or downloads them from the internet
- Each application ships all dependencies by itself -> no sharing

Modern

- Winget (CLI package manager) or Microsoft Store
  - No need to download some sketchy installer
  - Each program still ships all its dependencies

Installing Software

Windows

Traditional

- `.exe` or `.msi` as **installer**
  - Manually downloaded from some website
- Contains required application files or downloads them from the internet
- Each application ships all dependencies by itself -> no sharing

Modern

- Winget (CLI package manager) or Microsoft Store
  - No need to download some sketchy installer
  - Each program still ships all its dependencies

Linux

Traditional

- Package managers
  - Every application is installed via system package manager
  - Each distro provides its own repositories with different packages and versions
- Shared libraries are packaged as dependencies
  - Applications actually reuse shared dependencies
  - Upside: Much more space efficient
  - Downside: More complicated packaging, "dependency hell"

Installing Software

Windows

Traditional

- `.exe` or `.msi` as **installer**
  - Manually downloaded from some website
- Contains required application files or downloads them from the internet
- Each application ships all dependencies by itself -> no sharing

Modern

- Winget (CLI package manager) or Microsoft Store
  - No need to download some sketchy installer
  - Each program still ships all its dependencies

Linux

Traditional

- Package managers
  - Every application is installed via system package manager
  - Each distro provides its own repositories with different packages and versions
- Shared libraries are packaged as dependencies
  - Applications actually reuse shared dependencies
  - Upside: Much more space efficient
  - Downside: More complicated packaging, "dependency hell"

Modern

- AppImage: Self-contained binary (no package manager)
- Snap: Package manager for self-contained applications (GUI + CLI), proprietary store operated by Canonical
- Flatpak: Package manager (decentralized) for sandboxed, self-contained GUI applications
- Toolpak: Supposed to be flatpak for CLI

Package Managers

A **package manager**

- installs, updates, and removes software
- keeps track of dependencies (e.g., libraries required by applications)

Different distros use different package managers and package formats:

Distro Family / Type	Package Manager	Package Format
Debian / Ubuntu	apt / dpkg	.deb
Fedora / RHEL	dnf / yum	.rpm
Arch Linux	pacman	.pkg.tar.zst
Alpine Linux	apk	.apk
NixOS	nix	store paths
Universal formats	flatpak, snap, appimage	portable

> Some systems (e.g., rpm-ostree, transactional-update) are immutable - changes apply atomically after reboot.

> Most package managers can update the entire system with a single command!

Linux file system explained

In Windows, different disk partitions have different letters.

In Linux, there is the file system tree. Directories in the tree are separated via `/`.

Each disk can be *mounted* at any path in the tree.

Linux file system explained

In Windows, different disk partitions have different letters.

In Linux, there is the file system tree. Directories in the tree are separated via `/`.

Each disk can be *mounted* at any path in the tree.

`/` is called the **root directory** -> everything starts here

Linux file system explained

In Windows, different disk partitions have different letters.

In Linux, there is the file system tree. Directories in the tree are separated via `/`.

Each disk can be *mounted* at any path in the tree.

`/` is called the **root directory** -> everything starts here

Some directories are **virtual**

- e.g. (`/proc`, `/sys`)
- these directories exist only in memory and are not stored on the disk

Linux file system explained

```
/
├── bin -> usr/bin      # Essential user commands (ls, cp, mv)
├── boot                # Bootloader files (kernel, initramfs)
├── dev                 # Device files (e.g. /dev/sda, /dev/tty)
├── etc                 # System configuration files
├── home                # User home directories (/home/alice)
├── lib -> usr/lib      # Shared libraries for essential programs
├── lib64 -> usr/lib64  # 64-bit libraries
├── media               # Auto-mounted removable media (legacy)
├── mnt                 # Temporary mount points (manual)
├── opt                 # Optional / 3rd-party software
├── proc                # Virtual filesystem with process info
├── root                # Home directory for root user
├── run                 # Volatile runtime data (since last boot)
├── sbin -> usr/sbin    # System binaries (fsck, ip, systemctl)
├── srv                 # Data served by the system (web, ftp)
├── sys                 # Virtual filesystem for hardware devices
├── tmp                 # Temporary files (auto-cleared)
├── usr                 # User programs, libraries, docs
└── var                 # Variable data (logs, cache, spool)
```

Binaries & Their Locations

Directory	Purpose
/bin	Essential user commands needed for booting and repairing the system (e.g., `ls`, `cp`, `mv`).
/sbin	Essential system binaries for booting and administration (e.g., `fsck`, `ip`, `mount`).
/usr/bin	Most user applications (non-essential, e.g., `vim`, `git`, `python`).
/usr/sbin	Admin and system tools that are not essential for booting (e.g., `sshd`, `apachectl`).
/usr/local/bin	Locally installed user applications (not managed by package manager).
/usr/local/sbin	Locally installed system admin tools (manual builds, custom scripts).
/home/<user>/.local/bin/	User-specific binaries installed via `pip`, `cargo`, `npm` etc. - only available to that user.
/opt/	3rd-party applications - often entire self-contained software bundles (e.g., Google Chrome, proprietary tools).

Binaries & Their Locations

Directory	Purpose
/bin	Essential user commands needed for booting and repairing the system (e.g., ls, cp, mv).
/sbin	Essential system binaries for booting and administration (e.g., fsck, ip, mount).
/usr/bin	Most user applications (non-essential, e.g., vim, git, python).
/usr/sbin	Admin and system tools that are not essential for booting (e.g., sshd, apachectl).
/usr/local/bin	Locally installed user applications (not managed by package manager).
/usr/local/sbin	Locally installed system admin tools (manual builds, custom scripts).
/home/<user>/.local/bin/	User-specific binaries installed via pip, cargo, npm etc. - only available to that user.
/opt/	3rd-party applications - often entire self-contained software bundles (e.g., Google Chrome, proprietary tools).

- Modern Linux: /bin and /sbin are often symlinks to /usr/bin and /usr/sbin
- Local and user-specific directories let you install software without touching the system

Binaries & Their Locations

Directory	Purpose
/bin	Essential user commands needed for booting and repairing the system (e.g., `ls`, `cp`, `mv`).
/sbin	Essential system binaries for booting and administration (e.g., `fsck`, `ip`, `mount`).
/usr/bin	Most user applications (non-essential, e.g., `vim`, `git`, `python`).
/usr/sbin	Admin and system tools that are not essential for booting (e.g., `sshd`, `apachectl`).
/usr/local/bin	Locally installed user applications (not managed by package manager).
/usr/local/sbin	Locally installed system admin tools (manual builds, custom scripts).
/home/<user>/.local/bin/	User-specific binaries installed via `pip`, `cargo`, `npm` etc. - only available to that user.
/opt/	3rd-party applications - often entire self-contained software bundles (e.g., Google Chrome, proprietary tools).

- Modern Linux: `/bin` and `/sbin` are often symlinks to `/usr/bin` and `/usr/sbin`
- Local and user-specific directories let you install software without touching the system

All standard directories are described in the file system hierarchy manual page - run `man hier` to read it.

Drivers on Linux

- Drivers allow the **kernel** to talk to hardware

Drivers on Linux

- Drivers allow the **kernel** to talk to hardware
- Most drivers are included in the Linux kernel -> No need for manual installation

Drivers on Linux

- Drivers allow the **kernel** to talk to hardware
- Most drivers are included in the Linux kernel -> No need for manual installation
- Plug & Play is common:
  - Network adapters, USB devices, webcams usually work out of the box

Drivers on Linux

- Drivers allow the **kernel** to talk to hardware
- Most drivers are included in the Linux kernel -> No need for manual installation
- Plug & Play is common:
  - Network adapters, USB devices, webcams usually work out of the box
- Exceptions:
  - Some GPUs (e.g., Nvidia) have proprietary drivers
  - Some Wi-Fi chipsets require extra firmware
  - Printers: Try to use AirPrint (driverless)

Drivers on Linux

- Drivers allow the **kernel** to talk to hardware
- Most drivers are included in the Linux kernel -> No need for manual installation
- Plug & Play is common:
  - Network adapters, USB devices, webcams usually work out of the box
- Exceptions:
  - Some GPUs (e.g., Nvidia) have proprietary drivers
  - Some Wi-Fi chipsets require extra firmware
  - Printers: Try to use AirPrint (driverless)
- Tools to check drivers:
  - `lspci`, `lsusb` -> list devices
  - `lsmod` -> list kernel modules (drivers)
  - `dmesg` -> check system log for driver messages
  - More on that in the later lectures

How do I install my own distro?

We recommend using the latest version of Fedora Workstation from https://fedoraproject.org/workstation/

Installing Fedora as dedicated Linux system or via dual boot (if you want to keep Windows) is recommended. We do not recommend using a VM.

How do I install my own distro?

We recommend using the latest version of Fedora Workstation from https://fedoraproject.org/workstation/

Installing Fedora as dedicated Linux system or via dual boot (if you want to keep Windows) is recommended. We do not recommend using a VM.

Preparing for Installation

- **Back up all of your data**
- Download the ISO file (and verify the checksum) or use the Fedora Media Writer
- Create a bootable media using dd or the Fedora Media Writer tool

How do I install my own distro?

We recommend using the latest version of Fedora Workstation from https://fedoraproject.org/workstation/

Installing Fedora as dedicated Linux system or via dual boot (if you want to keep Windows) is recommended. We do not recommend using a VM.

Preparing for Installation

- **Back up all of your data**
- Download the ISO file (and verify the checksum) or use the Fedora Media Writer
- Create a bootable media using dd or the Fedora Media Writer tool

Installation

- Boot from your USB stick (use UEFI if possible)
- Fedora will boot into a live system, from where you can start the graphical installation
  1. Select the system language and keyboard layout
  2. Define the installation destination (the disk where you want Fedora installed). You can choose to use the entire disk or share it with an existing operating system (dual boot)
  3. Configure storage encryption (recommended for notebooks and portable devices for better security)
  4. Review your configuration and install Fedora Workstation

Thank you!

Thank you for your attention!

Don't forget the feedback in Moodle please!