Open Operating Systems

Linux — the base of the system

Łukasz Gołek

WIMiC AGH

Linux Tree | /dev Directory | Permissions | Everything is a File | VFS

Arch Linux and Gentoo — Course Plan

COURSE OVERVIEW

1. Hardware, files and user permissions
2. The Arch Way & Installation (Part I)
3. Installation (Part II) & System Configuration
4. Bootloaders & Network Management
5. Package Management
6. Xorg, Wayland & Desktop Environments
7. Gentoo – The Deep Dive: Introduction to Gentoo & Portage
8. USE Flags & Optimization
9. The Linux Kernel (The Manual Way)
10. Your suggestions

You don't have to know everything at once!

The most important thing is that you understand how your system works — the rest will come with time. I've been using Linux for over thirty years and I still don't remember the syntax of less frequently used commands. Don't worry.

Linux is a simple, logical system once you understand the fundamentals.

SECTION 1

VM Setup

For students running AGHOS in VirtualBox

AGHOS GRUB Bootloader

SECTION 1 · VM SETUP

Select your language variant and press Enter to boot.

Editing Kernel Boot Parameters

SECTION 1 · VM SETUP

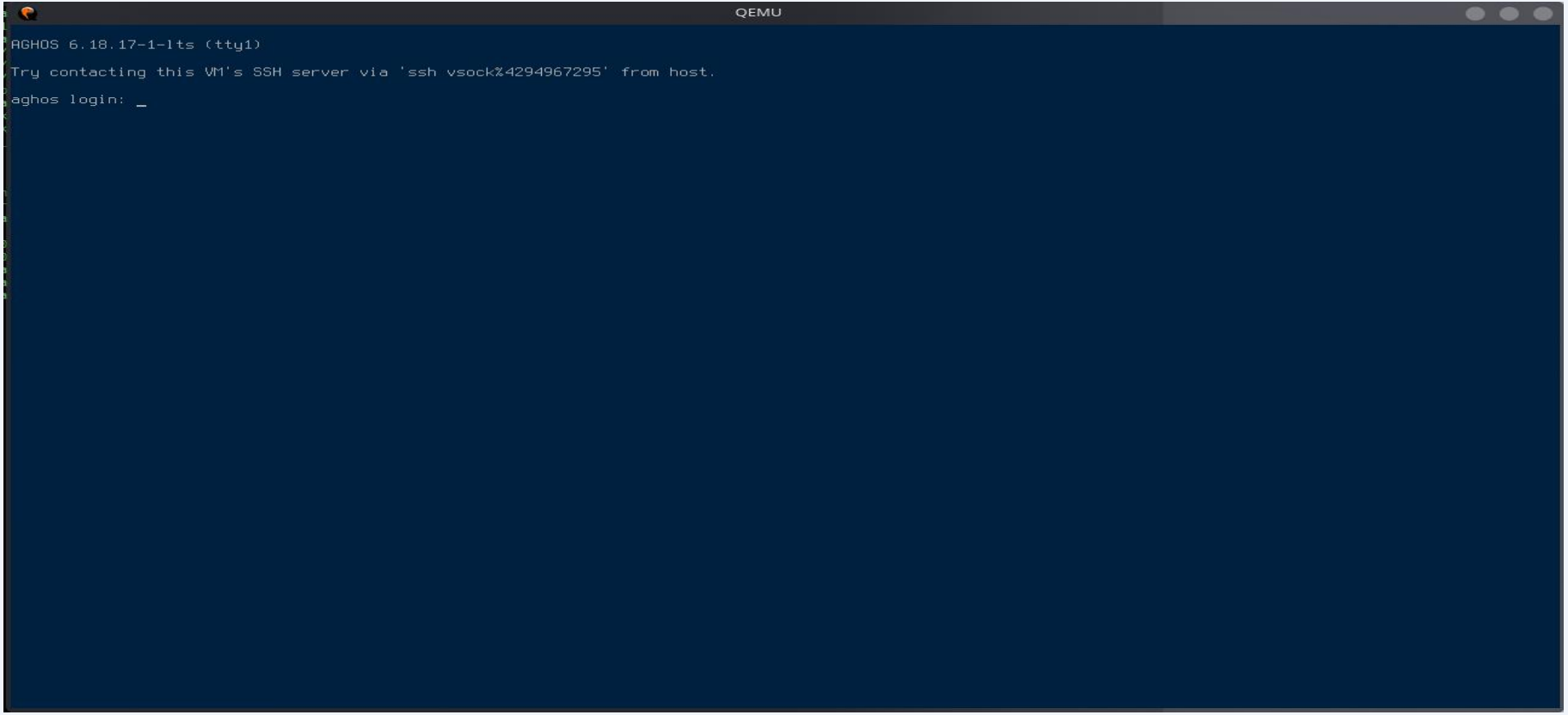
How to edit kernel parameters

1. Select the desired entry in GRUB
2. Press E to edit the boot command
3. Navigate to the end of the kernel line
4. Add the number 3 at the end
5. Press Ctrl+X or F10 to boot

Adding '3' tells the kernel to boot into runlevel 3 — multi-user mode without a graphical interface.

AGHOS Login Screen

SECTION 1 · VM SETUP



Type aghos as the login — no password required.

Setting a Password & Switching Users

SECTION 1 · VM SETUP

```
Login as: aghos   (no password)
$ sudo su              # become root
# passwd aghos         # set a temporary learning password
```

The password is temporary — it will be lost when the VM restarts.

su

switch user

Switches to another user account. Requires the target user's password. Starts a new shell as that user. By default switches to root.

sudo

superuser do

Executes a single command with elevated privileges. Requires your own password. Controlled via /etc/sudoers for fine-grained permissions.

sudo su

get root shell

Runs su as root using sudo. Gives you a root shell without requiring the root password. A common shortcut.

passwd

change password

Change your own password.
sudo passwd john — change another user's password as superuser (root).

Connecting via SSH (Recommended)

SECTION 1 · VM SETUP

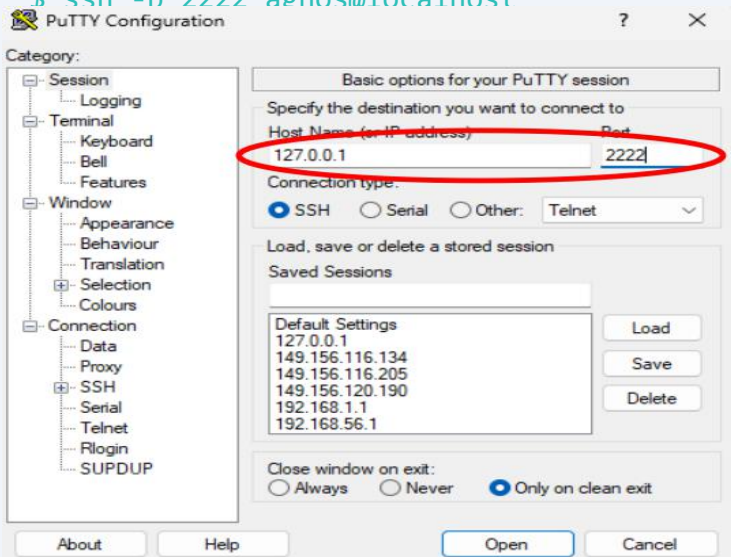
ű Optional but highly recommended — SSH allows proper copy/paste (Ctrl+C / Ctrl+Shift+V) instead of typing directly in VirtualBox.

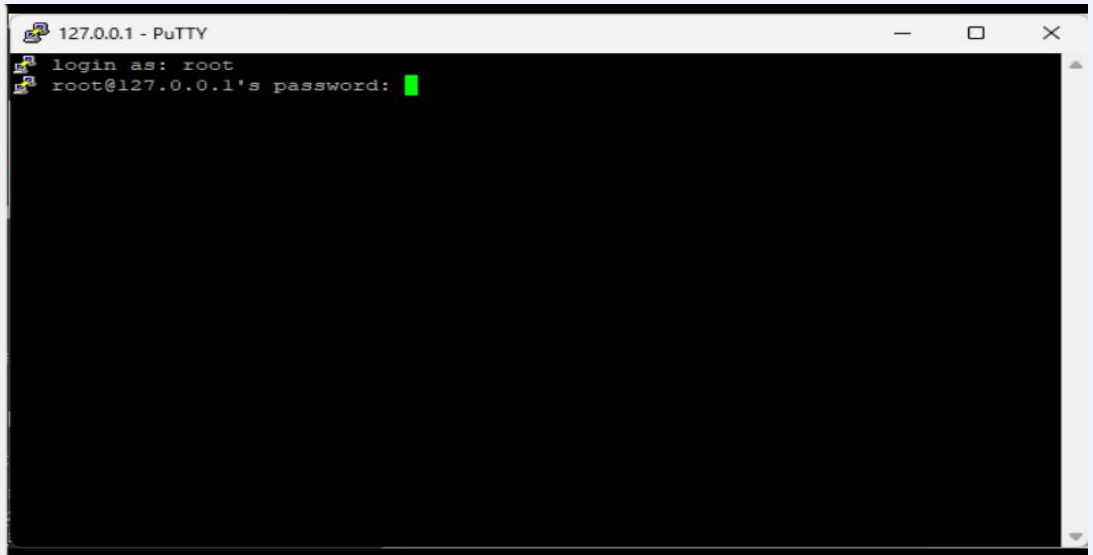
```
# Inside the VM:
$ systemctl start sshd

# From your host machine:
$ ssh -p 2222 aghos@localhost
```

How it works

VirtualBox forwards port 2222 on the host to port 22 inside the VM. Any terminal works: PuTTY, Konsole, Windows Terminal.





SECTION 2

The Linux Directory Tree

Understanding the filesystem hierarchy

The most important directory in the entire Linux system.

Every file, every folder, every device — it all lives somewhere under /.

Think of it as the root of a tree: all branches grow from here.

The Full Linux Directory Hierarchy

SECTION 2 · DIRECTORY TREE

```
/
├── /bin    essential user binaries (ls, cp, bash…)
├── /boot   boot files — bootloader, kernel
├── /dev    device files — hardware as files ← we'll explore this
├── /etc    system configuration files
│    ├── /etc/passwd  — user account definitions
│    ├── /etc/fstab   — filesystem mount table
│    └── /etc/hosts   — static hostname-to-IP mappings
├── /home   user home directories
├── /lib    system libraries
├── /mnt    mount points for external resources
├── /opt    optional / third-party software
├── /proc   virtual FS — process & kernel info (not on disk)
├── /root   home directory of the root user
├── /sys    virtual FS — hardware management interface
├── /tmp    temporary files (cleared on reboot)
├── /usr    programs and libraries not needed for boot
│    ├── /usr/bin    non-essential user commands
│    ├── /usr/lib    libraries for /usr/bin
│    ├── /usr/local  locally compiled software
```

Key Distinction: /root vs /home

SECTION 2 · DIRECTORY TREE

```
/
├── /home                  ← where regular users live
│    ├── /home/aghos
│    ├── /home/albert
│    ├── /home/kasia
│    └── /home/ziggie
└── /root                  ← home of the root (admin) user — NOT the same as /
```

/home — regular users

Every regular user gets their own subdirectory here. When you log in as a regular user, this is your starting directory. Typically the largest partition — holds movies, music, documents.

/root — administrator home

The home directory of the root user. Intentionally separate from /home — so even if /home is on a different partition or unavailable, root can always log in.

'/' (root directory) ≠ 'root user'. Same word — completely different concepts. Keep this in mind — it trips up many beginners.

Key Directories — What Lives Where

SECTION 2 · DIRECTORY TREE

/dev

Every connected device — hard drive, USB stick, keyboard — has a corresponding file here. Linux treats ALL hardware as files.

/etc

The system's control panel. All configuration files for services, users, and programs live here. Think of it as the Windows Registry, but human-readable.

/bin

Essential commands every user can run: ls, cp, mv, bash — the basic building blocks of working in the terminal.

/boot

Everything needed to start the system: the kernel itself and bootloader configuration. Without this, the system won't start.

/usr

Where most installed programs and their libraries live — the 'software' directory of the system.

/tmp

Temporary files. Usually mounted as tmpfs (RAM-backed). Cleared on every reboot. Great for scratch space.

Critical Distinction: /home vs /var

SECTION 2 · DIRECTORY TREE

/home — large files, rare writes

User files: movies, music, pictures, documents, games.

- Files are typically gigabytes in size
- Written once, read many times
- Access patterns: mostly sequential reads
- Changes rarely

Best served by: Btrfs with compression (zstd)
→ saves 30–60% on compressible content

/var — small files, constant writes

Variable data: logs, database records, mail queues, package caches.

- Files are often kilobytes in size
- Thousands of tiny writes per second
- Access patterns: random small I/O
- Changes constantly

Best served by: XFS or ext4
→ avoid Btrfs COW (write amplification!)

Remember these two — they will explain filesystem choices, partition strategies, and mount options in coming lectures.

You probably already know these filesystems:

FAT (12, 16, 32)

The File Allocation Table — the universal USB filesystem. 4 GB file size limit. No Unix permissions. Required for /boot/efi.

exFAT

Microsoft's FAT for Flash. Removes the 4 GB limit. Still no Unix permissions. Widely supported.

NTFS

Windows default since Windows NT. Has journaling. Supports permissions. Readable/writable on Linux, but not ideal.

Next lecture: Linux-native filesystems — ext4, XFS, Btrfs, and why they matter.

SECTION 3

The /dev Directory

Everything is a file — including your hardware

/dev

Short for devices. In Linux, everything is treated as a file — including hardware.

Your hard drive, USB stick, keyboard, terminal, webcam — they all appear as files in /dev.

The kernel creates these files at boot time. They exist only in memory, not on disk.

In Linux, Every Hardware Device Is a File

SECTION 3 · /dev

The same tools you use for regular files (cat, cp, dd) work identically with hardware device files.

/dev/sda

First serial drive (SATA or USB).

/dev/sda1

First partition of the first serial drive.

/dev/sda2

Second partition of the first serial drive.

/dev/sdb1,2…

Partitions on the second serial drive.

/dev/video0

First video capture device (webcam). video1 for the second.

/dev/input/mouse0

First mouse. Every movement, every click flows through this file as a stream.

Pattern: device type + number starting from zero. The number means 'first found', 'second found', and so on.

Device Files — The Uniform Interface

SECTION 3 · /dev

Device files are not typical files, but they can be controlled using the same four operations:

`open()`	Open the device file to get a file descriptor.
`read()`	Read data from the device — bytes, blocks, or a stream.
`write()`	Send data to the device — write to disk, send to printer, etc.
`close()`	Release the file descriptor and the device.

The same C code that reads /etc/hosts also reads /dev/sda. The kernel routes the call — your program doesn't need to know.

Device Types in /dev

SECTION 3 · /dev

Block Devices ⋆

`/dev/sda, /dev/nvme0n1`

Hard drives, SSDs, USB drives. Random access — you can jump (seek) to any position. Data stored in fixed-size blocks.

Character Devices ⋆

`/dev/tty, /dev/snd`

Terminals, keyboards, sound cards. Sequential stream — data flows byte by byte. No seeking backwards.

Network Devices

`/dev/eth0, /dev/enp0s3`

Ethernet and wireless interfaces. Exposed via /sys/class/net rather than /dev directly.

Pseudo-Terminals

`/dev/pts/0`

Virtual terminals used by SSH sessions, terminal emulators, screen/tmux.

Special / Crypto

`/dev/random, /dev/urandom, /dev/null`

Random number generators. /dev/null — the black hole that discards everything written to it.

Input Devices

`/dev/input/`

Keyboards, mice, joysticks, touchpads. Each gets its own file under /dev/input/.

Everything Is a File — Linux's Unified I/O Model

SECTION 3 · /dev

The Unix Philosophy

Unix, since the 1970s, built its entire I/O model on one single abstraction: the file.

It doesn't matter if you're talking about data on disk, a serial port, or a network socket — you open it, read it, write to it, and close it. That is literally the whole interface.

The same syscalls work for files, devices, sockets, and processes.

```
read   (fd, buf, n)
write  (fd, buf, n)
open   (path, flags)
close  (fd)
ioctl  (fd, cmd)

File Descriptor Table (per process):
  fd 0  stdin  →  /dev/tty
  fd 1  stdout →  /dev/tty
  fd 2  stderr →  /dev/tty
  fd 3  file   →  /home/data.txt
```

/dev/sda

Block device (HDD/SSD)

/dev/tty0

Char device (terminal)

/proc/cpuinfo

Virtual FS (kernel data)

/dev/null

Special file (discard)

socket

Network (TCP/UDP)

pipe

IPC (stdin→stdout)

Block Devices vs Character Devices

SECTION 3 · /dev

Block Devices

- Fixed-size data blocks
- E.g., Hard Drives, SSDs

/dev/sda1

Character Devices

- Stream of data (bytes)
- E.g., Cameras, Mice, Microphones

/dev/video0 /dev/input/mouse0

Interfaces to Hardware Devices

Special files for communicating with hardware

Block: random access, fixed-size sectors, buffered I/O. Character: sequential stream, no buffering, no seeking.

Block Devices — Data Access Model

SECTION 3 · /dev

`/dev/sda, /dev/nvme0n1, /dev/mmcblk0 — random access, buffered I/O`

1

process calls read(fd, buf, 4096)

The read() syscall goes to the VFS — the Virtual Filesystem Switch.

2

VFS checks the Page Cache

Cache HIT → return data immediately from RAM. Zero disk I/O. This is why reading the same file twice is instant.

3

Cache MISS → Block I/O Layer

VFS descends to the Block I/O layer. The mq-deadline scheduler optimises read order to minimise seek time.

4

nvme/ahci driver → physical storage

The driver communicates directly with the storage hardware. Data returns up the stack.

Character Devices — Data Access Model

SECTION 3 · /dev

/dev/tty, /dev/urandom, /dev/input/event0 — sequential stream, no buffering, no seeking

Page Cache	Block: ű Yes — data is buffered in RAM. Same file read twice → instant.	Char: ╳ No — every read() goes directly to the driver.
lseek()	Block: ű Yes — you can jump to any sector in the disk.	Char: ╳ No — you cannot rewind a stream of bytes from a keyboard.
Access pattern	Block: Random — sectors accessed in any order.	Char: Sequential — bytes flow in one direction only.
Examples	Block: /dev/sda, /dev/nvme0n1, /dev/mmcblk0	Char: /dev/tty, /dev/input/event0, /dev/urandom

The elegance of Unix: the same abstraction — files — but two completely different behaviours underneath. That is why everything-is-a-file works.

SECTION 4

Linux Groups & Permissions

Users, groups, chmod, chown

Linux Groups & Permissions — Reference

SECTION 4 · PERMISSIONS

ℹ This is a reference slide — study it at your own pace. I'll share the full presentation after class.

1 — Managing Groups

`groupadd audio`

create group 'audio'

`groupadd -g 1050 sci`

with explicit GID

`groupdel audio`

delete group

`groupmod -n sound audio`

rename group

`getent group audio`

inspect members

`cat /etc/group`

all groups on system

2 — Assigning Users

`usermod -aG audio alice`

add alice to audio group

`usermod -aG disk,video bob`

add to multiple groups

`gpasswd -d alice audio`

remove user from group

`id alice`

show all groups of user

`groups alice`

list user's groups

`newgrp audio`

switch active group

3 — chmod & chown

`chmod 660 /dev/dsp`

rw-rw---- (owner+group)

`chmod u+x script.sh`

add execute for owner

`chmod o-r secret.txt`

remove read from others

`chown root:audio /dev/dsp`

set owner:group

`chmod 4755 /usr/bin/sudo`

setuid bit (run as owner)

`chmod 2770 /srv/shared`

setgid (inherit group)

Octal: 4=read 2=write 1=exec | owner=hundreds group=tens others=units | setuid=4000 setgid=2000

Privilege Inheritance — Why Linux Is Safe

SECTION 4 · PERMISSIONS

Every process in Linux inherits the UID/GID of the user who launched it — no exceptions. A program cannot give itself more privileges than its parent had.

What malware CAN do (as alice)

→ Read/delete/encrypt everything in /home/alice
→ Exfiltrate alice's documents, SSH keys, browser data
→ Send emails or HTTP requests as alice
→ Crypto-mine using alice's CPU quota

Removing the infection

rm -rf /home/alice
→ wipes user + malware in one command
→ system untouched, /etc intact, other users unaffected

useradd -m alice # fresh account, clean slate
Total remediation time: under 60 seconds.

Windows comparison

╳ Drivers run in ring 0 — malware can escalate to kernel level
╳ Registry survives user deletion — malware persists across reinstall
╳ COM/DLL injection across process boundaries — lateral movement
╳ Cleaning requires full AV scan + manual registry audit + service inspection
→ multi-hour forensic exercise, not a one-liner

The Unix permission model was designed in 1969. It is 55 years old. It still contains malware better than any modern AV product on Windows.

SECTION 5

Linux vs Windows Architecture

HAL, backward compatibility, and the cost of 'it just works'

Linux vs Windows: Architectural Differences

SECTION 5 · ARCHITECTURE

	LINUX ▶	⚠ WINDOWS
Kernel model	▶ Monolithic kernel with loadable modules. Drivers can be inserted/removed at runtime.	HAL (Hardware Abstraction Layer) — legacy since Windows NT 3.1. Never rewritten.
Hardware access	▶ Direct hardware access via VFS — clean, predictable.	Win32 API compatibility layer on top of HAL — layers on layers.
Driver safety	▶ Drivers compiled or loaded as modules. A crash doesn't necessarily kill the system.	Drivers run in ring 0 — same privilege as kernel. One faulty driver = BSOD.
Configuration	▶ Files in /etc — human-readable, version-controllable, scriptable.	Registry — a monolithic binary database. Difficult to audit or version.
Standards	▶ POSIX standard — predictable, portable interfaces.	Decades of backward compatibility shims dating to 1985.
Security	▶ SELinux, AppArmor, namespaces, cgroups, capability system.	Attack surface grows with every new compatibility layer added.

Windows Security Architecture — A Visual Metaphor

SECTION 5 · ARCHITECTURE

New hoses attached to old hoses, tape over the leaks, more leaks appearing every year. That is literally what Windows looks like on the inside.

Microsoft's Business Model — Structural Complexity as a Moat

SECTION 3 · ARCHITECTURE

Hypothesis — not an accusation of intentional backdoors, but a structural observation about incentive alignment.

1. Structural Lock-in

The Win32/HAL/Registry architecture creates enormous switching costs. Cleaning up legacy would break millions of enterprise apps — Microsoft knows this and has no economic incentive to fix it.

2. Fear as a Service

Regular high-profile vulnerabilities (EternalBlue, PrintNightmare, MSHTML…) drive enterprise customers toward Microsoft's own security products: Defender, Sentinel, Entra ID. The problem and the solution are sold by the same vendor.

3. Patch Paralysis

Deep HAL coupling means patching one subsystem risks breaking drivers, OEM hardware, or line-of-business apps. Microsoft can show 'effort' while delivering limited actual reduction in attack surface.

SECTION 6

Practical: The Terminal

Let's fight with the shell

/dev — Quick Reference

SECTION 6 · PRACTICAL

```
# Storage devices
/dev/sda          ← first hard drive
/dev/sdb          ← second hard drive
/dev/sdc          ← third hard drive
/dev/sda1         ← partition 1 of first drive
/dev/sda2         ← partition 2 of first drive

# Network (via /sys/class/net)
/sys/class/net/eth0
/sys/class/net/eth1
/sys/class/net/wlan0

# Cameras / video
/dev/video0       ← first camera
/dev/video1       ← second camera

# Special
/dev/null         ← discard everything written here
/dev/random       ← cryptographically secure random
```

Why this matters

In the Linux system, hardware is visible as a file. This is completely different from Windows.

Because hardware is a file, you can:

- dd if=/dev/sda of=disk.img — clone an entire disk
- cat /dev/random | head -c 32 | xxd — read random bytes
- echo 'hello' > /dev/tty1 — write to another terminal
- lsof /dev/sda — see which process has the disk open

The same mental model, the same tools, everywhere.

Let's fight with the shell

Popular terminal emulators:

```
xterm   aterm   uxterm   konsole   yakuake   kitty   alacritty   gnome-terminal
```

Pick any terminal you like. The shell inside is what matters — most likely bash or zsh. All behave the same from a learning perspective.

⚠ **Never work from the root account on network stations!**

Essential Navigation Commands

SECTION 6 · PRACTICAL

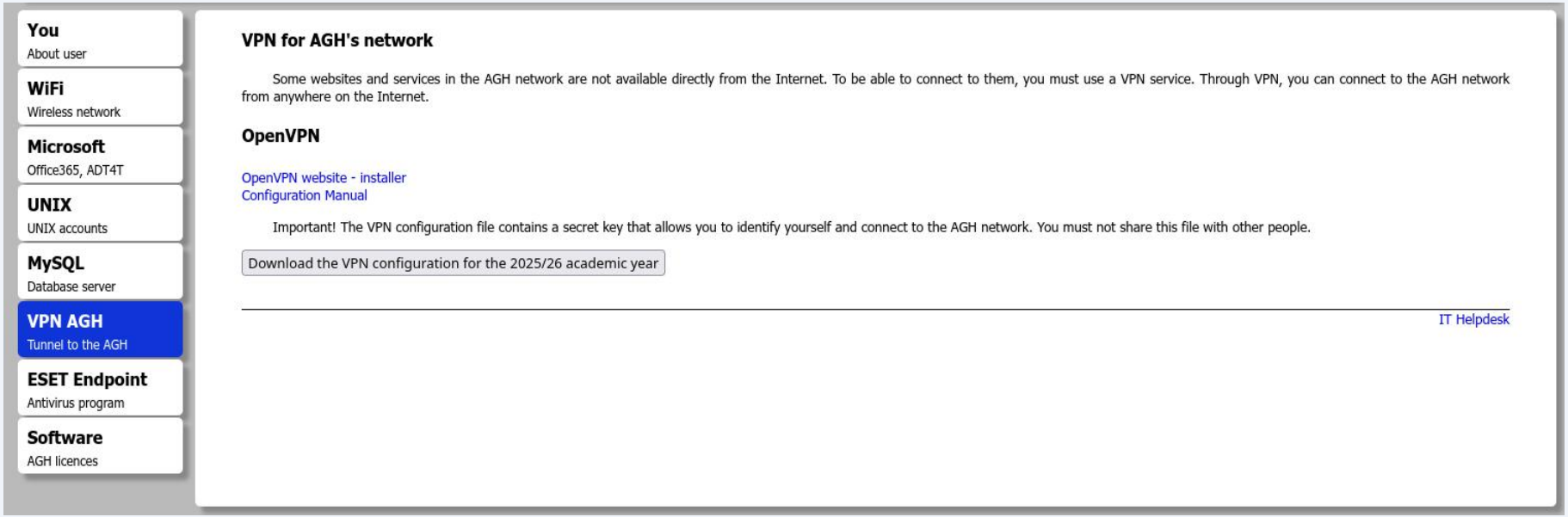
1. `pwd`
   print working directory
2. `ls -alh`
   list files
3. `cd ~ / cd .. / cd .`
   change directory
4. `mv src dst`
   move / rename
5. `touch filename`
   create or timestamp
6. `cat file`
   concatenate / print
7. `grep 'pattern' file`
   global regex print
8. `wc -l file`
   word counter
9. `du -sh path`
   disk usage
10. `df -h`
    disk free

Before Next Class — Activate AGH VPN

SECTION 6 · HOMEWORK

You will need AGH VPN from the next class onwards. Activate it before you arrive.

Go to: https://panel.agh.edu.pl/



VPN AGH → Download the OpenVPN configuration for 2025/26 · Do not share the config file — it contains your personal key.