LABORATORUL 2: Navigarea în sistemul de fișiere

Durată estimată: 2,5 – 3 ore
Obiective: Înțelegerea structurii FHS, navigare eficientă, montarea filesystem-urilor și căutarea fișierelor

---

2.1 Introducere

După ce am învățat să găsim informații despre comenzi și să identificăm sistemul, este timpul să înțelegem cum este organizat filesystem-ul Linux și cum ne deplasăm eficient prin el. Acest capitol acoperă:

Filesystem Hierarchy Standard (FHS) - organizarea directoarelor
Comenzi de navigare (pwd, cd, ls, tree)
Montarea și unmontarea filesystem-urilor
Căutarea fișierelor (locate, find, whereis)
Use cases practice în telecomunicații

De ce este important:

Găsești rapid fișiere de configurare
Înțelegi unde să faci backup-uri
Troubleshooting eficient (unde sunt logs?)
Lucru uniform pe orice distribuție Linux

2.2 Filesystem Hierarchy Standard (FHS)

FHS definește structura standard a directoarelor în sistemele Linux/Unix. Această organizare asigură că:

Toate distribuțiile Linux au aceeași structură de bază
Aplicațiile știu unde să caute fișiere de configurare
Administratorii găsesc rapid ce caută
Backup-urile și migrările sunt predictibile

2.2.1 Principii fundamentale FHS

1. Totul pornește de la root (`/`)

Spre deosebire de Windows care are drive-uri separate (C:, D:, E:), Linux organizează totul într-o ierarhie unică pornind de la `/` (root).

```
# Windows structure (multiple trees):
C:\
D:\
E:\

# Linux structure (single tree):
/
├── bin
├── etc
├── home
├── var
└── ...
```

2. Separarea pe categorii logice:

Statice vs. Variabile:

`/usr` - conținut static (nu se modifică frecvent)

`/var` - conținut variabil (logs, cache, crește constant)

Shared vs. Specific:

`/usr` - poate fi shared între multiple sisteme (read-only în producție)

`/etc` - config specific fiecărui sistem

Essential vs. Optional:

`/bin`, `/sbin` - comenzi esențiale pentru boot

`/usr/bin` - comenzi non-esențiale (instalate după boot)

3. "Everything is a file"

În Linux, tot ce poți accesa este reprezentat ca fișier:

Fișiere obișnuite - documente, scripts
Directoare - containere pentru alte fișiere
Device files (`/dev/`) - acces la hardware
Sockets - comunicare între procese
Pipes - transfer date între procese

Overview directoare principale:

```
# Listare directoare root
ls -l /
```

```
# Vizualizare ca arbore (nivel 1)
tree -L 1 /
```

Exercițiul 2.1:
Explorare inițială root:

1. Rulați `ls -l /` - listați toate directoarele din root
2. Numărați câte directoare există
3. Observați culorile diferite (dacă `ls --color=auto` este setat):
   Albastru = directoare
   Cyan = symbolic links
   Alb = fișiere normale
4. Rulați `tree -L 1 /` - vizualizare ca arbore (doar nivel 1)

Exercițiul 2.2:
Verificare permisiuni root directories:

1. Rulați `ls -ld /bin /sbin /etc /home /root`
2. Observați owner-ul (majoritatea: root)
3. Observați permisiunile pentru `/root` vs `/home`
4. De ce `/root` are permisiuni mai restrictive? (hint: `drwx------`)

2.2.2 Directoare esențiale pentru boot

Aceste directoare conțin fișierele necesare pentru ca sistemul să pornească (boot process):

/bin - Binary executables (user)

Conține comenzi esențiale de care au nevoie TOȚI utilizatorii și care sunt necesare chiar și în single-user mode (recovery):

```
# Explorare /bin
ls /bin | head -20
```

```
# Comenzi comune în /bin:
# ls, cp, mv, rm, mkdir, cat, grep, ps, echo, , sh
```

De ce sunt aici și nu în /usr/bin?

Dacă `/usr` nu este montat (e.g., în recovery mode), aceste comenzi TREBUIE să fie disponibile
Sunt fundamentale pentru utilizare de bază

```
# Verificare tipuri fișiere
file /bin/ls
# Output: /bin/ls: ELF 64-bit LSB executable...

# Verificare dacă /bin este link simbolic către /usr/bin (modern)
ls -ld /bin
# Pe sisteme moderne (Ubuntu 20.04+): /bin -> usr/bin
```

/sbin - System binaries (admin)

Similar cu `/bin`, dar pentru comenzi de system administration:

```
# Explorare /sbin
ls /sbin | head -20

# Comenzi comune în /sbin:
# mount, umount, fdisk, iptables, reboot, shutdown, ifconfig
```

Diferența /bin vs /sbin:

`/bin` - orice user poate rula (ls, cat, echo)
`/sbin` - majoritatea necesită root (mount, reboot, iptables)

/boot - Boot loader și kernel

Conține tot ce este necesar pentru boot process:

```
# Explorare /boot
ls -lh /boot

# Fișiere importante:
# vmlinuz-*       → kernel-ul Linux
# initrd.img-*    → initial ramdisk (drivers pentru boot)
# grub/           → GRUB bootloader config
# config-*        → kernel compilation config
```

Ce se întâmplă la boot:

1. BIOS/UEFI citește GRUB din `/boot/grub/`
2. GRUB încarcă kernel-ul `vmlinuz-*`
3. Kernel încarcă `initrd.img-*` (temporary root filesystem)
4. Initrd montează root filesystem-ul real
5. Sistemul continuă boot-ul normal

```
# Verificare kernel curent folosit
uname -r
# Compară cu: ls /boot/vmlinuz-*
```

/lib - Shared libraries

Librării partajate (shared objects) necesare pentru executabilele din `/bin` și `/sbin`:

```
# Explorare /lib
ls /lib | head -20

# Librării importante:
# libc.so.*       → C standard library
# ld-linux*.so    → dynamic linker/loader
# modules/        → kernel modules (drivers)
```

Ce sunt shared libraries?

Cod refolosibil de multiple programe
Similar cu `.dll` files pe Windows
Extensie: `.so` (shared object)

```
# Vezi ce librării folosește o comandă
ldd /bin/ls

# Output:
# linux-vdso.so.1 => (0x00007fff...)
# libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
# /lib64/ld-linux-x86-64.so.2
```

Exercițiul 2.3:
Explorare directoare boot:

1. Listați `/bin` și identificați 10 comenzi pe care le recunoașteți
2. Listați `/sbin` și identificați comenzi de admin (mount, reboot, etc.)
3. Verificați dacă `/bin` este link simbolic: `ls -ld /bin`
4. Pe sisteme moderne, unde arată efectiv? (hint: `usr/bin`)

Exercițiul 2.4:
Investigație kernel:

1. Găsiți versiunea kernel curent: `uname -r`
2. Listați fișierele din `/boot`: `ls -lh /boot`
3. Găsiți kernel-ul corespunzător: `ls /boot/vmlinuz-*`
4. Verificați dimensiunea kernel-ului: `ls -lh /boot/vmlinuz-$(uname -r)`
5. Explorați `/boot/grub/grub.cfg` (doar primele 50 linii): `head -50 /boot/grub/grub.cfg`

Exercițiul 2.5:
Librării și dependențe:

1. Verificați dependențele comenzii `ls`: `ldd /bin/ls`
2. Câte librării folosește?
3. Găsiți locația `libc.so.6` (C standard library)
4. Repetați pentru `grep`: `ldd /bin/grep`
5. Observați librării comune între `ls` și `grep`

2.2.3 Directoare configurare și utilizatori

/etc - Configuration files

Cel mai important director pentru administratori! Conține TOATE fișierele de configurare pentru sistem și aplicații.

Principii /etc:

Fișiere text (human-readable)
Editabile cu orice text editor
Necesită root pentru modificare
Best practice: backup înainte de editare

```
# Explorare /etc
ls /etc | head -30

# Subdirectoare și fișiere importante:
# /etc/passwd        → user accounts info
# /etc/group         → group definitions
# /etc/hosts         → hostname to IP mapping
# /etc/hostname      → system hostname
# /etc/fstab         → filesystem mount configuration
# /etc/crontab       → scheduled tasks
# /etc/ssh/          → SSH server config
# /etc/apache2/      → Apache web server (Debian)
# /etc/httpd/        → Apache web server (RHEL)
# /etc/network/      → network configuration
# /etc/systemd/      → systemd configs
```

Organizare /etc:

Fișiere configurare globale → direct în `/etc/`
Configurări specifice aplicații → subdirectoare `/etc/app-name/`

```
# Exemple citire configs
cat /etc/hostname
# Output: server01

head /etc/passwd
# Output: user account info (username:x:UID:GID:...)

cat /etc/hosts
# Output:
# 127.0.0.1 localhost
# 192.168.1.10 server01
```

Pattern-uri comune în /etc:

Fișier/Director	Scop	Exemplu folosire
`/etc/app.conf`	Config principală app	`/etc/ssh/sshd_config`
`/etc/app.d/`	Config snippets	`/etc/sysctl.d/`
`/etc/default/app`	Default settings	`/etc/default/grub`
`/etc/systemd/system/`	Systemd units	Service configurations

/home - User directories

Fiecare utilizator are propriul subdirector în `/home`:

```
# Structură /home
ls -la /home

# Output:
# /home/user1/
# /home/user2/
# /home/admin/
```

Fiecare `/home/username` conține:

- Documente personale
- Fișiere configurare user (dotfiles: `.bashrc`, `.profile`)
- Desktop, Downloads, Documents (dacă GUI)
- Cache și config specifice aplicații (`.config/`, `.cache/`)

```
# Conținutul home-ului tău
ls -la ~/
# SAU
ls -la $HOME

# Fișiere ascunse importante (dotfiles):
# .bashrc       → shell configuration
# ._history     → command history
# .ssh/         → SSH keys
# .config/      → application configs
# .cache/       → application cache
```

Variabila $HOME:

```
echo $HOME
# Output: /home/username

cd ~
# Echivalent cu: cd $HOME
# Echivalent cu: cd /home/username
```

/root - Root user home

Home directory pentru user-ul root, **separat** de `/home`:

```
# Locație
ls -ld /root
# Output: drwx------ 5 root root ... /root

# De ce e separat?
# 1. Securitate - permisiuni mai restrictive (700)
# 2. /home poate fi pe o partiție separată care se montează mai târziu
# 3. root trebuie să aibă acces la home-ul său chiar dacă /home nu e montat
```

Exercițiul 2.6:
Explorare /etc:

1. Listați `/etc` și numărați câte fișiere și directoare sunt: `ls /etc | wc -l`
2. Găsiți fișierul de hostname: `cat /etc/hostname`
3. Găsiți fișierul hosts: `cat /etc/hosts`
4. Explorați un config de aplicație, ex: `ls /etc/ssh/` sau `ls /etc/apache2/`
5. Găsiți configurația fstab: `cat /etc/fstab`

Exercițiul 2.7:
Explorare home directory:

1. Verificați calea home-ului vostru: `echo $HOME`
2. Listați conținutul incluzând hidden files: `ls -la ~/`
3. Găsiți fișierul `.bashrc`: `cat ~/.bashrc | head -20`
4. Verificați istoricul comenzilor: `tail ~/._history`
5. Explorați `.ssh/` (dacă există): `ls -la ~/.ssh/`

Exercițiul 2.8:
Diferențe /home vs /root:

1. Verificați permisiunile: `ls -ld /home /root`
2. Observați diferența de permisiuni (755 vs 700)
3. Încercați: `ls /root` - aveți acces? De ce nu? (trebuie root)
4. Ca root: `sudo ls -la /root`
5. Comparați conținutul cu home-ul vostru

2.2.4 Directoare device și system info

/dev - Device files

În Linux, **"totul este un fișier"**, inclusiv hardware-ul. `/dev` conține fișiere speciale care reprezintă dispozitivele hardware:

```
# Explorare /dev
ls -l /dev | head -30

# Device files importante:
# /dev/sda, /dev/sdb     → hard disks (SATA/SCSI)
# /dev/sda1, /dev/sda2   → partitions
# /dev/nvme0n1           → NVMe SSD
# /dev/tty*              → terminals
# /dev/null              → "black hole" (discard output)
# /dev/zero              → infinite stream of zeros
# /dev/random            → random data generator
# /dev/loop*             → loop devices (mount ISO)
```

Tipuri device files:

```
# Vezi tipul cu ls -l
ls -l /dev/sda
# Output: brw-rw---- ... /dev/sda
# b = block device (hard disk, USB)

ls -l /dev/tty1
```

```
# Output: crw------- ... /dev/tty1
# c = character device (terminal, serial port)

ls -l /dev/stdin
# Output: lrwxrwxrwx ... /dev/stdin -> /proc/self/fd/0
# l = symbolic link
```

Folosire practică:

```
# Discard output (black hole)
some_noisy_command > /dev/null 2>&1

# Generate random data
head -c 20 /dev/urandom | base64

# Zero-fill a file
dd if=/dev/zero of=bigfile bs=1M count=100
```

/proc - Process and kernel info

Virtual filesystem (nu ocupă spațiu pe disk!) generat de kernel în timp real. Conține informații despre procese și kernel:

```
# Explorare /proc
ls -l /proc | head -20

# Structură:
# /proc/[PID]/    → info despre procesul cu acel PID
# /proc/cpuinfo   → info CPU
# /proc/meminfo   → info RAM
# /proc/version   → kernel version
# /proc/sys/      → kernel parameters (editabile!)
```

Info sistem:

```
# CPU info
cat /proc/cpuinfo | head -30

# Output: processor, vendor_id, cpu MHz, cache size...
# Memory info
cat /proc/meminfo | head -10

# Output:
# MemTotal: 16384000 kB
# MemFree: 8192000 kB
# MemAvailable: 12000000 kB

# Kernel version
cat /proc/version

# Uptime (seconds + idle time)
cat /proc/uptime
```

Info procese:

```
# Info despre un proces specific (ex: PID 1 = init/systemd)
ls /proc/1/

# Output:
# cmdline  → command line used to start
# exe      → symbolic link to executable
# environ  → environment variables
# status   → process status
# fd/      → file descriptors opened
```

/sys - Device and driver info

Similar cu `/proc`, dar mai structurat și dedicat devices/drivers:

```
# Explorare /sys
```

```
ls /sys
# /sys/block/       → block devices (disks)
# /sys/class/net/   → network interfaces
# /sys/devices/     → device hierarchy
# /sys/firmware/    → firmware info
```

/run - Runtime data

Runtime data pentru procese (creat la boot, șters la shutdown):

```
# Explorare /run
ls /run

# PID files   → /run/sshd.pid (process ID pentru servicii)
# Lock files  → /run/lock/
# Sockets     → /run/systemd/ (IPC sockets)
```

Exercițiul 2.9:
Explorare devices:

1. Listați block devices: `ls -l /dev/sd*` sau `ls -l /dev/nvme*`
2. Găsiți device-ul pentru primul hard disk
3. Listați partițiile: dacă aveți `/dev/sda`, listați `/dev/sda1`, `/dev/sda2`, etc.
4. Verificați tipul (block device): `ls -l /dev/sda`
5. Folosiți `lsblk` pentru vizualizare ierarhică: `lsblk`

Exercițiul 2.10:
Explorare /proc:

1. CPU info: `cat /proc/cpuinfo` - câte procesoare aveți?
2. Memorie: `cat /proc/meminfo | grep MemTotal`
3. Kernel version: `cat /proc/version`
4. Uptime: `cat /proc/uptime` (output în secunde)
5. Explorați un proces: `ls /proc/1/` (PID 1 = init)

Exercițiul 2.11:
Virtual filesystems:

1. Verificați dimensiunea /proc: `du -sh /proc 2>/dev/null | head -1`
2. Observați că output-ul e ciudat (virtual filesystem!)
3. Verificați dimensiunea /sys: `du -sh /sys 2>/dev/null | head -1`
4. Rulați `mount | grep proc` - unde e montat /proc?
5. Rulați `mount | grep sys` - unde e montat /sys?

2.2.5 Directoare programe

/usr - User programs (CEL MAI MARE director!)

Conține majoritatea programelor instalate și datele read-only. Este de obicei cel mai mare director din sistem.

Structură /usr:

```
# Explorare /usr
ls /usr

# Subdirectoare importante:
# /usr/bin/        → executabile non-esențiale (majoritatea programelor!)
# /usr/sbin/       → system binaries non-esențiale
# /usr/lib/        → libraries pentru /usr/bin
# /usr/local/      → software instalat manual (de admin, nu din package manager)
# /usr/share/      → date shared (documentație, icoane, man pages)
# /usr/include/    → C/C++ header files
# /usr/src/        → source code (kernel sources)
```

Diferența /bin vs /usr/bin:

Aspect	/bin	/usr/bin
Când se folosesc	Boot process, recovery mode	Normal operation
Necesitate	ESENȚIAL pentru boot	Non-esențial
Exemple	ls, cat, , mount	gcc, python, vim, firefox
Montare	Trebuie disponibil înainte de /usr	Poate fi pe partiție separată

```
# Câte programe sunt în /usr/bin?
ls /usr/bin | wc -l
# Output: ~2000-4000 (depinde de ce e instalat)

# Vs. /bin
ls /bin | wc -l
# Output: ~100-200
```

2.2.5.1 /usr/local - Software instalat manual

Unde instalezi software când **NU folosești package manager**-ul:

```
# Structură /usr/local
ls /usr/local

# /usr/local/bin/    → executabile instalate manual
# /usr/local/lib/    → librării instalate manual
# /usr/local/share/  → date instalate manual
```

De ce există /usr/local?

> Separare clară: pachete din package manager vs. instalări manuale
> La upgrade OS, /usr/local nu e atins
> Admin-ul controlează complet ce e în /usr/local

2.2.5.2 /usr/share - Shared data

Date partajate între toate aplicațiile (architecture-independent):

```
# Explorare /usr/share
ls /usr/share | head -20

# /usr/share/man/      → man pages (documentație)
# /usr/share/doc/      → documentație pachete
# /usr/share/icons/    → icoane aplicații
# /usr/share/pixmaps/  → imagini
# /usr/share/locale/   → translations (internationalization)
```

/opt - Optional/third-party software

Software comercial sau third-party care nu urmează structura standard:

```
# Explorare /opt
ls /opt

# Exemple tipice:
# /opt/google/chrome/   → Google Chrome
# /opt/teamviewer/      → TeamViewer
```

Pattern /opt:

Fiecare vendor are propriul subdirector: `/opt/vendor/app/`
Self-contained (tot ce e nevoie e în subdirectorul respectiv)
Nu se amestecă cu restul sistemului

/srv - Service data

Date servite de sistem (web server, FTP server, etc.):

```
# Structură /srv
ls /srv:
# /srv/www/     → website files (Apache/Nginx)
# /srv/ftp/     → FTP server data
# /srv/tftp/    → TFTP server data (PXE boot)
```

De ce /srv și nu /var/www?

`/srv` = data PE CARE o servești (focus pe conținut)
`/var` = data care VARIAZĂ (focus pe schimbare)
În practică, multe distribuții folosesc `/var/www` din obișnuință

Exercițiul 2.12:
Explorare /usr:

1. Listați conținutul /usr: `ls /usr`
2. Numărați programe în /usr/bin: `ls /usr/bin | wc -l`
3. Găsiți un program cunoscut: `ls /usr/bin/python*` sau `ls /usr/bin/vim*`
4. Verificați dimensiunea /usr: `du -sh /usr`
5. Comparați cu /bin: `du -sh /bin`

Exercițiul 2.13:
Explorare /usr/share/doc (deja cunoscut din Lab 1):

1. Listați pachete documentate: `ls /usr/share/doc/ | head -20`
2. Alegeți un pachet și explorați: `ls /usr/share/doc//`
3. Citiți README: `less /usr/share/doc//README` (dacă există)
4. Găsiți changelog: `ls /usr/share/doc//*changelog*`
5. Căutați exemple: `ls /usr/share/doc//examples/` (dacă există)

Exercițiul 2.14:
Diferențiere /bin vs /usr/bin:

1. Verificați dacă `ls` e în /bin sau /usr/bin: `which ls`
2. Pe sisteme moderne, `/bin` e link către `/usr/bin`: `ls -ld /bin`
3. Verificați un program non-esențial: `which gcc` (probabil /usr/bin)
4. Verificați `mount`: `which mount` (poate fi /bin sau /sbin)

2.2.6 Directoare date variabile

/var - Variable data

Conține date care **cresc în timp** și se modifică constant:

```
# Explorare /var
ls /var

# Subdirectoare importante:
# /var/log/      → LOG FILES (CEL MAI IMPORTANT pentru troubleshooting!)
# /var/cache/    → application cache
# /var/tmp/      → temporary files (NU se șterge la reboot)
# /var/spool/    → queues (print, mail, cron jobs)
# /var/lib/      → state info pentru packages
# /var/www/      → web server files (Debian/Ubuntu)
```

2.2.6.1 /var/log - Log files

CEL MAI IMPORTANT subdirector pentru troubleshooting!

```
# Explorare /var/log
ls -lh /var/log

# Log files importante:
# syslog         → general system log
# auth.log       → authentication attempts
```

```
# kern.log       → kernel messages
# dmesg          → boot messages
# apache2/       → Apache web server logs (Debian)
# httpd/         → Apache web server logs (RHEL)
# nginx/         → Nginx web server logs
# mysql/         → MySQL database logs
```

Vizualizare logs:

```
# Ultimele 20 linii din syslog
tail -20 /var/log/syslog

# Follow log în timp real (ca tail -f)
tail -f /var/log/syslog

# Căutare în logs
grep "error" /var/log/syslog

# Logs mai vechi (compressed)
zcat /var/log/syslog.1.gz | grep "error"
```

Log rotation:

Logs-urile cresc constant
`logrotate` le rotește automat (zilnic/săptămânal)
Pattern: `syslog`, `syslog.1`, `syslog.2.gz`, `syslog.3.gz`

2.2.6.2 /var/cache - Application cache

Cache pentru aplicații (poate fi șters fără probleme):

```
ls /var/cache

# Exemple:
# apt/           → package cache (Debian)
# yum/           → package cache (RHEL)
```

2.2.6.3 /var/spool - Queues

Cozi pentru diverse servicii:

```
ls /var/spool

# Subdirectoare:
# cron/        → cron jobs în așteptare
# mail/        → mail queue
# cups/        → print queue
```

/tmp - Temporary files

Fișiere temporare care **se șterg la reboot**:

```
# Explorare /tmp
ls /tmp

# Caracteristici:
# - World-writable (oricine poate scrie)
# - Se șterge automat la reboot
# - Uneori montat ca tmpfs (în RAM pentru performanță)
```

Diferența /tmp vs /var/tmp:

`/tmp` → se șterge la reboot
`/var/tmp` → NU se șterge la reboot (persistent între reboot-uri)

```
# Verificare dacă /tmp e în RAM
mount | grep /tmp

# Output dacă e tmpfs:
# tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
```

Exercițiul 2.15:
Explorare /var/log (CEL MAI IMPORTANT):

1. Listați /var/log: `ls -lh /var/log`
2. Găsiți ultimele 20 linii din syslog: `tail -20 /var/log/syslog`
3. Căutați cuvântul "error": `grep -i error /var/log/syslog | head -10`
4. Verificați logs vechi comprimate: `ls /var/log/*.gz`
5. Decomprimați și căutați: `zcat /var/log/syslog.1.gz | grep -i failed | head -5`

Exercițiul 2.16:
Verificare spațiu /var:

1. Verificați dimensiunea /var: `du -sh /var`
2. Verificați doar /var/log: `du -sh /var/log`
3. Găsiți cele mai mari 10 fișiere din /var/log:

```
du -ah /var/log 2>/dev/null | sort -rh | head -10
```

4. Verificați spațiu disponibil pe partiția /var: `df -h /var`

Exercițiul 2.17:
Explorare /tmp:

1. Listați /tmp: `ls -la /tmp`
2. Verificați permisiunile: `ls -ld /tmp` (trebuie să fie drwxrwxrwt - sticky bit!)
3. Verificați dacă e tmpfs (în RAM): `mount | grep /tmp`
4. Creați un fișier test: `touch /tmp/test_file_$(whoami).txt`
5. Verificați dacă există: `ls -l /tmp/test_file_*`

2.2.7 Mount points

/media și /mnt - Puncte de montare

Locuri unde se montează dispozitive externe:

/media - Auto-mount removable media:

```
# Când inserezi un USB stick
ls /media/$USER
# Output:
# /media/username/USB_DRIVE/
```

Folosit de sistemul de auto-mount (udisks2) pentru:

- USB flash drives
- CD/DVD
- SD cards
- External hard drives

/mnt - Manual mount points:

```
# Creat pentru mount-uri temporare manuale
ls /mnt

mkdir /mnt/backup
mount /dev/sdb1 /mnt/backup
# ... folosești ...
umount /mnt/backup
```

Folosit de administratori pentru mount-uri temporare:

- Testing filesystem-uri noi
- Recovery operations
- Temporary network shares
- Backup drives

Diferența /media vs /mnt:

- `/media` - mount automat de sistem pentru removable media
- `/mnt` - mount manual de administrator pentru orice

Exercițiul 2.18:
Explorare mount points:

1. Verificați ce e montat în /media: `ls /media/` și `ls /media/$USER/`
2. Verificați /mnt: `ls /mnt`
3. Vedeți toate filesystem-urile montate: `mount | grep "^/dev"`
4. Sau mai clar: `df -h`
5. Verificați ce tip de filesystem sunt: `mount | grep ext4` sau `mount | grep ntfs`

2.3 Comenzi de navigare în sistem

Acum că înțelegem structura, să vedem cum navigăm eficient:

2.3.1 pwd - Present Working Directory

Afișează directorul curent în care te afli:

```
pwd
# Output: /home/username/documents
cd /var/log
pwd
# Output: /var/log
```

Când e util pwd:

Scripts (pentru a verifica unde ești)
După `cd -` (pentru a confirma unde ai ajuns)
În combinație cu alte comenzi: `ls $(pwd)`

2.3.2 cd - Change Directory

Schimbă directorul curent:

```
# Absolute path
cd /var/log

# Relative path
cd ../cache       # urcă un nivel, apoi intră în cache
```

```
# Home directory
cd ~              # SAU doar: cd
# Output: /home/username

# Directorul anterior
cd -
# Toggle între ultimele 2 directoare

# Director părinte
cd ..

# Rămâi în directorul curent (folosit în scripts)
cd .
```

Paths - absolute vs relative:

```
# ABSOLUTE path (începe cu /)
cd /etc/ssh
cd /var/log/apache2

# RELATIVE path (nu începe cu /)
cd documents      # relativ la directorul curent
cd ../bin         # urcă un nivel, apoi intră în bin
cd ../../etc      # urcă două nivele, apoi intră în etc
```

Tips & Tricks cd:

```
# Quick toggle între două directoare
cd /var/log
cd /etc
cd -              # revine la /var/log
cd -              # revine la /etc

# Cd în subdirector lung
cd /var/log
```

```
# Environment variable
cd $HOME        # echivalent cd ~

# Spații în nume (escape sau quotes)
cd My\ Documents/
cd "My Documents/"
```

2.3.3 ls - List directory contents

Lista conținut directoare:

```
# Basic listing
ls

# Long listing (detailed)
ls -l

# Include hidden files (dotfiles)
ls -a

# Long + hidden
ls -la
# Human-readable sizes
ls -lh

# Sort by modification time (newest first)
ls -lt

# Reverse sort
ls -ltr        # oldest first

# Doar directoare
ls -ld */
# Recursiv (toate subdirectoarele)
ls -R
# Listare cu inode numbers
ls -li
```

Interpretare ls -l output:

```
ls -l /bin/ls
# Output: -rwxr-xr-x 1 root root 142144 Sep 5 2023 /bin/ls
#         | | | | | | | | | | |  |   |    |    |        |
#         | | | | | | | | | | |  |   |    |    |        └── filename
#         | | | | | | | | | | |  |   |    |    └── modification date
#         | | | | | | | | | | |  |   |    └── size (bytes)
#         | | | | | | | | | | |  |   └── group
#         | | | | | | | | | | |  └── owner
#         | | | | | | | | | | └── hard links count
#         | | | | | | | | | └── others execute
#         | | | | | | | | └── others write
#         | | | | | | | └── others read
#         | | | | | | └── group execute
#         | | | | | └── group write
#         | | | | └── group read
#         | | | └── owner execute
#         | | └── owner write
#         | └── owner read
#         └── type: - (file), d (directory), l (link), b (block), c (char)
```

Culori ls (când --color=auto):

Albastru - directoare
Cyan - symbolic links
Verde - executabile
Roșu - arhive (.tar, .gz, .zip)
Magenta - imagini/media
Galben - device files

Tips & Tricks ls:

```
# Doar fișierele .conf din /etc
ls /etc/*.conf
```

```
# Doar directoarele (trailing slash)
ls -d */

# Sortare după dimensiune (cel mai mare primul)
ls -lhS

# Cea mai recentă modificare
ls -lt | head -1

# Număr total fișiere
ls | wc -l

# Ascunde backup files (~)
ls --hide='*~'

# Cu full path
ls -d $PWD/*
```

2.3.4 tree - Hierarchical view

Afișează structura ca arbore (trebuie instalat: `apt install tree` sau `yum install tree`):

```
# Nivel 1
tree -L 1 /

# Nivel 2
tree -L 2 /var

# Doar directoare
tree -d -L 2 /etc

# Cu dimensiuni
tree -h -L 2 /var/log

# Cu permisiuni
tree -p -L 1 /home
```

```
# Ignoră anumite directoare
tree -I 'node_modules|.git' -L 3

# Output în fișier
tree -L 2 /etc > etc_structure.txt
```

Tips & Tricks tree:

```
# Combinații utile:
# -L N      → depth limit
# -d        → directories only
# -h        → human-readable sizes
# -p        → permissions
# -u        → username
# -D        → last modification time
# -I pattern → ignore pattern
# Overview filesystem
tree -L 1 -d /

# Structure /var cu dimensiuni
tree -L 2 -h -d /var

# Documentație pachet
tree -L 3 /usr/share/doc/

# Configurare apache
tree -L 2 /etc/apache2
```

Exercițiul 2.19:
Practică navigare:

1. Verificați unde sunteți: `pwd`
2. Mergeți la root: `cd /`
3. Verificați: `pwd` (ar trebui să fie `/`)
4. Mergeți la /var/log: `cd /var/log`
5. Reveniți la directorul anterior: `cd -`
6. Mergeți home: `cd` sau `cd ~`
7. Verificați: `pwd` (ar trebui `/home/username`)

Exercițiul 2.20:
Practică paths (absolute vs relative):

1. Din home (`cd ~`), mergeți la /etc folosind absolute path: `cd /etc`
2. Din /etc, mergeți la /var folosind relative path: `cd ../var`
3. Din /var, mergeți la /var/log folosind relative path: `cd log`
4. Urcați un nivel: `cd ..`
5. Verificați: `pwd` (ar trebui `/var`)

Exercițiul 2.21:
Practică ls cu opțiuni:

1. Listați /etc: `ls /etc`
2. Long listing: `ls -l /etc`
3. Include hidden: `ls -la /etc`
4. Sortare după timp: `ls -lt /etc | head -20`
5. Doar directoare: `ls -ld /etc/*/`
6. Cele mai mari fișiere: `ls -lhS /var/log | head -10`

Exercițiul 2.22:
Vizualizare cu tree:

1. Verificați dacă tree e instalat: `which tree`
2. Dacă nu: `sudo apt install tree` (Debian) sau `sudo yum install tree` (RHEL)
3. Vizualizați root nivel 1: `tree -L 1 /`
4. Vizualizați /var nivel 2: `tree -L 2 /var`
5. Doar directoare /etc nivel 2: `tree -d -L 2 /etc`
6. Cu dimensiuni: `tree -h -L 2 /var/log`

2.4 Montarea sistemelor de fișiere

2.4.1 Conceptul de mounting

În Linux, pentru a accesa conținutul unui dispozitiv de stocare (hard disk, USB, CD, network share), trebuie să-l **montezi** într-un director existent din ierarhie.

Mounting înseamnă:

Asocierea unui filesystem (ex: /dev/sda1) cu un director (ex: /mnt/backup)
După montare, accesezi conținutul prin acel director
Unmounting = deconectare sigură

Analogie:

Windows: Inserezi USB → apare drive D:
Linux: Inserezi USB → trebuie montat în `/media/username/USB_DRIVE/`

```
# Înainte de mount
ls /mnt/backup
# Output: directory gol

# După mount /dev/sdb1 în /mnt/backup
mount /dev/sdb1 /mnt/backup
ls /mnt/backup
# Output: conținutul de pe /dev/sdb1
```

De ce e nevoie de mounting:

Flexibilitate: Montezi unde vrei în ierarhie
Securitate: Control complet asupra access-ului
Multi-dispozitiv: Mai multe dispozitive în același arbore
Network shares: NFS, CIFS montate transparent

2.4.2 Comanda mount

Sintaxă:

```
mount [opțiuni] <device> <mount_point>

# Exemple:
mount /dev/sdb1 /mnt/backup
mount -t ext4 /dev/sdc1 /mnt/data
mount -o ro /dev/sdd1 /mnt/readonly
```

Verificare ce e montat:

```
# Toate filesystem-urile montate
mount

# Sau doar device-uri fizice
mount | grep "^/dev"

# Sau cu df (disk free)
df -h

# Sau cu lsblk (block devices)
lsblk
```

Mount options importante:

```
# Read-only
mount -o ro /dev/sdb1 /mnt/backup

# Read-write (default)
mount -o rw /dev/sdb1 /mnt/backup
```

```
# No execution permission (securitate)
mount -o noexec /dev/sdb1 /mnt/untrusted

# No setuid (securitate)
mount -o nosuid /dev/sdb1 /mnt/untrusted

# User permissions
mount -o user /dev/sdb1 /mnt/usb
```

Mount network shares:

```
# NFS (Network File System)
mount -t nfs server:/export/path /mnt/nfs

# CIFS (Windows share)
mount -t cifs //server/share /mnt/windows -o username=user,password=pass

# Better: cu credentials file
mount -t cifs //server/share /mnt/windows -o credentials=/root/.smbcredentials
```

Tips & Tricks mount:

```
# Mount cu label (în loc de /dev/sdb1)
mount LABEL=backup /mnt/backup

# Mount cu UUID (mai stabil)
mount UUID=abc123-def456 /mnt/backup

# Găsire UUID
blkid /dev/sdb1

# Mount ISO file (loop device)
mount -o loop backup.iso /mnt/iso

# Mount toate din /etc/fstab
mount -a
```

```
# Remount cu alte opțiuni (fără umount)
mount -o remount,ro /mnt/backup
```

2.4.3 Comanda umount

Sintaxă:

```
umount <mount_point>
# SAU
umount <device>

# Exemple:
umount /mnt/backup
umount /dev/sdb1
```

Erori comune și soluții:

```
# Eroare: device is busy
umount /mnt/backup
# umount: /mnt/backup: target is busy

# Verificare ce proces folosește
lsof /mnt/backup
# SAU
fuser -m /mnt/backup

# Kill procesele care folosesc
fuser -km /mnt/backup

# Lazy umount (așteaptă ca procesele să termine)
umount -l /mnt/backup

# Force umount (periculos, poate pierde date!)
umount -f /mnt/backup
```

Best practices umount:

```
# 1. Verifică că nu ești în directorul montat
pwd
# Dacă output e /mnt/backup, faci: cd ~

# 2. Verifică că nu rulează procese
lsof /mnt/backup

# 3. Sync (force write la disk)
sync

# 4. Umount
umount /mnt/backup

# 5. Verifică
mount | grep /mnt/backup
# Ar trebui să nu mai apară
```

2.4.4 /etc/fstab - Configurare mount permanent

Fișierul `/etc/fstab` (filesystem table) conține filesystem-uri care trebuie montate automat la boot.

Structură /etc/fstab:

```
cat /etc/fstab

# Format:
# <device> <mount_point> <type> <options> <dump> <pass>

# Exemple:
UUID=abc123-def  /           ext4  defaults      0  1
UUID=def456-ghi  /home       ext4  defaults      0  2
UUID=jkl789-mno  swap        swap  defaults      0  0
/dev/sdb1        /mnt/backup ext4  defaults      0  2
//server/share   /mnt/windows cifs credentials=... 0  0
```

Câmpurile fstab:

1. **Device** - UUID, LABEL, sau /dev/sdXN
2. **Mount point** - unde se montează
3. **Filesystem type** - ext4, ntfs, nfs, cifs, swap
4. **Options** - defaults, ro, noexec, etc.
5. **Dump** - backup cu `dump` (0=nu, 1=da) - deprecated
6. **Pass** - fsck check order (0=skip, 1=root, 2=altele)

Opțiuni comune:

```
defaults    → rw, suid, dev, exec, auto, nouser, async
ro          → read-only
rw          → read-write
noexec      → no execution permission
nosuid      → no setuid
noauto      → nu monta automat la boot (mount manual)
user        → users non-root pot monta
nofail      → continuă boot chiar dacă mount eșuează (important!)
```

Editare /etc/fstab:

```
# IMPORTANT: Backup înainte!
sudo cp /etc/fstab /etc/fstab.backup

# Editare
sudo nano /etc/fstab
# SAU
sudo vim /etc/fstab

# Adaugă linie nouă:
UUID=abc123-def /mnt/backup ext4 defaults,nofail 0 2

# Test fără reboot
sudo mount -a
# Verificare
mount | grep /mnt/backup

# Dacă e OK, la următorul reboot se va monta automat
```

Găsire UUID pentru fstab:

```
# Toate device-urile cu UUID
blkid

# Output:
# /dev/sda1: UUID="abc123-def456" TYPE="ext4" LABEL="root"
# /dev/sdb1: UUID="ghi789-jkl012" TYPE="ext4" LABEL="backup"

# Doar pentru un device
blkid /dev/sdb1

# Sau cu lsblk
lsblk -o NAME,UUID,FSTYPE,SIZE,MOUNTPOINT
```

Exemple fstab pentru diverse scenarii:

```
# USB drive automat (cu noauto pentru mount manual)
UUID=xxx-yyy /mnt/usb ext4 noauto,user,rw 0 0

# Network share cu nofail (să nu blocheze boot-ul)
//server/share /mnt/smb cifs credentials=/root/.smbcredentials,nofail 0 0

# Readonly partition
UUID=zzz-www /mnt/readonly ext4 ro,nofail 0 0

# Swap partition
UUID=aaa-bbb none swap sw 0 0
```

Exercițiul 2.23:
Explorare mount:

1. Vedeți toate filesystem-urile montate: `mount | grep "^/dev"`
2. Sau mai clar: `df -h`
3. Verificați ce tip de filesystem are root: `mount | grep " / "`
4. Listați block devices: `lsblk`
5. Găsiți UUID-ul root filesystem: `blkid | grep sda1` (sau nvme0n1p1)

Exercițiul 2.24:
Mount manual practice (OPȚIONAL - doar dacă aveți USB):

1. Creați un mount point: `sudo mkdir -p /mnt/test`
2. Dacă aveți un USB stick, găsiți-i device-ul: `lsblk`
3. Montați (ex: /dev/sdb1): `sudo mount /dev/sdb1 /mnt/test`
4. Verificați: `mount | grep /mnt/test`
5. Listați conținut: `ls /mnt/test`
6. Unmontați: `sudo umount /mnt/test`
7. Verificați: `mount | grep /mnt/test` (nu ar trebui să mai apară)

Exercițiul 2.25:
Explorare /etc/fstab:

1. Citiți /etc/fstab: `cat /etc/fstab`
2. Identificați root filesystem (mount point `/`)
3. Observați UUID-ul sau LABEL-ul
4. Verificați opțiunile (defaults?)
5. Găsiți swap partition (dacă există)
6. Comparați cu `mount` output - sunt toate din fstab montate?

2.5 Căutarea și localizarea fișierelor

După ce înțelegem structura, trebuie să găsim rapid fișiere în sistem. Avem trei tool-uri principale:

2.5.1 locate - Căutare rapidă în database

`locate` caută fișiere într-o database pre-construită, actualizată nightly. Este extrem de rapid pentru că nu caută live pe disk.

Sintaxă:

```
locate pattern

# Exemple:
locate nginx.conf
locate "*.log"
locate -i password    # case-insensitive
```

Caracteristici locate:

Avantaj: FOARTE rapid (seconds chiar pe TB de date)
Dezavantaj: Database update nightly - fișiere noi de azi NU apar!

Opțiuni utile:

```
# Count matches (doar număr, nu lista)
locate -c nginx.conf

# Exact basename (nu substring)
locate -b '\nginx.conf'

# Case-insensitive
locate -i CONFIG.CONF

# Limit results
locate nginx.conf | head -20

# Show only existing files (skip deleted)
locate -e nginx.conf
```

Update database manual:

```
# Database se update automat nightly, dar poți forța:
sudo updatedb

# Database location:
# /var/lib/mlocate/mlocate.db (modern)
# /var/lib/locate/locatedb (vechi)
```

Use cases locate:

```
# Găsire config files
locate nginx.conf
# Output: toate path-urile care conțin "nginx.conf"

# Găsire toate configs pentru apache
locate apache2.conf
locate httpd.conf    # RHEL
```

```
# Security audit - fișiere sensibile
locate -i password
locate .ssh
locate id_rsa

# Câte log files există?
locate -c "*.log"

# Documentație pentru un pachet
locate -i readme | grep nginx
```

Limitări locate:

- Fișiere create astăzi **NU apar** până mâine (database update nightly)
- Pe sisteme fresh install, database poate fi goală - rulează `updatedb` prima dată
- Nu găsește fișiere șterse recent (până la următorul updatedb)
- Nu poți filtra după size, time, permissions

Când folosești locate:

- Știi numele fișierului sau pattern
- Fișierul există de cel puțin 1 zi
- Vrei rezultate instant (seconds)
- NU pentru fișiere create astăzi
- NU pentru căutări complexe (size + time + permissions)

2.5.2 find - Căutare live pe filesystem

`find` caută direct pe filesystem (real-time). Este mai lent, dar vede TOT și poate filtra după criterii complexe.

Sintaxă de bază:

```
find <path> <criteria> <action>

# Exemple simple:
find /etc -name "*.conf"
find /var/log -size +100M
find /home -mtime -7
```

Căutare după nume:

```
# Exact name
find /etc -name "hosts"

# Pattern (wildcard)
find /etc -name "*.conf"

# Case-insensitive
find /etc -iname "*.CONF"

# Exclude pattern
find /etc -name "*.conf" ! -name "*.bak"
```

Căutare după tip:

```
# Doar fișiere
find /etc -type f

# Doar directoare
find /etc -type d

# Symlinks
find /etc -type l

# Block devices
find /dev -type b

# Character devices
find /dev -type c
```

Căutare după dimensiune:

```
# Mai mari de 100MB
find /var/log -size +100M

# Mai mici de 1KB
find /tmp -size -1k

# Exact 512 bytes
find . -size 512c

# Unitățile:
# c = bytes
# k = kilobytes
# M = megabytes
# G = gigabytes
```

Căutare după timp:

```
# Modificate în ultimele 7 zile
find /var/log -mtime -7

# Modificate acum mai mult de 30 zile
find /var/log -mtime +30

# Accesate în ultimele 24 ore
find /tmp -atime -1

# Status changed în ultimele 2 ore
find /etc -cmin -120

# Timp units:
# -mtime N  → N days (modification)
# -atime N  → N days (access)
# -ctime N  → N days (status change)
# -mmin N   → N minutes (modification)
```

```
# -amin N   → N minutes (access)
# -cmin N   → N minutes (status change)
```

Căutare după permissions:

```
# Exact permissions 755
find /usr/bin -perm 755

# At least these permissions (user can write)
find /home -perm -u+w

# Any of these permissions (world-writable)
find /tmp -perm /o+w

# SUID files (security risk!)
find / -perm /4000 2>/dev/null

# SGID files
find / -perm /2000 2>/dev/null
```

Căutare după owner:

```
# Owner by name
find /home -user john

# Owner by UID
find /home -uid 1000

# Group by name
find /var -group www-data

# Files without valid owner (security risk!)
find / -nouser 2>/dev/null
```

Combinare criterii (AND și OR):

```
# AND implicit (spațiu între criterii)
find /var/log -name "*.log" -size +50M -mtime +30
# Găsește: logs ȘI mari (50MB+) ȘI vechi (30+ zile)

# AND explicit (-a sau -and)
find /var/log -name "*.log" -a -size +50M

# OR (-o sau -or)
find /var/log -name "*.log" -o -name "*.txt"

# NOT (!)
find /etc -name "*.conf" ! -name "*.bak"

# Grupare cu paranteză (escape cu \)
find /var/log \( -name "*.log" -o -name "*.txt" \) -size +10M
```

Acțiuni pe rezultate:

```
# Print (default)
find /etc -name "*.conf"

# Print explicit
find /etc -name "*.conf" -print

# Execute command pe fiecare rezultat
find /var/log -name "*.log" -exec gzip {} \;
# {} = placeholder pentru fișier
# \; = end of command

# Execute cu confirm
find /tmp -mtime +7 -ok rm {} \;
# Întreabă înainte de fiecare delete

# Execute batch (efficient pentru multe fișiere)
find /var/log -name "*.log" -exec gzip {} +
# + = execută cu multiple args
```

```
# Delete direct
find /tmp -mtime +7 -delete
# ATENȚIE: test fără -delete mai întâi!

# Print cu detalii (ca ls -l)
find /etc -name "*.conf" -ls
```

Tips & Tricks find:

```
# Limitare depth (max 2 nivele)
find /etc -maxdepth 2 -name "*.conf"

# Minimum depth (skip directorul curent)
find /var -mindepth 2 -name "*.log"

# Follow symlinks
find /usr -follow -name "python"

# Ignore errors (redirect stderr)
find / -name "*.conf" 2>/dev/null

# Găsire duplicate files (by name)
find / -name "config.txt" 2>/dev/null

# Găsire empty files/directories
find /tmp -empty

# Găsire executabile
find /usr/bin -type f -executable

# Găsire world-writable (security risk!)
find / -perm -002 -type f 2>/dev/null
```

Use cases telecom:

```
# Security audit - SUID binaries
find / -perm /4000 -type f 2>/dev/null

# Disk cleanup - logs vechi și mari
find /var/log -name "*.log" -size +500M -mtime +90

# Config orphans - backup files vechi
find /etc -name "*backup*" -mtime +180

# Recent changes - troubleshooting
find /etc -mtime -1 -type f

# Large files consuming disk
find /var -size +1G -type f 2>/dev/null

# Delete old temp files
find /tmp -mtime +7 -delete

# Compress old logs
find /var/log -name "*.log" -mtime +30 -exec gzip {} \;

# Find broken symlinks
find /usr -xtype l 2>/dev/null
```

Exercițiul 2.27:
Practică locate:

1. Găsiți toate fișierele care conțin "nginx" în nume: `locate nginx`
2. Câte fișiere .conf există: `locate -c "*.conf"`
3. Găsiți case-insensitive "README": `locate -i readme | head -20`
4. Update database: `sudo updatedb`
5. Căutați din nou după update

Exercițiul 2.28:
Practică find - căutare simplă:

1. Găsiți toate fișierele .conf din /etc: `find /etc -name "*.conf" 2>/dev/null`
2. Doar primele 20: `find /etc -name "*.conf" 2>/dev/null | head -20`
3. Câte sunt: `find /etc -name "*.conf" 2>/dev/null | wc -l`
4. Doar directoare din /var: `find /var -maxdepth 2 -type d 2>/dev/null`

Exercițiul 2.29:
Practică find - criterii complexe:

1. Fișiere mai mari de 100MB din /var: `find /var -size +100M -type f 2>/dev/null`
2. Fișiere modificate în ultimele 7 zile din /etc: `find /etc -mtime -7 -type f 2>/dev/null`
3. Logs vechi ȘI mari: `find /var/log -name "*.log" -size +50M -mtime +30 2>/dev/null`
4. World-writable files (security): `find /tmp -perm -002 -type f 2>/dev/null`

Exercițiul 2.30:
Challenge find:

1. Găsiți toate fișierele .log din /var care:
   Sunt mai mari de 10MB
   Au fost modificate în ultimele 30 zile
   Nu sunt comprimate (.gz)

```
find /var/log -name "*.log" -size +10M -mtime -30 ! -name "*.gz" 2>/dev/null
```

2. Numărați câte sunt: `... | wc -l`
3. Listați cele mai mari 5: `... -exec ls -lh {} \; | sort -k5 -rh | head -5`

2.5.3 whereis - Localizare comenzi (recap)

Am învățat `whereis` în Laboratorul 1, dar iată un recap rapid:

```
# Găsește executabil + man pages + source
whereis ls
# Output: ls: /bin/ls /usr/share/man/man1/ls.1.gz

# Doar binary
whereis -b python

# Doar man pages
whereis -m
```

Diferențe whereis vs which vs locate vs find:

Tool	Ce găsește	Viteză	Use case
`whereis`	Binary + man + source	Rapid	Verificare dacă comandă instalată
`which`	Primul exec din $PATH	Instant	Ce versiune rulează
`locate`	Orice fișier (database)	Foarte rapid	Găsire rapidă nume cunoscut
`find`	Orice (live search)	Lent	Căutări complexe real-time

Când folosești ce:

```
# Verificare dacă nginx instalat?
whereis nginx

# Ce versiune python rulează?
which python

# Unde sunt TOATE configs nginx?
locate nginx.conf

# Găsește configs modificate azi
find /etc -name "*.conf" -mtime 0
```

Exercițiul 2.31:
Challenge - alegere tool corect:

Pentru fiecare scenariu, alegeți tool-ul optim (whereis/which/locate/find) și scrieți comanda:

1. Verificare dacă `rsyslog` este instalat
2. Găsire toate fișierele care conțin "apache" în nume
3. Găsire config files modificate în ultimele 24 ore
4. Verificare ce versiune Python rulează când tastezi `python`
5. Găsire fișiere mai mari de 1GB din /var
6. Găsire rapidă a fișierului README.md pentru nginx
7. Găsire SUID binaries (security audit)

Soluții:

1. `whereis rsyslog`
2. `locate apache`
3. `find /etc -name "*.conf" -mtime 0 2>/dev/null`
4. `which python`
5. `find /var -size +1G -type f 2>/dev/null`
6. `locate nginx | grep README`
7. `find / -perm /4000 2>/dev/null`

2.4 Quick Reference Tables

Comenzi navigare:

Comandă	Acțiune	Exemplu
`pwd`	Afișează director curent	`pwd`
`cd /path`	Schimbă director (absolute)	`cd /var/log`
`cd path`	Schimbă director (relative)	`cd ../etc`
`cd ~` sau `cd`	Mergi la HOME	`cd ~`
`cd -`	Director anterior	`cd -`
`cd ..`	Director părinte	`cd ..`
`ls -la`	Lista detailed + hidden	`ls -la /etc`
`ls -lhS`	Lista sortată după dimensiune	`ls -lhS /var/log`
`tree -L 2`	Vizualizare arbore nivel 2	`tree -L 2 /etc`

Mount/Umount:

Comandă	Acțiune	Exemplu
`mount`	Vezi toate montate	`mount | grep ^/dev`
`mount device dir`	Montează device	`mount /dev/sdb1 /mnt/usb`
`mount -a`	Montează toate din fstab	`sudo mount -a`
`umount dir`	Unmontează	`umount /mnt/usb`
`blkid`	Vezi UUID devices	`blkid`
`lsblk`	Lista block devices	`lsblk`
`df -h`	Disk usage	`df -h`

Căutare fișiere:

Tool	Când folosești	Comandă exemplu
`locate`	Nume cunoscut, fișier vechi 1+ zile	`locate nginx.conf`
`find`	Criterii complexe, real-time	`find /var -size +100M -mtime +30`
`whereis`	Verificare comandă instalată	`whereis python`
`which`	Versiune comandă din PATH	`which python`

Directoare importante:

Director	Conținut	Use case principal
`/etc`	Config files	Modificare configurări
`/var/log`	Log files	Troubleshooting
`/home`	User data	Backup utilizatori
`/usr/bin`	Programs	Majoritatea comenzilor
`/dev`	Devices	Hardware access
`/proc`	Process/kernel info	System monitoring
`/tmp`	Temp files	Storage temporar
`/boot`	Kernel și boot	Boot management
`/usr/share/doc`	Documentație	Examples și README
