Utilizarea sistemelor de operare (pe scurt)

PERMISIUNI

Permisiunile fișierelor în Linux sunt gestionate printr-un sistem bazat pe trei seturi de permisiuni:

- **Utilizatorul (Owner)** – persoana care deține fișierul
- **Grupul (Group)** – un grup de utilizatori care pot accesa fișierul
- **Alții (Others)** – toți ceilalți utilizatori

Fiecare set de permisiuni poate include:

- `r` (read) – citire
- `w` (write) – scriere/modificare
- `x` (execute) – executare (necesar pentru scripturi și fișiere binare)

Acestea ocupa memoria astfel:

R	4	100
W	2	010
X	1	001

Putem vedea in poza de mai jos un exemplu:

- simbolul `-` indica faptul ca este un fisier obisnuit
- `rwx` **(Owner)** → `r` (4) + `w` (2) + `x` (1) = **7**
- `rwx` **(Group)** → `r` (4) + `w` (2) + `x` (1) = **7**
- `rwx` **(Others)** → `r` (4) + `w` (2) + `x` (1) = **7**

Binar este:

```
111 111 111
```

Cum transformam in binar? (folosim 7 ca exemplu)

- **Descompunerea în puteri ale lui 2**

Știm că numerele în binar se bazează pe puteri ale lui 2:

7=4+2+1= 2^2 + 2^1 + 2^0

- **Scriem în binar**

Fiecare putere a lui 2 este reprezentată prin `1`, iar celelalte poziții prin `0`:

2^2 (4)	2^1 (2)	2^0 (1)
1	1	1

Astfel, **7 în binar este** `111`.

Tipuri de fișiere

Simbol	Tip fișier	Descriere
`-`	Fișier obișnuit	Fișier normal (ex: fișiere text, imagini, etc.)
`d`	Director (Folder)	Director de fișiere
`l`	Legătură simbolică (symlink)	Legătură către un alt fișier sau director
`c`	Fișier de caracter	Fișier special pentru dispozitive de caracter (ex: `/dev/tty`)
`b`	Fișier de bloc	Fișier special pentru dispozitive de bloc (ex: hard disk-uri)
`p`	FIFO (named pipe)	Fișier pentru comunicații inter-proces (pipe)
`s`	Sockets	Fișier pentru comunicații de rețea

Atribut	**Reprezentare octală**	**Reprezentare simbolică**	**Descriere**	**Exemplu de setare**	**Exemplu de verificare**
setuid (SUID)	`4` (e.g., `4755`)	`u+s`	Fișierul rulează cu drepturile proprietarului, nu ale utilizatorului care-l execută.	`chmod u+s /usr/bin/passwd` sau `chmod 4755 /usr/bin/passwd`	`ls -l /usr/bin/passwd` (`-rwsr-xr-x`)
setgid (SGID)	`2` (e.g., `2755`)	`g+s`	**Fișier:** Rulează cu drepturile grupului proprietar.**Director:** Fișierele create moștenesc grupul directorului.	`chmod g+s /usr/bin/write` sau `chmod 2755 /shared_dir`	`ls -ld /shared_dir` (`drwxr-sr-x`)
sticky bit	`1` (e.g., `1777`)	`o+t`	Fișierele dintr-un director pot fi șterse doar de proprietar sau `root`.	`chmod o+t /tmp` sau `chmod 1777 /tmp`	`ls -ld /tmp` (`drwxrwxrwt`)
setuid + setgid	`6` (e.g., `6755`)	`u+s,g+s`	Combină SUID și SGID.	`chmod 6755 /usr/bin/special`	`ls -l /usr/bin/special` (`-rwsr-sr-x`)
setgid + sticky bit	`3` (e.g., `3777`)	`g+s,o+t`	Combină SGID și sticky bit.	`chmod 3777 /shared_restricted`	`ls -ld /shared_restricted` (`drwxrwsrwt`)

Atribut	**Valoare octală**	**Reprezentare simbolică**	**Efect**
setuid	`4`	`u+s`	Fișierul rulează cu drepturile **proprietarului**, nu ale utilizatorului care îl execută.
setgid	`2`	`g+s`	Fișierul rulează cu drepturile **grupului**, sau fișierele moștenesc grupul directorului.
sticky bit	`1`	`o+t`	Fișierele într-un director pot fi șterse doar de **proprietar** sau `root`.

VARIABLE SPECIALE IN SHELL

Variabile pentru Starea Execuției

$? – Codul de ieșire al ultimei comenzi

$? returnează codul de ieșire al ultimei comenzi executate:
- 0: comandă executată cu succes
- altă valoare: eroare în execuție

Exemplu:

```
ls non_existent_file
echo $?  # Va afișa 1, deoarece fișierul nu există
```

Variabile pentru Procese

- $$ – ID-ul procesului curent (PID)

```
echo $$  # Afișează PID-ul shell-ului curent
```

- $! – PID-ul ultimului proces din fundal

```
sleep 100 &
echo $!  # Afișează PID-ul procesului "sleep"
```

Variabile pentru Argumente

- $0 – Numele scriptului/comenzii curente
- $1, $2, ... – Argumentele individuale

```
# Exemplu de script:
echo "Primul argument: $1"
echo "Al doilea argument: $2"

# Rulare:
./script.sh arg1 arg2
```

Colecții de Argumente

- $@ – Lista completă de argumente (păstrează argumentele separate)
- $* – Lista completă de argumente (concatenează argumentele într-un singur șir)

```
# Diferența dintre $@ și $*:
echo "$@"  # Afișează argumentele separate
echo "$*"  # Afișează argumentele ca un singur șir
```

GLOBBING

Caractere de bază

Expresie	Semnificație	Exemplu	Se potrivește cu
`*`	Se potrivește cu orice secvență de caractere (inclusiv vidă)	`*.txt`	`file.txt`, `test.txt`, dar NU `file.txt.bak`
`?`	Se potrivește cu un singur caracter	`file?.txt`	`file1.txt`, `fileA.txt`, dar NU `file10.txt`
`[abc]`	Se potrivește cu ORICE caracter din listă	`file[12].txt`	`file1.txt`, `file2.txt`, dar NU `file3.txt`
`[a-z]`	Se potrivește cu un caracter dintr-un interval	`file[a-c].txt`	`filea.txt`, `fileb.txt`, `filec.txt`, dar NU `filed.txt`
`[^abc]`	Se potrivește cu ORICE caracter care NU este în listă	`file[^12].txt`	`file3.txt`, `fileA.txt`, dar NU `file1.txt`

Exemple practice

Comandă	Explicație

`ls *.sh`	Listează toate fișierele care se termină cu `.sh`
`rm file?.txt`	Șterge `file1.txt`, `fileA.txt`, dar NU `file10.txt`
`ls [a-d]*`	Listează toate fișierele care încep cu `a`, `b`, `c` sau `d`
`ls !(backup)*`	Listează toate fișierele care **nu** încep cu `backup`
`cp file[!0-9].txt /backup/`	Copiază fișierele care **nu** se termină cu cifre în `/backup/`

CUM CALCULAM CATE IP-URI SUNT?

Pentru a calcula numărul de IP-uri disponibile într-o rețea, folosim următoarea formulă:

```
2^(32-prefix) = număr total de IP-uri

Exemplu pentru /24:
2^(32-24) = 2^8 = 256 IP-uri

Ip-uri utilizabile:
256-2 = 254
```

De reținut că primul și ultimul IP din range sunt rezervate (pentru network address și broadcast address), deci numărul real de IP-uri utilizabile este numărul total minus 2.

Deci, IP-uri utilizabile sunt: **numarul total de ip-uri - 2**

STRUCTURA DIRECTORULUI RADACINA (/)

Director	Descriere
`/`	Directorul rădăcină, rădăcina întregului sistem de fișiere.
`/bin`	Contine fișiere binare (executabile) esențiale pentru funcționarea sistemului (comenzi de bază precum `ls`, `cp`, `mv`).
`/boot`	Conține fișiere necesare pentru încărcarea sistemului, inclusiv kernel-ul și fișierele de configurare pentru bootloader (de ex. `grub`).
`/dev`	Contine fișierele de dispozitive (dispozitive hardware și virtuale, cum ar fi hard disk-uri, terminale).
`/etc`	Conține fișierele de configurare ale sistemului și ale aplicațiilor (de ex. `passwd`, `hostname`, `network/interfaces`).
`/home`	Directorul utilizatorilor, fiecare utilizator are un subdirector în acest director (`/home/user`).
`/lib`	Contine biblioteci esențiale pentru programele din `/bin` și `/sbin`.
`/libexec`	Conține programe auxiliare (executabile) utilizate de alte aplicații sau servicii, dar nu sunt accesibile direct utilizatorului.
`/media`	Director pentru montarea dispozitivelor externe (CD-ROM, DVD, stick-uri USB etc.).
`/mnt`	Utilizat pentru montarea temporară a sistemelor de fișiere (de exemplu, un sistem de fișiere pe un server de rețea).
`/opt`	Contine aplicații sau pachete software suplimentare care nu sunt incluse în distribuția de bază.
`/proc`	Contine informații despre procesele și kernel-ul sistemului, accesibile prin fișiere virtuale.
`/root`	Directorul home pentru utilizatorul `root` (administratorul sistemului).
`/run`	Contine fișiere temporare de runtime care sunt utilizate de aplicații și servicii.
`/sbin`	Conține executabilele esențiale pentru administrarea sistemului (comenzi de administrare precum `fdisk`, `reboot`, `ifconfig`).
`/srv`	Conține fișiere pentru servicii care sunt oferite de sistem (de exemplu, fișierele pentru un server web sau FTP).
`/sys`	Contine informații despre sistemul de operare și kernel, fișierele sunt virtuale și permit interacțiunea cu kernel-ul (de ex. controlul dispozitivelor).
`/tmp`	Director pentru fișiere temporare care pot fi șterse la restartul sistemului.
`/usr`	Contine majoritatea programelor și aplicațiilor utilizatorilor, inclusiv binare, librării, documentație, și fișiere de configurare.

`/var`	Conține fișiere variabile, cum ar fi jurnale de sistem (log-uri), cozi de mesaje, baze de date sau fișiere temporare pentru aplicații.

Explicații suplimentare:

- **`/home/user`**: Fiecare utilizator are un subdirector în `/home`, unde își stochează fișierele personale și configurațiile.
- **`/dev`**: Acest director conține fișiere speciale pentru interacțiunea cu dispozitivele hardware, cum ar fi `/dev/sda` pentru hard disk sau `/dev/tty` pentru terminale.
- **`/tmp`**: Este un director temporar care este folosit de sistem și de aplicații pentru a stoca fișiere temporare. De obicei, fișierele din acest director sunt șterse la repornirea sistemului.
- **`/usr/local`**: De obicei, fișierele software instalate manual sunt plasate în acest director, în afacerea distribuită a sistemului de operare.

METODE DE REDIRECTARE

Sursă	Destinație	Exemplu comandă
intrare (stdin)	Fisier	`./program < fisier_intrare`
ieșire (stdout)	Fisier	`./program > fisier_iesire`
eroare (stderr)	Fisier	`./program 2> fisier_erori`
eroare (stderr) & ieșire (stdout)	Fisier	`./program 2>&1`

Redirectări folosind fișiere speciale

Comanda	Efect
`./program 2> /dev/null`	Mesajele de la ieșirea de eroare standard nu sunt afișate
`./program > /dev/null 2>&1`	Nici un mesaj nu este afișat
`> new_file`	Creează un fișier gol cu numele `new_file`
`cat /dev/null > new_file`	Creează un fișier cu același conținut cu `/dev/null`

CLASIFICAREA SISTEMELOR DE FISIERE

Tip	Exemplu	Descriere
sisteme de fișiere cu suport fizic	FAT32, NTFS, Ext4, APFS	Se regăsesc de obicei pe un mediu de stocare
sisteme de fișiere virtuale	procfs, devfs, SSHFS	Conțin fișiere/date generate de SO sau de o altă componentă software
sisteme de fișiere pentru rețea	NFS, SMB	Utilizate pentru accesul la fișiere aflate în rețea

MANAGEMENTUL UTILIZATORILOR

Informații despre utilizatori

Fișier	Rol	Informații
`/etc/passwd`	Informații utilizatori	Nume de utilizator, UID, director home, shell de login, GID
`/etc/shadow`	Parole utilizatori	Nume de utilizator, parolă criptată, informații expirare parolă
`/etc/group`	Informații grupuri	Nume grup, GID, utilizatori aferenți

Utilitare de investigare utilizatori

Utilitar	Rol	Fișiere investigate

`id`	Informații despre utilizator	`/etc/passwd` , `/etc/group`
`groups`	Grupurile utilizatorului curent	`/etc/group`
`users` , `w` , `who`	Utilizatorii autentificați în sistem acum	`/var/run/utmp`
`whoami`	Numele utilizatorului curent	N/A
`finger` , `pinky`	Informații complete despre un utilizator	`/etc/passwd` , `/etc/group`

Utilitare de gestionare a utilizatorilor

Operație	Utilitare	Fișiere modificate
Adăugare utilizator	`useradd`	`/etc/passwd` , `/etc/shadow` , `/etc/group`
Ștergere utilizator	`userdel`	`/etc/passwd` , `/etc/shadow` , `/etc/group`
Modificare utilizator	`usermod`	`/etc/passwd` , `/etc/shadow` , `/etc/group`
Adăugare grup	`groupadd`	`/etc/group`
Ștergere grup	`groupdel`	`/etc/group`
Modificare grup	`groupmod`	`/etc/group`
Modificare shell	`chsh`	`/etc/passwd`
Modificare informații utilizator	`chfn`	`/etc/passwd`
Schimbare parolă	`passwd`	`/etc/shadow`

RETELISTICA SI INTERNET

Tipuri de servicii de Internet

Tip de serviciu	Exemple de servicii
Acces la distanță	SSH (Secure Shell), Remote Desktop, VNC
Acces de informații	WWW (World Wide Web), Wikipedia, Google
Livrare de conținut (multimedia)	Netflix, YouTube, Amazon Prime, HBO GO, Spotify
Comunicare / messaging	E-mail, forumuri, instant messaging, Facebook Messenger, WhatsApp, Slack, Signal, Telegram, Microsoft Teams
Servicii online, intermediere tranzacții	Amazon, eBay, PayPal, Revolut, Glovo, Uber, AirBnb, Booking.com
Divertisment online	Gaming, Steam
Spațiu de stocare	Google Drive, Microsoft One Drive, Dropbox
Resurse de calcul, sisteme distribuite	Amazon EC2 / S3, Google Compute Engine, Microsoft Azure, Rackspace
Colaborare	GitHub, Microsoft SharePoint, Google Drive, Trello

Medii de transmisie

Mediu	Semnal	Viteză maximă	Avantaje	Dezavantaje
Cablu de cupru electric	Electric	10 Gbps	Cost redus, individual fiecăruia sistem (nepartajat)	Incomod
Aer electromagnetic	Electromagnetic	300 Mbps	Flexibilitate	Viteză redusă, partajat
Fibra optică (lumină)	Optic	100 Gbps	Viteză mare	Cost mare

Tabelul cu porturi implicite utilizate de diverse servicii de rețea

Nr.	Protocol	Port	Descriere
1	FTP	21/tcp	Transfer fișiere

2	SSH	22/tcp	SSH Remote Login
3	Telnet	23/tcp	Terminal de rețea
4	SMTP	25/tcp	Mail (Simple Mail Transfer Protocol)
5	Domain	53/tcp	Domain Name Server (DNS)
6	Domain	53/udp	Domain Name Server (DNS)
7	HTTP	80/tcp	WorldWideWeb (HTTP)
8	POP3	110/tcp	POP version 3
9	NTP	123/tcp	Network Time Protocol
10	NTP	123/udp	Network Time Protocol
11	IMAP2	143/tcp	Interim Mail Access Protocol 2 and 4
12	SNMP	161/tcp	Simple Network Management Protocol
13	SNMP	161/udp	Simple Network Management Protocol
14	BGP	179/tcp	Border Gateway Protocol
15	IRC	194/tcp	Internet Relay Chat
16	LDAP	389/tcp	Lightweight Directory Access Protocol
17	LDAP	389/udp	Lightweight Directory Access Protocol
18	HTTPS	443/tcp	HTTP protocol over TLS/SSL

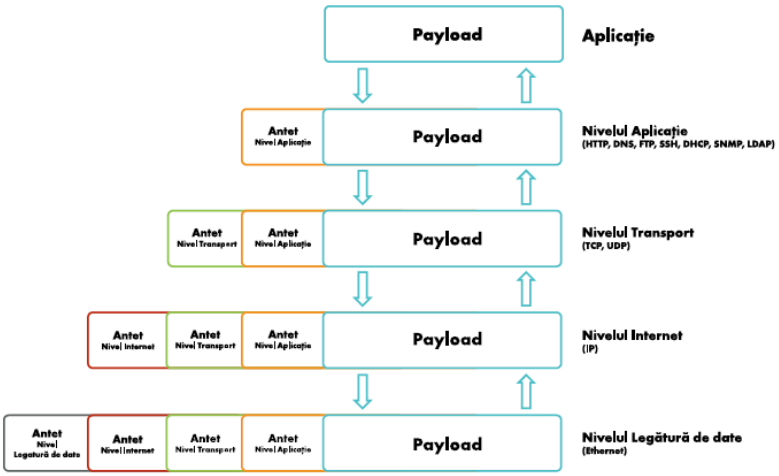
Tabel cu cele mai cunoscute abrevieri de la rețelistica

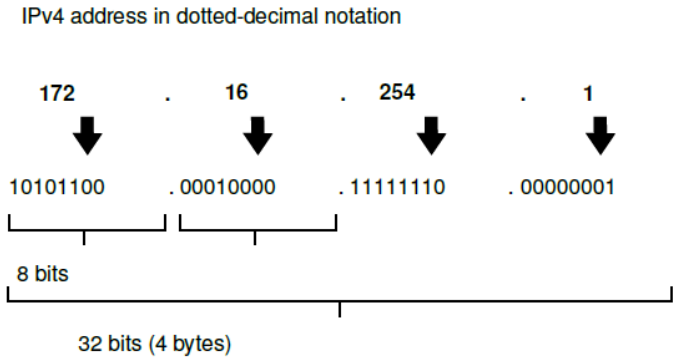
Abreviere	Descriere
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
FTP	File Transfer Protocol
SSH	Secure Shell
SMTP	Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
POP3	Post Office Protocol 3
VPN	Virtual Private Network
LAN	Local Area Network
WAN	Wide Area Network
NIC	Network Interface Card
URL	Uniform Resource Locator
IP	Internet Protocol

IPv4 OCUPA 32 BITI

IPv6 OCUPA 128 BITI

Stiva TCP/IP





Comandă	Descriere	Scop
`ip address show`	Afișează toate interfețele de rețea, adresele MAC/IP și masca de rețea.	Pentru a verifica configurația IP a interfețelor (ex: eth0, wlan0).
`ip route show`	Listează tabela de rutare, inclusiv adresa gateway-ului implicit.	Pentru a identifica ruta de acces la internet sau alte rețele.
`cat /etc/resolv.conf`	Afișează serverele DNS configurate.	Pentru a verifica ce servere DNS sunt folosite pentru rezolvarea numelor de domeniu.
`ping <adresă_IP>`	Testează conectivitatea cu o adresă IP sau domeniu.	Pentru a verifica dacă un host este accesibil (ex: `ping 8.8.8.8`).
`host <domeniu>`	Rezolvă un nume de domeniu în adresă IP folosind DNS.	Pentru a verifica dacă DNS-ul funcționează corect (ex: `host google.com`).
`traceroute <domeniu>`	Afișează traseul (rutele intermediare) până la o adresă IP sau domeniu.	Pentru a diagnostica unde apare o pierdere de conectivitate pe traseu.
`ethtool <interfață>`	Afișează starea fizică a unei interfețe de rețea (ex: viteza, duplex).	Pentru a verifica dacă cablul este conectat sau dacă interfața este activă.

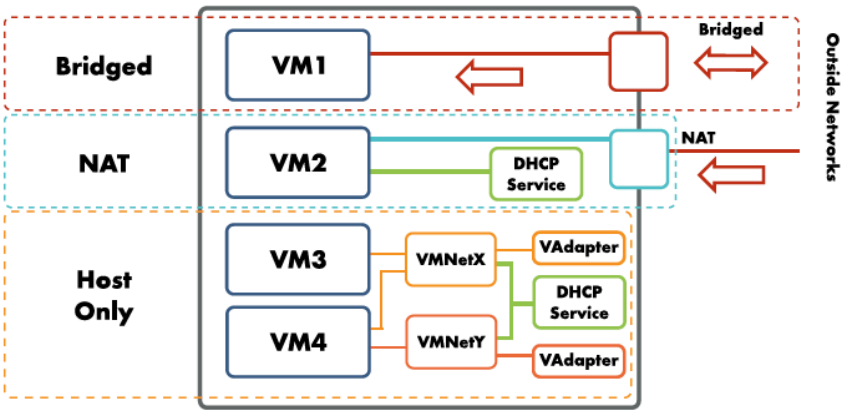
MASINI VIRTUALE

Tehnologii ce implementează mecanismul de container în sistemele Linux

Tehnologie	Descriere
LXC	Oferă posibilitatea rulării unor servicii într-un mediu izolat de sistemul de bază.
OpenVZ	Similar cu LXC, dar nu este prezent în mod implicit în nucleul Linux.
Docker	Permite rularea unei singure aplicații într-un container.

Crearea mașinilor virtuale

Parametru	Descriere
Numele mașinii virtuale	Se specifică numele mașinii virtuale.
Numărul de procesoare (nuclee/core-uri)	Se specifică numărul de nuclee sau core-uri ale procesorului virtualizat.
Cantitatea de memorie disponibilă	Se specifică cantitatea de memorie RAM care va fi alocată mașinii virtuale.
Mărimea discului	Se specifică dimensiunea discului virtualizat.
CD-ROM și imagine .iso asociată	Se specifică dacă va exista un CD-ROM virtual și ce imagine .iso va fi asociată acestuia.
Tipul de rețea	Se specifică tipul de rețea pentru mașina virtuală (detaliat în secțiunea 14.2.1).



ARHIVARE

Prin procesul de arhivare, mai multe fișiere și directoare sunt strânse la un loc într-un fișier unic, realizându-se de obicei și reducerea dimensiunii (compresie). În general, noțiunea de arhivare se referă doar la lipirea datelor într-un singur loc, în vreme ce compresia se referă la codificarea datelor pentru a reduce dimensiunea fișierului rezultat.

tar

Parametru	Descriere
`c` (create)	Pentru a crea arhiva
`x` (extract)	Dezarhivează
`t` (list)	Listează conținut
`v` (verbose)	Arată ce se întâmplă
`C` (output directory)	Schimbă directorul înainte de extragere
`f nume_arhiva.tar` (file)	Numele arhivei

> **Observație!** Parametrul `f` din `nume_arhiva.tar` este unic – astfel, când îl folosiți pentru a indica un fișier, acesta trebuie să fie ultimul din lista de parametri.

compresie

Cele mai utilizate utilitare de compresie în Linux sunt:

- `gzip` – oferă viteză mare de compresie, dar cu un raport de compresie mai mic (fișierele rezultate sunt mai mari).
- `bzip2` – are viteză mai mică, dar obține un raport de compresie mai bun (fișiere mai mici).

Comenzii `tar` i se pot transmite parametrii:

- `z` pentru compresie cu **gzip**
- `j` pentru compresie cu **bzip2**

Utilizare:

- Crearea unei arhive .tar (fără compresie)

  `tar -cvf nume_arhiva.tar /calea/catre/fisiere_sau_directoare/`

- Listarea conținutului arhivei

  `tar -tvf nume_arhiva.tar`

- Compresia arhivei cu gzip (.tar.gz)

  `tar -czvf nume_arhiva.tar.gz /calea/catre/fisiere/`

- Compresia cu bzip2 (.tar.bz2)

  `tar -cjvf nume_arhiva.tar.bz2 /calea/catre/fisiere/`

- Dezarhivarea arhivei in directorul curent

  `tar -xvf nume_arhiva.tar`

- Dezarhivarea arhivei intr-un alt director

  `tar -xvf nume_arhiva.tar -C /calea/catre/director_destinatie/`

> **Pentru fisere cu compresie mai adaugam parametrul in functie de compresie. (`z` sau `j`), ne putem folosii de extensia arhivei pentru a vedea tipul de compresie sau folosim utilitarul `file`**

ECHIVALENTE COMENZI LINUX SI WINDOWS

Descriere	Comandă Linux	Comandă Windows
Afișează informații despre comandă	`comanda --help`	`comanda /?`
Schimbă directorul curent	`cd`	`cd`
Afișează directorul curent	`pwd`	`chdir`
Șterge ecranul consolei	`clear`	`cls`
Copiază un fișier	`cp`	`copy`
Șterge un fișier	`rm`	`del`
Afișează conținutul directorului curent	`ls`	`dir`
Editează un fișier text	`vim`	`edit`
Închide shell-ul curent	`exit`	`exit`
Compară două fișiere și afișează diferențele	`diff`	`fc`
Caută fișiere	`find`	`find`
Formatează un disc	`mkfs` / `mke2fs`	`format`
Creează un director nou	`mkdir`	`mkdir`
Mută un fișier	`mv`	`move`
Redenumește un fișier	`mv`	`ren`
Afișează ora sistemului	`date`	`time`

dpkg

Comandă	Descriere	Exemplu
`dpkg -i`	Instalează un pachet `.deb`	`sudo dpkg -i pachet.deb`
`dpkg -r`	Dezinstalează un pachet (păstrează fișierele de configurare)	`sudo dpkg -r nume_pachet`
`dpkg -P`	Dezinstalează complet un pachet (inclusiv fișierele de configurare)	`sudo dpkg -P nume_pachet`
`dpkg -l`	Listează toate pachetele instalate	`dpkg -l`
`dpkg -L`	Afișează fișierele instalate de un pachet	`dpkg -L nume_pachet`
`dpkg -s`	Afișează starea unui pachet (dacă este instalat, versiune etc.)	`dpkg -s nume_pachet`
`dpkg -S`	Caută care pachet deține un anumit fișier	`dpkg -S /usr/bin/nume_comanda`
`dpkg --configure`	Reconfigurează un pachet instalat	`sudo dpkg --configure -a`
`dpkg-reconfigure`	Reconfigurează interactiv un pachet	`sudo dpkg-reconfigure nume_pachet`

Listează pachetele care conțin un anumit cuvânt: `dpkg -l | grep "cuvant_cheie"`

snap

Comanda	Descriere	Exemplu
`snap find <cuvânt_cheie>`	Caută pachete	`snap find svg`
`sudo snap install <pachet>`	Instalează pachet	`sudo snap install inkscape`
`snap info <pachet>`	Arată informații pachet	`snap info inkscape`
`sudo snap remove <pachet>`	Dezinstalează pachet	`sudo snap remove inkscape`
`sudo snap refresh <pachet>`	Actualizează un pachet	`sudo snap refresh inkscape`
`sudo snap refresh`	Actualizează toate pachetele	`sudo snap refresh`
`snap download <pachet>`	**Descarcă pachet (fără instalare)**	`snap download inkscape`
`snap list`	Listează pachetele instalate	`snap list`
`snap list | grep <cuvânt_cheie>`	Caută pachet instalat	`snap list | grep inkscape`

Gestionarea pachetelor

Acțiune	APT/dpkg (Debian/Ubuntu)	DNF/RPM (RHEL/Fedora)	Pacman (Arch Linux)	PKG (FreeBSD)	Brew (macOS/Linux)	Choco (Windows)
Caută pachete	`apt search <pachet>`	`dnf search <pachet>`	`pacman -Ss <pachet>`	`pkg search <pachet>`	`brew search <pachet>`	`choco search <pachet>`
Instalează pachet	`sudo apt install <pachet>`	`sudo dnf install <pachet>`	`sudo pacman -S <pachet>`	`sudo pkg install <pachet>`	`brew install <pachet>`	`choco install <pachet>`
Șterge pachet	`sudo apt remove <pachet>`	`sudo dnf remove <pachet>`	`sudo pacman -R <pachet>`	`sudo pkg delete <pachet>`	`brew uninstall <pachet>`	`choco uninstall <pachet>`
Actualizează lista pachetelor	`sudo apt update`	`sudo dnf check-update`	`sudo pacman -Sy`	`sudo pkg update`	`brew update`	`choco upgrade all --noop`
Actualizează toate pachetele	`sudo apt upgrade`	`sudo dnf upgrade`	`sudo pacman -Syu`	`sudo pkg upgrade`	`brew upgrade`	`choco upgrade all`
Descarcă pachet (fără instalare)	`apt download <pachet>`	`dnf download <pachet>`	`pacman -Sw <pachet>`	`pkg fetch <pachet>`	`brew fetch <pachet>`	`choco download <pachet>`
Listează pachete instalate	`dpkg -l` / `apt list --installed`	`rpm -qa` / `dnf list installed`	`pacman -Q`	`pkg info`	`brew list`	`choco list --local-only`
Caută pachet instalat	`dpkg -l *<nume>*`	`rpm -qa *<nume>*`	`pacman -Qs <nume>`	`pkg info \*<nume>\*`	`brew list \*<nume>\*`	`choco list <nume>`

Care pachet deține un fișier?	`dpkg -S /cale/fișier`	`rpm -qf /cale/fișier`	`pacman -Qo /cale/fișier`	`pkg which /cale/fișier`	`brew which <fișier>`	**nu există**
Arată fișierele unui pachet	`dpkg -L <pachet>`	`rpm -ql <pachet>`	`pacman -Ql <pachet>`	`pkg info -l <pachet>`	**nu există**	`choco list --files <pachet>`
Arată informații despre pachet	`apt show <pachet>`	`dnf info <pachet>`	`pacman -Si <pachet>`	`pkg info <pachet>`	`brew info <pachet>`	`choco info <pachet>`
Repară dependințe	`sudo apt --fix-broken install`	`sudo dnf autoremove`	`sudo pacman -D --asdeps`	`sudo pkg audit -F`	`brew doctor`	`choco upgrade all`

Un **proces** este un program în execuție care utilizează resursele sistemului de calcul pentru a efectua una sau mai multe acțiuni.

Altfel spus:

- Un **program** este o entitate **statică** – un fișier executabil stocat pe disc.
- Un **proces** este o entitate **dinamică** – o instanță a unui program care rulează în memorie și consumă resurse (CPU, memorie, etc.).

Procesului i se asociază un **context de execuție**, care include starea curentă, registrele CPU, memoria alocată și alte resurse necesare funcționării.

Atributele unui proces

Exemplu (output-ul comenzii `ps -ef`)

```
UID   PID  PPID  C STIME TTY      TIME       CMD
root    1     0  0 Oct02 ?    00:00:05   /sbin/init splash
```

Atribut	Rol	Valoarea Actuală	Momentul Atribuirii	Modificabil
PID	Identificator unic al procesului	`1` (init)	Pornire	Nu
PPID	Identificatorul procesului părinte	`0` (kernel)	Pornire	Da
Program Executabil	Imaginea procesului (cod + date)	`/sbin/init splash`	Pornire	Nu
UID/GID	Permisiuni (utilizator/grup care deține procesul)	`root` (UID 0)	Pornire	Nu
Prioritate (nice)	Importanța în alocarea resurselor CPU	`0` (valoare implicită)	Pornire	Da
Terminal (TTY)	Interfața de comunicare cu utilizatorul	`?` (fără terminal)	Pornire	Da
Fișiere Deschise	Resursele I/O folosite de proces	-	Rulare	Da
Stare	Starea curentă (ex: R-Running, S-Sleeping)	`S` (Sleeping)	Rulare	Da
Timp CPU (TIME)	Timpul total de utilizare a CPU	`00:00:05`	Rulare	Da
Memorie Consumată	Cantitatea de memorie RAM alocată	-	Rulare	Da

Spațiu de Adrese	Harta memoriei virtuale a procesului	-	Pornire	Da

> **Valoarea actuala a fost preluata din exemplu**

Sistemul de operare oferă utilitare și comenzi pentru a afișa procese, pentru a urmări atributele și resursele lor. Utilitarele din această categorie sunt de două tipuri:

- cele care monitorizeaza procesele sistemului: `top`, `htop`, `iotop`, `sysstat`.
- cele care afișează un snapshot al momentului (procese active în acest moment și atributele lor): `ps`, `pgrep`, `pidof`, `pstree`, `pmap`

ps

Parametru	Descriere	Exemplu de utilizare	Exemplu de output
`-e`	Afișează **toate** procesele din sistem	`ps -e`	`PID TTY TIME CMD1 ? 00:00:01 systemd`
`-f`	Format extins (afișează UID, PID, PPID, etc.)	`ps -f`	`UID PID PPID C STIME TTY TIME CMD`
`-u`	Afișează procesele unui anumit utilizator	`ps -u root`	Procesele ruleate de `root`
`-p`	Afișează un proces specific după PID	`ps -p 1234`	Detalii despre procesul cu PID `1234`
`-C`	Afișează procesele după numele comenzii	`ps -C firefox`	Toate procesele `firefox`
`--forest`	Arată ierarhia proceselor (părinți-copii)	`ps -ef --forest`	`systemd─┬─firefox───5`
`-o`	Format personalizat (câmpuri specifice)	`ps -eo pid,ppid,cmd`	`PID PPID CMD1 0 /sbin/init`
`--sort`	Sortează procesele după un criteriu (ex: `-%mem`, `-pcpu`)	`ps -eo pid,%mem --sort=-%mem`	Procesele sortate după utilizarea memoriei
`-L`	Afișează firele de execuție (threads)	`ps -L -p 1234`	Firele asociate procesului `1234`
`-H`	Afișează procesele în format ierarhic (similar cu `pstree`)	`ps -eH`	Ierarhia proceselor
`-aux`	Combinație: `-a` (toate procesele), `-u` (detalii utilizator), `-x` (fără TTY)	`ps -aux`	Listă completă cu detalii
`-T`	Afișează procesele asociate cu terminalul curent	`ps -T`	Procesele din sesiunea curentă
`-c`	Afișează numele comenzii scurte (fără cale completă)	`ps -c firefox`	`firefox` (în loc de `/usr/bin/firefox`)
`--no-headers`	Exclude antetul din output	`ps -e --no-headers`	Listă brută de PID și comenzi
`-M`	Afișează informații despre namespace-uri	`ps -M -p 1234`	Detalii namespace pentru PID `1234`

nice

Parametru	Descriere	Exemplu de Utilizare	Explicatie Exemplu
`nice -n <valoare> <comanda>`	Lansează o comandă cu o prioritate specifică (niceness). Valoarea implicită este `10`.	`nice -n 5 tar -czf backup.tar.gz /home`	Rulează `tar` cu niceness `5` (prioritate mai mare decât implicit).
`nice --adjustment= <valoare> <comanda>`	Echivalent cu `-n`. Valoarea poate fi negativă (nevoie de `sudo`).	`sudo nice --adjustment=-5 apt upgrade`	Rulează `apt upgrade` cu prioritate crescută (`-5`). Necesită `sudo`.
`nice -<valoare> <comanda>`	Scurtătură pentru valori negative (ex: `-5` = `--adjustment=-5`).	`sudo nice -5 dd if=/dev/zero of=/dev/null`	Rulează `dd` cu prioritate crescută (`-5`).
`nice --help`	Afișează ajutorul pentru comandă.	`nice --help`	Arată opțiunile disponibile și sintaxa.
`nice --version`	Afișează versiunea comenzii.	`nice --version`	Ex: `nice (GNU coreutils) 8.32`

pstree

Parametru	Descriere	Exemplu	Output Exemplu
`pstree`	Afișează toate procesele în format arbore	`pstree`	`systemd─┬─NetworkManager─┬─dhclient`
`pstree -p`	Afișează PID-uri pentru fiecare proces	`pstree -p`	`systemd(1)─┬─firefox(1234)`
`pstree -u`	Afișează utilizatorul care deține fiecare proces	`pstree -u`	`systemd(root)─┬─bash(user)`
`pstree -H <PID>`	Highlightează un proces specific în arbore	`pstree -H 1234`	Afișează arborele cu PID `1234` evidențiat
`pstree -T`	Afișează doar procesele fără terminal (detached)	`pstree -T`	`systemd───sshd───sshd`
`pstree -s <PID>`	Afișează lanțul de părinți ai unui proces	`pstree -s 5678`	`systemd───nginx───worker(5678)`
`pstree -a`	Afișează argumentele comenzilor	`pstree -a`	`firefox,--profile,/tmp/profile`

pgrep

Parametru	Descriere	Exemplu	Output Exemplu
`pgrep <nume>`	Caută PID-urile proceselor după nume	`pgrep firefox`	`1234` (PID-ul procesului Firefox)
`pgrep -u <user>`	Caută procesele unui utilizator specific	`pgrep -u root`	`1 2 10` (PID-uri ale proceselor `root`)
`pgrep -l`	Afișează și numele proceselor	`pgrep -l sshd`	`567 sshd`
`pgrep -f <pattern>`	Caută procese după șablonul comenzii complete	`pgrep -f "nginx -g"`	`890` (PID pentru `nginx -g daemon;`)
`pgrep -x <nume>`	Caută procese cu nume exact	`pgrep -x bash`	`4567` (doar procese numite `bash`)
`pgrep -n`	Afișează cel mai recent proces găsit	`pgrep -n firefox`	`1234` (cel mai nou PID Firefox)
`pgrep -o`	Afișează cel mai vechi proces găsit	`pgrep -o sshd`	`567` (cel mai vechi PID `sshd`)

su / sudo

Parametru/Atribut	`su` (Substitute User)	`sudo` (Super User DO)
Scop	Schimbă utilizatorul curent (implicit la `root`)	Execută o singură comandă cu drepturi de `root` sau alt utilizator

Utilizare implicită	`su` → cere parola `root`
	`su -`
	→ login complet cu mediu `root`
		`sudo <comandă>` → cere parola utilizatorului curent

Schimbare mediu de lucru	`su` → păstrează mediul curent `su -` → încarcă mediul țintă (ex: `$PATH` al `root`)	Păstrează mediul curent (dacă nu se folosește `-i` pentru login interactiv)
Cerințe de configurare	Necesită parola `root`	Utilizatorul trebuie adăugat în `/etc/sudoers`
Siguranță	Riscul de a lăsa sesiune `root` deschisă	Execută doar comanda specificată, apoi revine la drepturi normale
Jurnalizare	Nu înregistrează comenzi în mod implicit	Loghează comenzi în `/var/log/auth.log`

Exemple	`su -` → login ca `root`
	`su - utilizator`
	→ schimbă utilizator
		`sudo apt update`
		`sudo -u postgres psql`
		→ rulează ca `postgres`

Parametri utili	`-` sau `-l` → login interactiv `-c <comandă>` → rulează o comandă	`-i` → login interactiv ca `root -u <user>` → rulează ca alt utilizator
Ieșire rapidă	`exit` sau `Ctrl+D`	N/A (se revine automat la utilizatorul curent)

Expresii regulate / REGEX

Construcție	Rol / Efect	Exemplu	Explicație

`.`	Potrivește **orice caracter** (cu excepția newline).	`a.c`	Potrivește "abc", "a1c", "a-c" (orice caracter între `a` și `c`).
`a?`	Caracterul `a` apare **o dată sau deloc**.	`ba?b`	Potrivește "bb" sau "bab".
`a*`	Caracterul `a` apare **de zero sau de mai multe ori**.	`ba*b`	Potrivește "bb", "bab", "baaab", etc.
`a+`	Caracterul `a` apare **cel puțin o dată**.	`ba+b`	Potrivește "bab", "baaab", dar nu "bb".
`^`	Potrivește **începutul de linie**.	`^Hello`	Potrivește "Hello" doar dacă apare la începutul liniei.
`$`	Potrivește **sfârșitul de linie**.	`world$`	Potrivește "world" doar dacă apare la sfârșitul liniei.
`[...]`	Potrivește **orice caracter din setul specificat**.	`[aeiou]`	Potrivește orice vocală (a, e, i, o, u).
`\`	**Escaping** – folosit pentru a trata metacaracterele ca literali.	`\.`	Potrivește caracterul literal `.` (nu orice caracter).
`[A-Z][a-z]+`	Potrivește un **nume propriu** (literă mare urmată de litere mici).	`John`, `Alice`	Potrivește cuvinte care încep cu majusculă și continuă cu minuscule.
`[A-Z][a-z]+ [A-Z][a-z]+`	Potrivește **prenume și nume** (două cuvinte cu majusculă).	`John Doe`, `Mary Smith`	Potrivește perechi de cuvinte cu formă de nume propriu.
`[a-zA-Z_][a-zA-Z_0-9]*`	Potrivește un **nume de variabilă** (începe cu literă sau `_`, urmat de litere, cifre sau `_`).	`var1`, `_temp`	Valabil pentru identificatori în majoritatea limbajelor de programare.
`[0-9]{10}`	Potrivește un **număr de telefon** (10 cifre consecutive).	`0722123456`	Potrivește exact 10 cifre.
`^#include +<[^>]+>$`	Potrivește o **directivă** `#include` **în C** (cu `<...>`).	`#include <stdio.h>`	Potrivește linii care încep cu `#include`, urmate de un fișier între `< >`.

- `[abc]` – Potrivește `a`, `b` **sau** `c`.
- `[^abc]` – Potrivește **orice caracter care NU este** `a`, `b` **sau** `c`.

ex: `[^>]` – Potrivește **orice caracter care NU este** `>`.

Prelucare

Comandă	Rol / Efect	Tip de Acțiune	Exemplu de Utilizare
`grep`	Extrage liniile care se potrivesc cu o expresie regulată.	Extragere	`grep "error" log.txt`
`cut`	Extrage anumite coloane (câmpuri) dintr-un fișier.	Selecție	`cut -d',' -f1,3 data.csv`
`nl`	Afișează numărul liniei pentru fiecare linie.	Reformatare	`nl document.txt`
`wc`	Afișează numărul de linii, cuvinte sau caractere.	Sumarizare	`wc -l file.txt`
`sort`	Sortează liniile unui fișier.	Ordonare	`sort names.txt`
`uniq`	Elimină liniile duplicate (necesită intrare sortată).	Extragere, Reformatare	`sort file.txt | uniq`
`fmt`	Reformatează textul pentru o lățime specificată.	Reformatare	`fmt -w 50 long_text.txt`
`paste`	Alătură conținutul mai multor fișiere pe coloane.	Reformatare	`paste file1.txt file2.txt`
`join`	Combină fișiere pe baza unor câmpuri comune.	Reformatare	`join file1.txt file2.txt`
`rev`	Inversează caracterele fiecărei linii.	Reformatare	`rev text.txt`
`tail`	Afișează doar liniile finale ale unui fișier.	Selecție	`tail -n 10 log.txt`

`head`	Afișează doar liniile inițiale ale unui fișier.	Selecție	`head -n 5 data.txt`
`tac`	Inversează ordinea liniilor unui fișier.	Reformatare	`tac file.txt`
`tr`	Traduce sau șterge caractere.	Înlocuire	`tr 'a-z' 'A-Z' < file.txt`
`tee`	Clonează conținutul la ieșirea standard și într-un fișier.	Duplicare	`ls | tee directories.txt`
`sed`	Efectuează prelucrări complexe pe text (înlocuiri, ștergeri, etc.).	Toate tipurile	`sed 's/old/new/g' file.txt`
`awk`	Limbaj de programare pentru prelucrarea textului pe câmpuri.	Toate tipurile	`awk '{print $1}' data.txt`

Tip RAID	Descriere	Număr minim discuri	Capacitate utilă	Toleranță la defecte	Av
`RAID 0`	Datele sunt distribuite (striping) pe toate discurile fără replicare.	2	Suma capacităților tuturor discurilor.	**Nicio toleranță** – pierderea unui disc = pierderea tuturor datelor.	Pe ric (ci pa
`RAID 1`	Datele sunt replicate (mirroring) pe discuri pereche.	2	Capacitatea celui mai mic disc (din cauza replicării).	Poate supraviețui defectării **unui disc**.	Si ric da
`RAID 5`	Datele și sumele de control (paritate) sunt distribuite pe toate discurile.	3	(N−1)×capacitate discului(*N*−1)×capacitate discului (unde N*N* = număr discuri).	Poate supraviețui defectării **unui singur disc**.	Bu înt ca pe sig
`RAID 10`	Combină RAID 0 (striping) și RAID 1 (mirroring).	4	N2×capacitate discului2*N*×capacitate discului.	Poate supraviețui defectării **mai multor discuri** (dacă nu fac parte din aceeași pereche mirrored).	Pe sig ex

BASH

File & Directory Tests

Test	Description	Example
`-e`	File exists (any type)	`[ -e "file.txt" ]`
`-f`	Regular file (not dir/symlink)	`[ -f "file.txt" ]`
`-d`	Directory exists	`[ -d "/path" ]`
`-L` / `-h`	Symbolic link	`[ -L "symlink" ]`
`-r`	File is readable	`[ -r "file.txt" ]`
`-w`	File is writable	`[ -w "file.txt" ]`
`-x`	File is executable	`[ -x "script.sh" ]`
`-s`	File size > 0	`[ -s "file.txt" ]`
`-S`	File is a socket	`[ -S "/tmp/sock" ]`
`-p`	Named pipe (FIFO)	`[ -p "/tmp/pipe" ]`

`-c`	Character special file	`[ -c "/dev/tty" ]`
`-b`	Block special file	`[ -b "/dev/sda" ]`
`-g`	Setgid bit set	`[ -g "file" ]`
`-u`	Setuid bit set	`[ -u "file" ]`
`-k`	Sticky bit set	`[ -k "/tmp" ]`
`-N`	Modified since last read	`[ -N "file.txt" ]`
`-O`	You are the owner	`[ -O "file.txt" ]`
`-G`	Your group owns it	`[ -G "file.txt" ]`
`file1 -nt file2`	`file1` is newer than `file2`	`[ "a.txt" -nt "b.txt" ]`
`file1 -ot file2`	`file1` is older than `file2`	`[ "a.txt" -ot "b.txt" ]`
`file1 -ef file2`	Same inode (hard links)	`[ "a.txt" -ef "b.txt" ]`

String Tests

Test	Description	Example
`-z`	String is empty	`[ -z "$var" ]`
`-n`	String is not empty	`[ -n "$var" ]`
`=`	Strings are equal	`[ "$a" = "$b" ]`
`!=`	Strings are not equal	`[ "$a" != "$b" ]`
`<`	Lexicographically before	`[[ "a" < "b" ]]`
`>`	Lexicographically after	`[[ "b" > "a" ]]`
`=~`	Regex match (only in `[[ ]]`)	`[[ "$str" =~ ^[0-9]+$ ]]`

Numeric Tests

Test	Description	Example
`-eq`	Equal	`[ "$a" -eq 10 ]`
`-ne`	Not equal	`[ "$a" -ne 10 ]`
`-lt`	Less than	`[ "$a" -lt 10 ]`
`-le`	Less than or equal	`[ "$a" -le 10 ]`
`-gt`	Greater than	`[ "$a" -gt 10 ]`
`-ge`	Greater than or equal	`[ "$a" -ge 10 ]`

Logical Operators

Operator	Description	Example
`!`	NOT (negation)	`[ ! -f "file.txt" ]`
`-a`	AND (deprecated, use `&&`)	`[ "$a" -eq 1 -a "$b" -eq 2 ]`
`-o`	OR (deprecated, use `||`)	`[ "$a" -eq 1 -o "$b" -eq 2 ]`
`&&`	Logical AND (outside `[ ]`)	`[ "$a" -eq 1 ] && [ "$b" -eq 2 ]`
`||`	Logical OR (outside `[ ]`)	`[ "$a" -eq 1 ] \|\| [ "$b" -eq 2 ]`

Advanced Conditions (`[[ ]]` vs `[ ]`)

Feature	`[ ]` (POSIX)	`[[ ]]` (Bash)
Regex (`=~`)	❌ No	✅ Yes
Word splitting	❌ Yes (quotes needed)	✅ No (safer)

Logical operators (`&&` , `||`)	❌ No (use `-a` , `-o`)	✅ Yes
Pattern matching (`==` , `!=`)	❌ No	✅ Yes (`[[ "file" == *.txt ]]`)
No filename expansion	❌ No	✅ Yes (`[[ "$var" == "*" ]]` treats `*` as literal)

Special Variables

Variable	Description
`$0`	Script name
`$1` , `$2` , ...	Positional arguments
`$#`	Number of arguments
`$@`	All arguments as separate strings
`$*`	All arguments as a single string
`$?`	Exit status of last command
`$$`	Current process ID (PID)
`$!`	PID of last background job

Redirections & Pipes

`cmd > file`	Redirect **stdout** to `file` (overwrite)	`ls > files.txt`
`cmd >> file`	Append **stdout** to `file`	`echo "new" >> log.txt`
`cmd 2> file`	Redirect **stderr** to `file`	`grep "error" 2> errors.log`
`cmd &> file`	Redirect **stdout + stderr** to `file`	`command &> output.log`
`cmd > file 2>&1`	Same as `&>` (older syntax)	`cmd > log.txt 2>&1`
`cmd < file`	Read **stdin** from `file`	`grep "text" < input.txt`
`cmd <<EOF`	**Here-document** (multiline input)	See below
`cmd <<< "string"`	**Here-string** (input from string)	`grep "foo" <<< "foo bar"`

Standard File Descriptors (FD)

FD	Name	Default Target	Description
`0`	**stdin**	Keyboard	Standard input (read data)
`1`	**stdout**	Terminal	Standard output (print data)
`2`	**stderr**	Terminal	Error messages

Advanced Redirections

Syntax	Description	Example
`cmd > /dev/null`	Discard **stdout**	`cmd > /dev/null`
`cmd 2> /dev/null`	Discard **stderr**	`cmd 2> /dev/null`
`cmd >&2`	Redirect **stdout → stderr**	`echo "Error!" >&2`
`cmd 2>&1`	Redirect **stderr → stdout**	`cmd 2>&1 | grep "error"`
`cmd1 | cmd2`	Pipe **stdout of** `cmd1` **→ stdin of** `cmd2`	`ls | grep ".txt"`
`cmd | tee file`	Print **and** save to `file`	`ls | tee files.txt`
`cmd |& tee file`	Pipe **stdout + stderr** to `tee`	`cmd |& tee log.txt`
`exec 3> file`	Open `file` as **FD 3** for writing	`exec 3> log.txt; echo "Hi" >&3`
`cmd1 <(cmd2)`	**Process substitution** (input as file)	`diff <(ls dir1) <(ls dir2)`

File Operations

```
if [[ -f "file.txt" ]]; then
    echo "File exists."
fi

mkdir -p /path/to/dir  # Creates dir if it doesn't exist
touch file.txt         # Creates empty file
rm -rf dir/            # Force-deletes directory
```

String Manipulation

```
str="Hello World"
echo ${#str}          # Length: 11
echo ${str:6:5}       # Substring: "World"
echo ${str//World/Bash} # Replace: "Hello Bash"
```

Loops

```
for i in {1..5}; do
    echo "$i"
done

while read line; do
    echo "$line"
done < file.txt
```

Functions

```
greet() {
    echo "Hello, $1!"
}
greet "Alice"  # Output: "Hello, Alice!"
```

MAKEFILE SI G++

1. Structura de bază a unui Makefile

```
target: dependencies
    <tab>command
```

- `target`: fișierul ce va fi generat (ex: executabilul).
- `dependencies`: fișierele de care targetul depinde.
- `command`: ce se execută pentru a construi targetul (prefixat cu TAB).

---

2. Compilare simplă

```
main: main.cpp
    g++ main.cpp -o main
```

---

3. Variabile uzuale

```
CXX = g++
CXXFLAGS = -Wall -Wextra -std=c++17 -O2
SRC = main.cpp utils.cpp
OBJ = $(SRC:.cpp=.o)
TARGET = program
```

4. Makefile modular

```
all: $(TARGET)

$(TARGET): $(OBJ)
    $(CXX) $(OBJ) -o $(TARGET)

%.o: %.cpp
    $(CXX) $(CXXFLAGS) -c $< -o $@

clean:
    rm -f $(OBJ) $(TARGET)

run: $(TARGET)
    ./$(TARGET)

.PHONY: all clean run
```

5. Variabile implicite în Make

- `$@` – targetul curent (ex: `program`)
- `$<` – prima dependență (ex: `main.cpp`)
- `$^` – toate dependențele (ex: `main.o utils.o`)

6. Funcții utile în Makefile

```
$(wildcard src/*.cpp)        # toate fișierele .cpp din src/
$(patsubst %.cpp, %.o, ...)   # transformă fișierele .cpp în fișiere .o
```

7. Adăugare directoare suplimentare

```
CXXFLAGS = -Iinclude/
LDFLAGS = -lpthread
```

Exemplu complet: Makefile + Structură + Cod

Structură directoare:

```
project/
├── Makefile
```

```
├── src/
│   ├── main.cpp
│   ├── utils.cpp
├── include/
│   ├── utils.hpp
```

Cod: `src/main.cpp`

```
#include <iostream>
#include "utils.hpp"

int main() {
    std::cout << "Result: " << add(3, 4) << std::endl;
    return 0;
}
```

Cod: `src/utils.cpp`

```
#include "utils.hpp"

int add(int a, int b) {
    return a + b;
}
```

Cod: `include/utils.hpp`

```
#ifndef UTILS_HPP
#define UTILS_HPP

int add(int a, int b);

#endif
```

Makefile (în directorul `project/`)

```
CXX = g++
CXXFLAGS = -Wall -Wextra -std=c++17 -O2 -Iinclude

SRC = src/main.cpp src/utils.cpp
OBJ = $(SRC:.cpp=.o)
TARGET = program

all: $(TARGET)

$(TARGET): $(OBJ)
    $(CXX) $(OBJ) -o $(TARGET)

src/%.o: src/%.cpp
    $(CXX) $(CXXFLAGS) -c $< -o $@

clean:
    rm -f $(OBJ) $(TARGET)
```

```
run: $(TARGET)
    ./$(TARGET)

.PHONY: all clean run
```

Exemplu 2

```
aCXX = g++
CXXFLAGS = -Wall -Wextra -std=c++17 -O2

TARGET = main
SRC = main.cpp
OBJ = $(SRC:.cpp=.o)

all: $(TARGET)

$(TARGET): $(OBJ)
    $(CXX) $(OBJ) -o $(TARGET)

%.o: %.cpp
    $(CXX) $(CXXFLAGS) -c $< -o $@

clean:
    rm -f $(OBJ) $(TARGET)

run: $(TARGET)
    ./$(TARGET)

.PHONY: all clean run
```

SCRIPTURI BASH

1. Structura de bază

```
#!/bin/bash
# Comentariu

echo "Hello, world!"
```

---

2. Variabile

```
NAME="Alice"
echo "Hello $NAME"
```

Tipuri:

- Locale: `VAR=value`

- De mediu: `export VAR=value`
- Read-only: `readonly VAR=value`

---

3. Citire input

```
read -p "Enter name: " NAME
read -s -p "Enter password: " PASS  # input ascuns
```

---

4. Argumente din linia de comandă

```
echo "$0"  # numele scriptului
echo "$1"  # primul argument
echo "$@"  # toate argumentele
```

---

5. Controlul erorilor

```
set -e      # oprește la prima eroare
set -u      # eroare dacă se folosește o variabilă nedefinită
set -o pipefail  # eșuează dacă oricare comandă din pipeline eșuează
```

---

6. Condții (`if` , `case`)

```
if [ "$NAME" = "Alice" ]; then
    echo "Hi Alice"
fi

case "$1" in
  start) echo "Starting";;
  stop) echo "Stopping";;
  *) echo "Unknown";;
esac
```

---

7. Buclă (`for` , `while` , `until`)

```
for file in *.txt; do
    echo "$file"
done

while read line; do
    echo "$line"
done < input.txt
```

---

8. Funcții

```
greet() {
    echo "Hello $1"
```

```
}

greet "Alice"
```

9. Trap semnale

```
trap 'echo "Caught SIGINT!"' SIGINT
trap cleanup EXIT

cleanup() {
   echo "Cleaning up before exit"
}
```

10. Execuție paralelă

```
long_task &
wait  # așteaptă toate joburile în fundal
```

11. Subshells și comenzi în linie

```
OUT=$(ls -l)
echo "$OUT"
```

12. Redirectări și logare

```
echo "OK" > out.txt          # suprascriere
echo "Error" >> log.txt      # adăugare
command > /dev/null 2>&1     # ignoră tot output-ul
```

13. Culori ANSI

```
RED='\e[31m'
NC='\e[0m'
echo -e "${RED}ERROR${NC}"
```

14. Verificări pe fișiere

```
[ -f file.txt ]     # fișier obișnuit
[ -d folder ]       # director
[ -r file ]         # citibil
[ -s file ]         # dimensiune > 0
```

15. Check comenzi/utilitare

```
if command -v curl &> /dev/null; then
    echo "curl found"
fi
```

16. Setări script portabil

```
#!/usr/bin/env bash
```

17. Date și timp

```
NOW=$(date +"%Y-%m-%d %H:%M:%S")
echo "Run at $NOW"
```

18. Timer / Delay

```
sleep 5  # pauză de 5 secunde
```

19. Menționare fișier sursă

```
source config.sh  # sau . config.sh
```

20. Debbuging

```
bash -x script.sh  # execuție cu urmărire
set -x             # activează debug din script
```

21. Shebang-uri alternative

- `/bin/bash` – clasic
- `/usr/bin/env bash` – portabil
- `/bin/sh` – compatibil POSIX

22. Script executabil global

```
chmod +x script.sh
mv script.sh /usr/local/bin/myscript
```

23. Testarea variabilelor

```
[ -z "$VAR" ]  # gol
[ -n "$VAR" ]  # nu e gol
```

24. Valori implicite (default)

```
echo "${NAME:-default}"  # dacă NAME nu e setat, folosește default
```

25. Timp execuție script

```
START=$(date +%s)
# ... script ...
END=$(date +%s)
echo "Duration: $((END - START)) seconds"
```

REGEX PRESETS

Pentru mailuri:

a. Email valid conform regulilor:

```
regex
CopyEdit
\b[a-zA-Z][a-zA-Z0-9._]*@[a-zA-Z]+\.[a-z]{2,4}\b
```

b. Email valid care se termină cu `.com`:

```
regex
CopyEdit
\b[a-zA-Z][a-zA-Z0-9._]*@[a-zA-Z]+\.com\b
```

> Poți folosi comenzi ca:

```
bash
CopyEdit
grep -E '\b[a-zA-Z][a-zA-Z0-9._]*@[a-zA-Z]+\.[a-z]{2,4}\b' company_emails.txt
```

2. Regex utile pentru diverse validări

Tip Validare	Regex	Descriere
Număr întreg	`^-?[0-9]+$`	Număr întreg (cu semn)
Număr zecimal	`^-?[0-9]*\.?[0-9]+$`	Număr float/zecimal
Număr telefon	`^\+?[0-9]{10,15}$`	Telefon internațional
Nume fișier	`^[a-zA-Z0-9_\-\.]+$`	Nume fișier valid (fără path)
IP valid	`^([0-9]{1,3}\.){3}[0-9]{1,3}$`	IPv4 simplu (nu verifică limite 0–255)
Linie goală	`^$`	Linie fără caractere

Numai litere	`^[a-zA-Z]+$`	Doar litere mari/mici
Numai cifre	`^[0-9]+$`	Doar cifre
Data simplă	`^[0-9]{4}-[0-9]{2}-[0-9]{2}$`	Format `YYYY-MM-DD`
URL simplu	`^https?://[a-zA-Z0-9.-]+\.[a-z]{2,4}(/.*)?$`	URL HTTP(S) simplificat
SHA1 hash	`^[a-f0-9]{40}$`	Hash SHA1
MAC address	`^([0-9A-Fa-f]{2}:){5}[0-9A-Fa-f]{2}$`	MAC hardware address

Script: creare X fișiere în directorul Y, fiecare de Z KB

Fișier: `create_files.sh`

```
#!/bin/bash

# Verificare argumente
if [ "$#" -ne 3 ]; then
    echo "Utilizare: $0 <numar_fisiere> <director> <dimensiune_kb>"
    exit 1
fi

X=$1  # număr fișiere
Y=$2  # director
Z=$3  # dimensiune în KB

# Creează directorul dacă nu există
mkdir -p "$Y"

# Creare fișiere
for i in $(seq 1 "$X"); do
    dd if=/dev/zero of="$Y/file_$i.txt" bs=1024 count="$Z" status=none
done

echo "Am creat $X fișiere în $Y, fiecare de $Z KB."
```

Rulare:

```
bash create_files.sh 5 /tmp/testdir 100
```

Script: verificare conectivitate la un IP + scriere rezultat

Fișier: `check_ping.sh`

```
#!/bin/bash

# Verificare argument
if [ "$#" -ne 1 ]; then
    echo "Utilizare: $0 <IP>"
    exit 1
fi

IP="$1"
```

```
# Verificare ping (1 pachet, 1s timeout)
if ping -c 1 -W 1 "$IP" > /dev/null 2>&1; then
   echo "DA" > ye\$\$
else
   echo "NU" > ye\$\$
fi
```

Rulare:

```
bash check_ping.sh 8.8.8.8
cat ye$$  # Afișează DA sau NU
```

```
#!/bin/bash

BASE="$HOME/acadnet"

# a) [4p] Creare ierarhie directoare: $HOME/acadnet/$FOO/$BAR
echo "[a] Creare ierarhie de directoare..."
for FOO in $(seq 1 1000); do
   for BAR in $(seq 1 10); do
      mkdir -p "$BASE/$FOO/$BAR"
   done
done

# b) [2p] Creare fișiere 'foobar' de 1 KB în $HOME/acadnet/$FOO
echo "[b] Creare fișiere foobar..."
for FOO in $(seq 1 1000); do
   dd if=/dev/zero of="$BASE/$FOO/foobar" bs=1024 count=1 status=none
done

# c) [2p] Creare link simbolic în $FOO/1 către $FOO/foobar
echo "[c] Creare link simbolic în subdirectorul /1..."
for FOO in $(seq 1 1000); do
   ln -s "../foobar" "$BASE/$FOO/1/link"
done

# d) [2p] Afișare spațiu utilizat
echo "[d] Spațiu utilizat:"
echo "Spațiu total în $BASE:"
du -sh "$BASE"

echo "Spațiu total pe filesystem-ul ce conține $HOME:"
df -h "$HOME" | awk 'NR==2 {print $3 " folosit din " $2}'
```

explicatii:

Punct	Ce face
(a)	Creează 1000 × 10 = 10.000 directoare, conform structurii `$HOME/acadnet/$FOO/$BAR`.
(b)	Creează câte un fișier `foobar` de 1KB în fiecare director `$FOO`.
(c)	În fiecare subdirector `$FOO/1` creează un link simbolic `link` care pointează spre `../foobar`.

(d)	Afișează spațiul ocupat de întreaga structură și sistemul de fișiere în care se află `$HOME`.

a) [3p] Configurare imprimantă virtuală cu CUPS-PDF – *"AcadNet Printer"*

1. **Instalare CUPS și driverul PDF**

```
sudo apt update
sudo apt install cups printer-driver-cups-pdf -y
```

1. **Pornește serviciul CUPS și asigură-te că rulează**

```
sudo systemctl enable cups
sudo systemctl start cups
```

1. **Adaugă imprimanta "AcadNet Printer"**

```
sudo lpadmin -p "AcadNet_Printer" -E -v cups-pdf:/ -m "CUPS-PDF.ppd"
```

> -E: activează imprimanta
>
> - `v cups-pdf:/`: imprimantă virtuală PDF
> - `m "CUPS-PDF.ppd"`: folosește driverul PDF

1. **Permite utilizatorilor să o folosească**

```
sudo lpadmin -p "AcadNet_Printer" -u allow:all
```

b) [4p] Adăugare pagină cu metadate la începutul fiecărui document

Vom folosi un *CUPS backend script sau un filter personalizat*. Mai simplu este un *filter* ce inserează pagină cu metadate.

1. **Creăm un filter custom:**

```
sudo mkdir -p /usr/lib/cups/filter
sudo nano /usr/lib/cups/filter/insert_acadnet_metadata.sh
```

1. **Conținutul scriptului:**

```
#!/bin/bash

# Variabile
PRINTER_NAME="$PRINTER"
JOB_ID="$JOB_ID"
CREATION_TIME=$(date -d @$JOB_TIME_CREATED)
PROCESS_TIME=$(date)

# Fișiere
TMPDIR=$(mktemp -d)
HEADER_PDF="$TMPDIR/header.pdf"
FINAL_PDF="$TMPDIR/final.pdf"

# Creează header cu informații
```

```
echo -e "Printer Name: $PRINTER_NAME\nJob ID: $JOB_ID\nCreated: $CREATION_TIME\nProcessed: $PROCESS_TIM
E" \
| enscript -B -o - | ps2pdf - "$HEADER_PDF"

# Combină header + documentul original
pdftk "$HEADER_PDF" "$6" cat output "$FINAL_PDF"

# Output final
cat "$FINAL_PDF"

# Cleanup
rm -rf "$TMPDIR"
exit 0
```

> $6 este fișierul original de print din pipeline-ul CUPS.

1. **Fă scriptul executabil:**

```
sudo chmod +x /usr/lib/cups/filter/insert_acadnet_metadata.sh
```

1. **Asociază scriptul cu imprimanta:**

```
sudo lpadmin -p "AcadNet_Printer" -o filter=/usr/lib/cups/filter/insert_acadnet_metadata.sh
```

---

c) [3p] Adăugare logo AcadNet pe prima pagină

Presupunem că ai un logo (`acadnet_logo.png`). Vom modifica scriptul de mai sus să-l insereze.

1. **Instalează `convert` (ImageMagick) și `pdftk` dacă nu sunt deja:**

```
sudo apt install imagemagick pdftk -y
```

1. **Modifică partea de generare a headerului în script:**

Înlocuiește linia:

```
echo -e "Printer Name: ..." | enscript ...
```

cu:

```
# Creează pagină text
echo -e "Printer Name: $PRINTER_NAME\nJob ID: $JOB_ID\nCreated: $CREATION_TIME\nProcessed: $PROCESS_TIM
E" \
> "$TMPDIR/text.txt"

enscript -B "$TMPDIR/text.txt" -o - | ps2pdf - "$TMPDIR/text.pdf"

# Adaugă logo
convert acadnet_logo.png -resize 100×100 "$TMPDIR/logo.pdf"

# Combină logo + text
pdftk "$TMPDIR/logo.pdf" "$TMPDIR/text.pdf" cat output "$HEADER_PDF"
```

> Logo-ul va apărea sus, urmat de metadatele pe prima pagină.
