Capitolul 2

Utilizarea sistemului de fișiere

2.1 Noțiuni de bază

Sistemele de fișiere par banale datorită familiarității lor. În fiecare zi lucrăm cu diverse fișiere, precum poze, texte sau melodii. De exemplu, le căutăm prin directoare, le deschidem, le modificăm, le închidem, le ștergem, le scoatem apoi din Recycle Bin deoarece le șterseserăm din greșeală, și așa mai departe. Dar sistemul de fișiere este una dintre componentele centrale ale sistemului de operare, care ne ajută să organizăm cantități impresionante de informații, procese și colaboratori.

- În primul rând, oferă posibilitatea **controlului unei cantități tot mai mari de documente**, permițându-ne să găsim un anumit fișier printre mii, milioane sau chiar miliarde de alte fișiere, în sistemele distribuite. Această contribuție este detaliată în Secțiunea 2.1.2 privind structura ierarhică.
- În al doilea rând, sistemele de fișiere **asigură separarea resurselor între utilizatorii multipli ai unui sistem de calcul**, fie cei umani sau non-umani. Vom discuta importanța acestei trăsături în Secțiunea 5.5 dedicată permisiunilor.

Sistemele de fișiere sunt diverse, **construite și optimizate pentru diferite contexte de utilizare**. Nu există un sistem de fișiere optim pentru toată lumea. Pentru a alege un sistem de fișiere trebuie să știm care sunt prioritățile în funcționarea sistemului și să acceptăm anumite compromisuri între cerințe contradictorii. De exemplu, creșterea resurselor de stocare a dus la dispariția crizelor de spațiu și a problematicii comprimării în sistemele personale; aceasta este însă tot mai relevantă în arhitecturile *cloud*. În zilele noastre, un utilizator stochează și folosește filme, poze, jocuri pe calculator, documente, mașini virtuale, arhive de informații; acestea ocupă spațiu considerabil pe un sistem laptop sau pe un dispozitiv mobil inteligent sau în Dropbox; acest spațiu ocupat nu mai este însă considerat o problemă dat fiind starea tehnologiilor și serviciilor de stocare.

De asemenea, cerința integrității datelor este foarte importantă în sistemele ce lucrează cu date critice, dar mai puțin importantă în sistemele personale, care au mai multă nevoie de simplitate și flexibilitate. Printre criteriile după care putem compara și alege sistemele de fișiere se numără:

- Asigurarea integrității datelor;
- Separarea eficientă a resurselor între diferiți utilizatori;

- Securizarea datelor prin setarea permisiunilor diferențiate de acces;
- Volumul gestionat: facilitatea în lucrul cu fișiere foarte mari sau cu un număr foarte mare de fișiere;
- Comprimarea fișierelor (*file compression*) pentru a maximiza spațiul de stocare pe disc (comprimarea înseamnă că aceleași date vor ocupa mai puțin spațiu, dar vor necesita efortul procesorului pentru a le decomprima la fiecare utilizare);
- Optimizarea spațiului de stocare prin gestiunea fișierelor duplicate și a zonelor ineficient scrise pe disc;
- Gestiunea posibilelor erori prin jurnalizare și reversibilitate, adică menținerea unei liste a modificărilor ce permite revenirea la o stare anterioară în cazul apariției unei erori.

2.1.1 Ce este un sistem de fișiere

Știm, intuitiv, ce sunt fișierele: sunt documentele noastre electronice, precum pozele, melodiile, proiectele pentru facultate sau programele executabile.

> Fișierul (*file*) reprezintă o formă de organizare digitală a informațiilor, având forma unei înșiruiri de octeți.

Informațiile sunt organizate în **fișiere** în vederea utilizării lor printr-o aplicație și a stocării lor de durată. În afara fișierelor create de utilizatorii umani, există și fișiere create de utilizatori automați. Unele dintre acestea sunt esențiale pentru funcționarea sistemului de calcul și sunt astfel ascunse de utilizatorii obișnuiți.

Fișierele sunt organizate la rândul lor în **directoare**.

> Un director (*folder* sau *directory*) reprezintă o colecție de fișiere și subdirectoare, identificată printr-un nume.

Dacă putem înțelege un fișier prin analogie cu o **foaie** pe care sunt scrise informații, putem înțelege un director prin analogie cu un **dosar** care conține file de hârtie dar și alte dosare. Un director, ca și un dosar, poate fi și gol.

Această analogie este utilă dar poate fi și înșelătoare. Dintr-o perspectivă tehnică, directoarele sunt tot fișiere. Ele nu „conțin” efectiv fișierele pe care le organizează, așa cum un dosar conține foi, ci doar numele lor - fiind similare cu o foaie pe care am scris o listă de documente. Prin urmare, directoarele în Linux sunt niște fișiere speciale care servesc organizării altor fișiere și directoare.

> **Înțeles pentru *folder* (director):** Conceptul mai general de director (*folder*), preluat din engleză și în limbajul nostru, se poate referi la forme de organizare a informațiilor care nu au corespondent în sistemul de fișiere. De exemplu, interfețele de email pot permite organizarea mesajelor pe *foldere*. Acestea nu vor fi regăsite în structura ierarhică a sistemului de fișiere, rămânând accesibile doar prin intermediul interfeței în care au fost create.

Fișierele reprezintă informații digitale inscripționate pe mediile fizice de stocare (hard disk, USB stick, DVD etc). Mediile de stocare pot fi considerate spații continue de octeți, pe care putem înscrie multe fișiere, de dimensiuni variabile. Pentru a putea citi sau scrie fișiere pe un mediu de stocare este necesar să cunoaștem sistemul de fișiere utilizat pentru organizarea acestuia.

Sistemul de fișiere este o parte a sistemului de operare ce se ocupă cu numele și atributele fișierelor, pe care le stochează într-o structură ierarhică. Sistemul de fișiere oferă o metodă de organizare fizică și logică a fișierelor într-un mediu de stocare:

- Stocarea fișierelor ca o înșiruire de octeți reprezintă organizarea fizică.
- Modul în care sunt adresate fișierele reprezintă organizarea logică.

Sistemul de fișiere ajută astfel utilizatorul să stocheze și să organizeze datele digitale pentru acces facil. Figura 2.1 prezintă rolul sistemului de fișiere în organizarea datelor; orgnizarea este de obicei ierarhică așa cum este prezentat în Secțiunea 2.1.2.

Fișierele sunt folosite de sistemul de operare pentru a organiza atât datele provenite de la utilizator, cât și cele generate de sistem. În sistemul de fișiere sunt stocate și organizate poze, documente, notițe ale utilizatorului, dar și fișiere de configurare, fișiere de tip jurnal (*log files*), baze de date necesare funcționării sistemului.

Deoarece este important ca utilizatorii să poată accesa fișierele stocate, sistemul de operare pune la dispoziție o interfață pentru a putea lucra cu sistemul de fișiere. În funcție de preferințe, există două tipuri de interfețe: de tip text (exemplu: interpretorul de comenzi) sau de tip grafic (exemplu: Windows File Explorer)

2.1.2 Structura ierarhică a sistemului de fișiere

De la an la an avem tot mai multe fișiere pe calculatoare și telefoane. Acumulăm poze, video-uri, melodii, precum și documente de la școală sau de la birou. Ce-ar însemna să găsim o poză într-o colecție de un milion de poze? Sistemul de fișiere ne vine într-ajutor printr-o inovație, și anume structurarea ierarhică a informației, înlocuind structura plată.

Vorbim despre o **structură plată** atunci când toate fișierele sunt într-un singur loc, adică un singur director. De exemplu, dacă un utilizator salvează toate documentele pe desktop, avem o structură plată. O structură **ierarhică**, sau arborescentă, apare atunci când fișierele sunt organizate în directoare (*folders*). Un director poate conține mai multe fișiere dar și alte directoare, fiecare dintre care poate conține mai multe fișiere dar și alte directoare și tot așa, până când ultimul director va conține doar fișiere sau va fi gol. Un director este analog unei crengi dintr-un arbore, pe care pot crește alte crengi dar și frunze (fișierele). Pe frunze nu crește nimic.

Să examinăm următorul exemplu. Pentru a găsi o poză anume (de exemplu, `selfiecumotanul-mai2018.jpg`) într-o colecție de 1.000.000 de poze, utilizatorul ar trebui să ceară sistemului de fișiere să parcurgă toate pozele până când găsește numele fișierului căutat. În cel mai rău caz, va trebui să parcurgă toate cele 1.000.000 de nume.

Ce se întâmplă însă dacă apelăm la o organizare pe trei niveluri, grupând pozele câte 100? Pe primul nivel, vom avea 100 de directoare. În fiecare dintre ele, includem 100 de

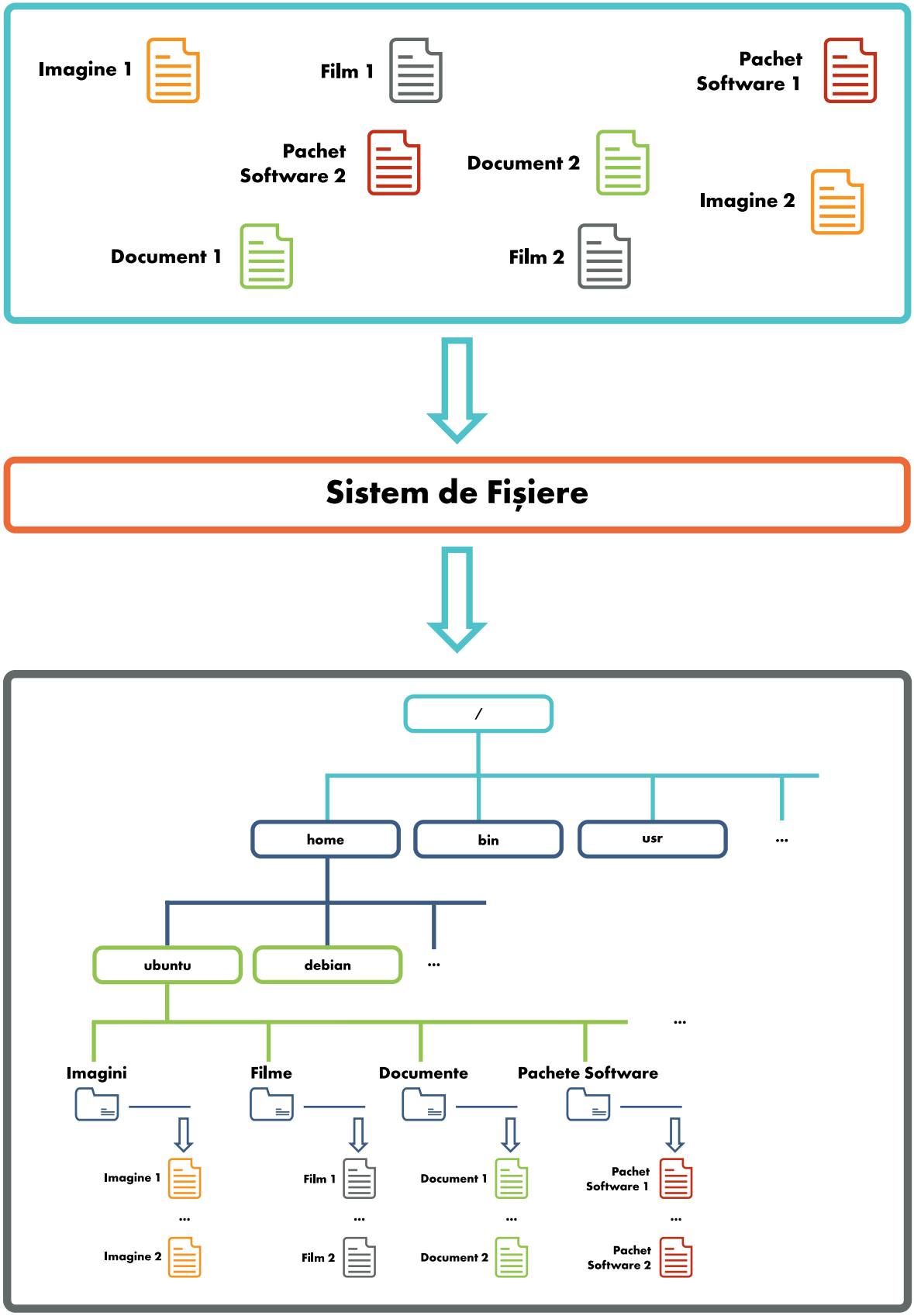

Figura 2.1: Sistemul de fișiere ca organizator de date digitale

subdirectoare. Apoi, în fiecare subdirector includem 100 de poze. Astfel, am stocat într-o ierarhie cu trei niveluri `100 directoare * 100 subdirectoare * 100 poze = 1.000.000 poze`.

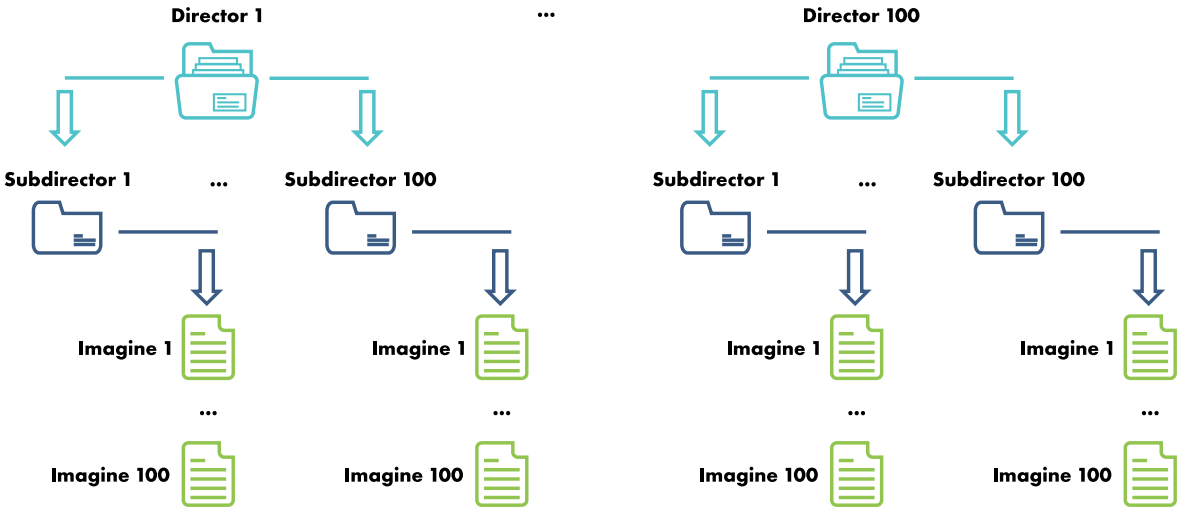

Figura 2.2: Exemplu de organizare ierarhică pe 3 niveluri a unui set de 1.000.000 poze

Cum putem găsi acum poza? Adresa ei ne va indica unde anume se află, în arborele de fișiere. De exemplu, dacă adresa sa completă este: Directorul `99/cat-pics/selfiecumotanul-mai2018.jpg`, sistemul de fișiere va căuta întâi directorul `99` printre cele 100 de directoare de pe primul nivel. Apoi, va căuta `cat-pics` printre cele 100 de subdirectoare din directorul 99. Apoi, în cele din urmă, va căuta poza dorită. În cel mai rău caz, sistemul de fișiere va realiza, de trei ori la rând, o căutare între 100 de elemente.

Prin structura arborescentă cu 3 niveluri am înlocuit deci o căutare într-un sac cu 1.000.000 elemente, pe care o singură persoană o realizează în ore dacă nu zile, cu o căutare în trei punguțe cu 100 de elemente fiecare, care este ușor de realizat chiar și manual, de o singură persoană.

Prin organizarea arborescentă pe directoare, cantități impresionante de fișiere pot fi gestionate la nivelul unei singure persoane. Sistemul de fișiere transformă astfel complexitatea volumului imens de informații și o face accesibilă pentru noi, utilizatorii umani. Mai mult despre găsirea fișierelor se află în Secțiunea 2.3.6 privind comenzile de căutare (**`find`**, **`locate`**, **`whereis`**, **`which`** și **`type`**).

Există însă un cost pe care îl plătim pentru acest control sporit al complexității. Fiecare director este un element suplimentar, creat de sistemul de fișiere, care trebuie să fie și el stocat undeva și ocupă astfel resurse. Aceste directoare nu există în structura plată, în care economisim astfel spațiu. În exemplul de mai sus, în structura ierarhică avem în total 100*100=10.000 directoare, create special pentru a organiza cele 1.000.000 de poze. Prin urmare, raportându-ne la numărul inițial de fișiere, pentru a controla mai bine informațiile plătim un cost de 10.000 / 1.000.000 = 1%.

Prezentăm în continuare exemple concrete de organizare a fișierelor în sistemele de operare moderne.

În Tabelul 2.1 este prezentată structura ierarhică în Linux.

Tabelul 2.1: Ierarhia într-un sistem de fișiere din mediul Linux

Director	Conținut
`/`	directorul rădăcină (*root directory*) – directorul cel mai cuprinzător, care conține celelalte directoare, considerate analoage trunchiului și ramurilor.
`/bin`	comenzi esențiale necesare bootării, întreținerii și depanării sistemului
`/boot`	fișiere necesare bootării, precum imaginea kernel-ului
`/dev`	fișiere speciale utilizate pentru accesul direct la dispozitivele hardware sau logice ale sistemului
`/etc`	fișiere pentru configurarea sistemului, precum `inittab`, `fstab` și `hosts`
`/home`	fișierele fiecărui utilizator din sistem - datele unui utilizator se găsesc în `/home/username`
`/media`	subdirectoare în care se montează unitățile optice, floppy etc.
`/mnt`	subdirectoare în care se montează alte sisteme de fișiere
`/opt`	pachete de aplicații de dimensiuni mari, accesibile tuturor utilizatorilor
`/proc`	sistem virtual de fișiere din care se obțin informații despre sistem și aplicațiile care rulează la un moment dat
`/root`	directorul home al utilizatorului root
`/sbin`	comenzi de bază accesibile numai utilizatorului root
`/tmp`	fișiere temporare
`/usr`	aplicații pentru uzul normal al sistemului de operare - `/usr/local` conține aplicațiile instalate/compilate de utilizator
`/var`	fișiere al căror conținut se schimbă foarte des, precum log-uri, fișiere temporare, cache (date reutilizabile), spool (date neprocesate)

În Figura 2.3 prezentăm cum arată grafic o ierarhie a sistemului de fișiere. Observăm că în rădăcină se află directoarele `home/`, `bin/`, `usr/` etc. În directorul `home/` se află subdirectoarele `ubuntu/` și `myuser/` ș.a.m.d.

> **Folosire caracter / (*slash*):** Atunci când folosim nume de directoare în Linux vom prefera să folosim sufixul `/` (*slash*) pentru a indica faptul că sunt directoare.

Ierarhia sistemului de fișiere în macOS este similară celei din Linux.

Structura fișierelor în Windows diferă față de cea din Linux. Aceasta este mai simplă și majoritatea directoarelor importante se află în `C:\Windows`, așa cum se observă în Tabelul 2.2.

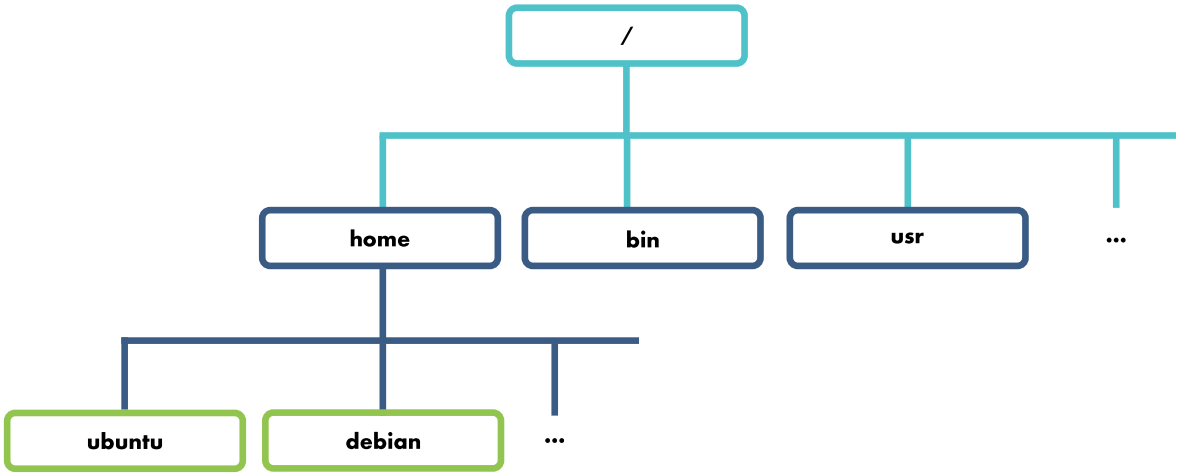

Figura 2.3: Ierarhia într-un sistem de fișiere din mediul Linux

Tabelul 2.2: Ierarhia într-un sistem de fișiere din Windows

Director	**Conținut**
`C:\`	directorul rădăcină
`C:\Windows`	Windows-ul și fișierele aferente
`C:\Documents and Settings`	configurările utilizatorilor și date specifice acestora
`C:\Program Files`	aplicații
`C:\Windows\System32`	drivere și fișiere de configurare Windows
`C:\Documents and Settings\username\My Documents`	datele unui utilizator (aceasta este calea implicită, ea poate fi modificată)

Deși în Linux și în macOS avem un singur director rădăcină, Windows are simultan pentru fiecare sistem de fișiere câte un director rădăcină:

- `A`, `B`: de obicei sunt rezervate pentru floppy disk-uri
- `C`: partiția de pe hard disk; pot exista mai multe, cărora li se asociază litere în ordine
- `D` (sau următoarea literă disponibilă după partițiile de pe hard disk-uri): se referă la CD-ROM/DVD-Rom

Windows alocă literele în funcție de partiții, nu după sistemul de fișiere. Dacă se modifică sistemul de fișiere de pe o partiție, litera asignată partiției va rămâne aceeași. Pe o partiție se poate afla la un moment dat un singur sistem de fișiere.

În Tabelul 2.3 de mai jos avem o comparație între căile importante din sistemele de operare cele mai cunoscute.

Observăm că în Linux înșiruirea de directoare este separată de caracterul `/` (*slash*), în timp ce în mediile Windows se folosește `\` (*backslash*).

Tabelul 2.3: Comparație între căile sistemelor de operare

Descriere	Windows	Linux	Mac OS
rădăcină	`C:`	`/`	`/`
director home	`C:\Documents and Settings\username`	`/home/username`	`/Users/username`
aplicații	`C:\Program Files`	`/bin`; `/sbin`; `/usr/bin`; `/usr/sbin`; `/usr/local/bin`;	`/opt/*/bin` `/Applications`; `/bin`; `/sbin`
configurări ale sistemului	Windows Registry	directoare specifice fiecărei aplicații, aflate în home-ul utilizatorului; `/etc`	`/Users/username/Library`; `/etc`

2.1.3 Căi relative și căi absolute

Revenind la exemplul din Figura 2.3, dorim să găsim poza `selfiecumotanul-mai2018.jpg`, care se află în subdirectorul `cat-pics`, în directorul `99`. Pentru a putea accesa poza, avem două posibilități: putem sa folosim o cale relativă sau o cale absolută. Alegerea căii pe care o vom folosi depinde de unde ne aflăm în momentul respectiv în ierarhia de fișiere și unde se află fișierul căutat. O cale este echivalentul unei adrese pentru identificarea fișierului.

Calea absolută reprezintă adresa completă a fișierului, începând cu directorul rădăcină. Astfel, o cale absolută va începe cu `/` (*slash*) sau `~` (tildă, *tilde*) în cazul Linux/macOS sau cu `C:`, `D:` etc, în cazul Windows.

Calea relativă este o cale ce pornește din directorul curent. Pornind din directorul curent se construiește o cale către fișierul destinație dorit. O cale relativă **nu** începe cu `/` (*slash*) sau `~` (tildă, *tilde*) în cazul Linux/macOS sau cu `C:`, `D:` etc, în cazul Windows.

Directorul *home*: Simbolul `~` este o prescurtare în Linux/macOS pentru directorul *home* al utilizatorului. De obicei acesta este `/home/<username>/` în Linux și în `/Users/<username>/` în macOS.

În fiecare director se găsesc **două directoare speciale**: `.` (punct) și `..` (punct punct). Directorul `.` (*punct*) indică spre același director, directorul curent. `..` (*punct punct*) indică spre directorul părinte în ierarhia de fișiere și directoare.

Pentru a ne întoarce în ierarhia de fișiere pas cu pas, ne folosim de `..` pentru a ajunge în directorul părinte. Putem să înlănțuim mai multe grupări `..` (de ex. `../../..`) pentru a ne întoarce mai sus în ierarhie. Avem un exemplu de comenzi mai jos:

```
1  student@uso:~$ pwd
2  /home/student
3  student@uso:~$ cd uso.git/labs/
4  student@uso:~/uso.git/labs$ pwd
5  /home/student/uso.git/labs
6  student@uso:~/uso.git/labs$ cd ..
7  student@uso:~/uso.git$ pwd
8  /home/student/uso.git
9  student@uso:~/uso.git$ cd ../..
10 student@uso:/home$ pwd
11 /home
12 student@uso:/home$ cd
13 student@uso:~$ pwd
14 /home/student
15 student@uso:~$ cd /usr/local
16 student@uso:/usr/local$ pwd
17 /usr/local
```

În exemplu de mai sus am folosit comanda **`pwd`** (*print working directory*) pentru a afișa directorul curent (numit și director de lucru). Directorul este indicat și în promptul comenzii, între caracterul `:` (două puncte) și caracterul `$` (dolar); simbolul ∼ (tildă) este echivalent pentru directorul home al utilizatorului, în cazul de mai sus `/home/student/`. Comanda **`cd`** (*change directory*) schimbă directorul curent; comanda primește ca argument o cale care poate fi cale relativă sau absolută.

La linia 6 din exemplu de mai sus am schimbat directorul de lucru în directorul părinte folosind comanda **`cd ..`** și am urcat un nivel în ierarhia de directoare; observăm că este afișată o cale mai scurtă la linia 8 (`/home/student/uso.git`) față de calea anterioară de la linia 5 (`/home/student/uso.git/labs`). La linia 9 este o cale relativă (`../..`) care urcă două niveluri în ierarhie. La linia 12 avem comanda **`cd`** fără argumente; această folosire a comenzii schimbă directorul de lucru în directorul home al utilizatorului, în cazul nostru `/home/student/`. La linia 15 avem o cale absolută (care începe cu `/` (*slash*)): `/usr/local`; comanda **`cd`** primește ca argument această cale și schimbă directorul curent în `/usr/local`.

Până acum am prezentat doar când folosim directorul special `..` (punct punct).

De multe ori, folosim construcția `.` (punct), care indică directorul curent, pentru comenzi ce execută script-uri/programe din acel director. Dacă în directorul curent avem un executabil numit `list_permissions`, atunci îl vom putea rula folosind comanda:

```
1  student@uso:~$ ./list_permissions
```

Aceasta înseamnă să se execute executabilul `list_permissions` din directorul curent.

Construcția `.` (punct) poate fi folosită în situații în care dorim să referim directorul curent. De exemplu dacă dorim să copiem un fișier în directorul curent rulăm comanda de mai jos:

```
1  student@uso:~$ pwd
2  /home/student
3  student@uso:~$ ls
4  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
     examples.desktop  uso.git  vm-actions-log.txt
5  student@uso:~$ cp /etc/passwd .
6  student@uso:~$ ls
```

```
7 Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
    examples.desktop  passwd  uso.git  vm-actions-log.txt
```

În exemplu de mai sus am folosit comanda **`ls`** (*list*) pentru a afișa conținutul directorului curent (chiar directorul home `/home/student`). Apoi, la linia 5 am folosit comanda **`cp`** (*copy*) pentru a copia fișierul `/etc/passwd` (cale absolută), dat ca primul argument, în directorul curent, reprezentat de construcția `.` (punct), dat ca al doilea argument. Apoi, la afișarea conținutului directorului (folosind comanda **`ls`**) vedem prezența fișierului `passwd`, acum copiat. În exemplu am folosit construcția `.` (punct) pentru a referi directorul curent, destinația comenzii de copiere.

2.2 Formatul fișierelor

Din perspectiva utilizatorului, fișierele se împart în diverse categorii, precum muzică, poze, jocuri și altele. Toate acestea sunt văzute de calculator ca o colecție de biți ce trebuie prelucrați pentru a putea fi redați. Calculatorul prelucrează fișierele în funcție de **formatul** acestora, pentru a ști ce programe sunt necesare pentru a le putea deschide și pentru a putea lucra cu ele.

Pentru început, putem clasifica fișierele în două mari categorii: fișiere de tip text (*text file*) și fișiere binare (*binary file*).

- **Fișierele de tip text** conțin linii compuse din caractere citibile (litere, cifre, semne de punctuație) fără să conțină elemente ce trebuie să fie interpretate de un program (precum grafice, cod executabil etc). Fișierele text pot conține text simplu (*plain text*), având extensia `.txt`, sau care conți cod sursă (de exemplu cu extensia `.c` sau `.java`) sau formate de prezentare precum HTML;
- **Fișierele binare** sunt toate fișierele care nu sunt de tip text, putând reprezenta: imagini, programe executabile, melodii, fișiere comprimate etc.

Formatul (sau tipul) fișierelor se referă la modul de codificare a informației în fișier, care permite apoi redarea sau utilizarea informației prin intermediul unei interfețe sau a unei aplicații. Formatul fișierului specifică felul în care informația va fi codificată în biți, în mediul digital.

Formatul fișierului este, de regulă, asociat cu **extensia** acestuia. Extensia reprezintă sufixul de la finalul numelui fișierului, separat de numele fișierului printr-un punct. Exemple de extensii sunt: `.txt` (fișiere text), `.tex` (document sursă LaTeX), `.mp3` (format audio), `.bmp` (imagine tip *bitmap*), `.png` (imagine tip *Portable Network Graphic*), etc.

Este important să remarcăm că, **dacă schimbăm manual extensia unui fișier, nu înseamnă că i-am schimbat tipul**. Formatul fișierului ține de proprietățile conținutului acestuia și nu se schimbă dacă modificăm numele sau extensia. Dacă, de exemplu, avem un fișier text în care am pus versurile cântecului „The Kinslayer” al formației Nightwish, și schimbăm numele fișierului din `kinslayer.txt` în `kinslayer.mp3`, fișierul text nu devine dintr-o dată cântec și, desigur, nu îl vom putea deschide cu o aplicație de tip MP3 player.

Fișierele pot fi convertite dintr-un format într-altul, de regulă în interiorul categoriilor mari de conținut. De exemplu, putem converti un fișier `.wav` într-un fișier `.mp3`, sau un fișier `.bmp` într-un fișier `.png`, sau un fișier `.doc` într-un fișier `.pdf`. Este posibil să convertim și fișiere în formate ce par foarte diferite, de exemplu un fișier `.txt` într-un fișier `.mp3`, apelând la soluții automate text-to-speech. Convertirea se realizează nu prin schimbarea manuală a extensiei, ci prin intermediul unei aplicații cu care deschidem fișierul și îl transformăm în formatul dorit.

Pentru a vedea de ce tip este un fișier în Linux, dincolo de extensie, folosim comanda **`file`**. Mai jos avem exemple de folosire a comenzii **`file`** pentru determinarea tipului unui fișier. Pentru fiecare fișier primit ca argument de comanda **`file`** sunt indicate informații despre tipul fișierului: executabil, imagine de kernel, script shell, fișier comprimat, fișier imagine.

```
1  student@uso:~$ file /bin/ls
2  /bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
   dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/
   Linux 3.2.0, BuildID[sha1]=9567f9a28e66f4d7ec4baf31cfbf68d0410f0ae6,
   stripped
3  student@uso:~$ file /boot/vmlinuz-4.15.0-29-generic
4  /boot/vmlinuz-4.15.0-29-generic: Linux kernel x86 boot executable bzImage
   , version 4.15.0-29-generic (buildd@lgw01-amd64-057) #31-Ubuntu SMP Tue
   Jul 17 15:39:52 UTC 2018, RO-rootFS, swap_dev 0x7, Normal VGA
5  student@uso:~$ file /etc/init.d/ssh
6  /etc/init.d/ssh: POSIX shell script, ASCII text executable
7  student@uso:~$ file /usr/share/man/man1/cp.1.gz
8  /usr/share/man/man1/cp.1.gz: gzip compressed data, max compression, from
   Unix
9  student@uso:~$ file /usr/share/pixmaps/htop.png
10 /usr/share/pixmaps/htop.png: PNG image data, 128 x 128, 8-bit/color RGBA,
    non-interlaced
```

2.2.1 Detecția formatului fișierelor

Pe sistemele Windows fiecare fișier are o extensie, ce ne indică formatul acestuia și astfel ajută sistemul de operare să determine ce aplicație să utilizeze pentru a deschide fișierul. Sistemele Linux folosesc comanda **`file`** pentru a afla formatul unui fișier. Comanda **`file`** lucrează independent de extensia fișierului. Acest lucru este avantajos deoarece extensia poate să fie atribuită greșit de către utilizator, precum putem vedea și în exemplul de mai jos.

```
1  student@uso:~$ file photo.jpg
2  photo.jpg: JPEG image data, JFIF standard 1.01
3  student@uso:~$ mv photo.jpg fisier.txt # schimbarea numelui din photo.jpg
    in fisier.txt (schimba extensia din jpg in txt la nivel superficial
4  student@uso:~$ file fisier.txt
5  fisier.txt: JPEG image data, JFIF standard 1.01
```

2.2.2 Atributele fișierelor

Pentru a controla funcționarea sistemelor de fișiere este importantă distincția dintre **date** și **metadate**. Un fișier conține efectiv informații sau date, precum muzică, versuri, imagini

sau bilete de avion.

> **Metadatele** sunt informații despre informații: cantitatea informațiilor, data ultimei accesări, cine le-a creat, cine le-a modificat, unde anume au fost editate ultima dată.

Această distincție este des utilizată datorită controverselor tot mai dure privind uzul, abuzul și protecția datelor personale. Momentul de glorie al metadatelor a fost, probabil, interviul din 2014 cu generalul Michael Hayden, fostul director al National Security Agency și al Central Intelligence Agency ale Statelor Unite, în care acesta a declarat: „We kill people based on metadata". Folosirea în domeniul militar și al securității naționale este doar un exemplu al uzului și abuzului metadatelor în profilarea diferitelor persoane individuale sau a tipurilor de cetățeni și clienți.

Sistemele de fișiere se bazează și ele pe crearea și gestionarea metadatelor referitoare la fișiere. Aceste metadate poartă denumirea de **atribute**. Ele fac posibilă accesarea eficientă și securizată a informațiilor.

În Linux putem afișa atribute ale fișierelor folosind comanda **`ls`** cu opțiunea `-l`; opțiunea înseamnă *long listing* și este folosită pentru afișarea detaliată (cu atribute a fișierelor), ca mai jos:

```
student@uso:~$ ls -l
total 60
drwxr-xr-x  2 student student 4096 aug  6 17:41 Desktop
drwxr-xr-x  3 student student 4096 aug 20 21:00 Documents
drwxr-xr-x  2 student student 4096 aug  6 17:41 Downloads
drwxr-xr-x  2 student student 4096 aug  6 17:41 Music
drwxr-xr-x  2 student student 4096 aug  6 17:41 Pictures
drwxr-xr-x  2 student student 4096 aug  6 17:41 Public
drwxr-xr-x  2 student student 4096 aug  6 17:41 Templates
drwxr-xr-x  2 student student 4096 aug  6 17:41 Videos
-rw-r--r--  1 student student 8980 aug  6 17:37 examples.desktop
-rw-r--r--  1 student student 2506 sep 30 11:28 passwd
drwxr-xr-x 15 student student 4096 aug 24 14:52 uso.git
-rw-r--r--  1 student student 4827 aug 21 14:37 vm-actions-log.txt
```

Cele mai simple atribute ale unui fișier sunt numele, dimensiunea și tipul. Tipul fișierului este indicat de primul caracter din rezultatul rulării comenzii **`ls -l`**, și poate fi:

- `-` = regular file
- `b` = block special file
- `c` = character special file
- `d` = directory
- `l` = symbolic link
- `n` = network file
- `p` = fIFO
- `s` = socket

În rularea de mai sus avem fișiere (primul caracter este `-`) și directoare (primul caracter este `d`).

Alte atribute importante descriu permisiunile, adică operațiile pe care diferite tipuri de utilizatori le pot realiza asupra respectivului fișier. Informații detaliate privind **tipurile de utilizatori** și configurarea permisiunilor acestora se regăsesc în Capitolul 5.

- `r` = permisiunea de a citi fișierul
- `w` = permisiunea de a scrie în fișier
- `x` = permisiunea de a executa fișierul
- `-` = absența permisiunii

2.3 Operații uzuale asupra fișierelor și directoarelor

Sistemul de fișiere ne permite realizarea mai multor tipuri de operațiuni asupra fișierelor. Operațiile uzuale asupra fișierelor includ: afișarea și schimbarea directorului, afișarea conținutului fișierului, listarea fișierelor dintr-un director, crearea fișierelor sau a directoarelor, copierea, mutarea, redenumirea sau ștergerea acestora, precum și arhivarea/dezarhivarea și realizarea unei versiuni de backup.

O problemă poate apărea dacă doi utilizatori doresc să citească sau să modifice același fișier simultan, deoarece nu este clar dacă acțiunile celor doi nu se vor încurca reciproc. De aceea, una dintre responsabilitățile sistemului de fișiere este să mențină separarea resurselor resurselor între utilizatori. Vom prezenta detaliat aceste aspecte în Capitolul 5.

2.3.1 Afișarea și schimbarea directorului curent

Pentru a afișa directorul curent folosim comand **`pwd`** (*print working directory*). De asemenea, dacă nu s-au efectuat manual schimbări asupra prompt-ului bash, aceste va afișa implicit directorul în care ne aflăm. Dacă dorim să ne mutăm în alt director, vom folosi comanda **`cd`** <**`cale`**>. Comanda **`cd`** are ca parametru o cale absolută sau relativă către destinația în care vrem să ajungem.

Pentru a înțelege mai bine cum funcționează cele două comenzi – **`pwd`** și **`cd`** – avem următorul exemplu, mai jos. Urmăriți modificările prompt-ului bash atunci când schimbăm directorul:

```
1  student@uso:~$ pwd
2  /home/student
3  student@uso:~$ cd ..
4  student@uso:/home$ pwd
5  /home
6  student@uso:/home$ cd ../usr/bin/
7  student@uso:/usr/bin$ pwd
8  /usr/bin
9  student@uso:/usr/bin$ cd .
10 student@uso:/usr/bin$ pwd
11 /usr/bin
12 student@uso:/usr/bin$ cd /
13 student@uso:/$ pwd
14 /
15 student@uso:/$ cd
```

```
16 student@uso:~$ pwd
17 /home/student
18 student@uso:~$ cd /usr/bin
19 student@uso:/usr/bin$ pwd
20 /usr/bin
21 student@uso:/usr/bin$ cd /home
22 student@uso:/home$ pwd
23 /home
24 student@uso:/home$ cd -
25 /usr/bin
26 student@uso:/usr/bin$ pwd
27 /usr/bin
28 student@uso:/usr/bin$ cd ~
29 student@uso:~$ pwd
30 /home/student
31 student@uso:~$ cd ./././.
32 student@uso:~$ pwd
33 /home/student
34 student@uso:~$ cd ../..
35 student@uso:/$ pwd
36 /
```

În exemplu de mai sus:

- **`cd ..`** (comanda de la linia 3) ne întoarce în directorul părinte
- **`cd ../usr/bin`** (comanda de la linia 6) este o cale relativă care pornește din directorul părinte al directorului curent
- **`cd .`** (comanda de la linia 9) nu modifică directorul, deoarece **`.`** (punct) face referire la directorul curent
- **`cd ~`** (comanda de la linia 28) ne deplasează în directorul home al utilizatorului curent; ~ (tildă) este echivalentul directorului home pentru utilizatorul curent
- **`cd`** (comanda de la linia 15) schimbă directorul tot în directorul home
- **`cd -`** (comanda de la linia 24) ne întoarce în directorul în care ne aflam anterior

2.3.2 Listarea fișierelor

Acum că știm să navigăm dintr-un director în altul, ne interesează să afișăm conținutul acestora. Comanda pe care o folosim este **`ls [opțiuni] <cale>`**. Dacă dorim să listăm conținutul directorului curent, executăm comanda **`ls`** fără a mai fi nevoie să specificăm calea.

În continuare găsim opțiunile folosite frecvent pentru această comandă:

- `-l` = afișează detalii despre fiecare director/fișier, precum dimensiunea, utilizator, grup, data modificare, drepturi de acces
- `-a` = afișează toate fișierele, inclusiv cele ascunse (cele care încep cu **`.`** (caracterul punct))
- `-h` = afișează dimensiunea fișierelor în format ușor de înțeles, respectiv dimensiunea în octeți este înlocuită cu dimensiunea în Kocteți/Mocteți/Gocteți dacă depășește un anumit ordin de mărime.

Exemplu de output atunci când utilizăm opțiunea `-l`:

```
student@uso:~$ ls -l uso.git/README.md
-rw-r--r-- 1 student student 146 aug 20 20:57 uso.git/README.md
```

Explicații privind formatul de output:

- primul caracter ne spune tipul fișierului, în cazul de față fișier obișnuit
- următoarele 3 grupuri de caractere (`rwx`) reprezintă permisiunile de acces, pe care le vom detalia în Capitolul 5
- `student` este utilizatorul fișierului
- `student` (a doua apariție) este grupul de care aparține fișierul
- `146` este dimensiunea fișierului, în octeți
- `aug 20 20:57` este data ultimei modificări a fișierului
- `20:28` este ora ultimei modificări
- `uso.git/README.md` este numele fișierului

Exemplele următoare conțin câteva modele de funcționare a comenzii **`ls`**:

```
student@uso:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
  examples.desktop  passwd  uso.git  vm-actions-log.txt
student@uso:~$ ls ~
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
  examples.desktop  passwd  uso.git  vm-actions-log.txt
student@uso:~$ ls -l
total 60
drwxr-xr-x  2 student student 4096 aug  6 17:41 Desktop
drwxr-xr-x  3 student student 4096 aug 20 21:00 Documents
drwxr-xr-x  2 student student 4096 aug  6 17:41 Downloads
drwxr-xr-x  2 student student 4096 aug  6 17:41 Music
drwxr-xr-x  2 student student 4096 aug  6 17:41 Pictures
drwxr-xr-x  2 student student 4096 aug  6 17:41 Public
drwxr-xr-x  2 student student 4096 aug  6 17:41 Templates
drwxr-xr-x  2 student student 4096 aug  6 17:41 Videos
-rw-r--r--  1 student student 8980 aug  6 17:37 examples.desktop
-rw-r--r--  1 student student 2506 sep 30 11:28 passwd
drwxr-xr-x 15 student student 4096 aug 24 14:52 uso.git
-rw-r--r--  1 student student 4827 aug 21 14:37 vm-actions-log.txt
student@uso:~$ ls -lh
total 60K
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Desktop
drwxr-xr-x  3 student student 4,0K aug 20 21:00 Documents
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Downloads
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Music
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Pictures
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Public
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Templates
drwxr-xr-x  2 student student 4,0K aug  6 17:41 Videos
-rw-r--r--  1 student student 8,8K aug  6 17:37 examples.desktop
-rw-r--r--  1 student student 2,5K sep 30 11:28 passwd
drwxr-xr-x 15 student student 4,0K aug 24 14:52 uso.git
-rw-r--r--  1 student student 4,8K aug 21 14:37 vm-actions-log.txt
student@uso:~$ ls -a
```

```
34 .              .bash_history  .config   .gconf      .local    .
   sudo_as_admin_successful  .vboxclient-display.pid      .viminfo
   Downloads  Templates         uso.git
35 ..             .bash_logout   .dbus     .gitconfig  .mozilla  .tmux
                        .vboxclient-draganddrop.pid  .vimrc     Music
   Videos            vm-actions-log.txt
36 .ICEauthority  .bashrc        .emacs    .gnupg      .profile  .tmux.conf
                   .vboxclient-seamless.pid     Desktop    Pictures
   examples.desktop
37 .bash_aliases  .cache         .emacs.d  .lesshst    .ssh      .vboxclient
   -clipboard.pid  .vim                          Documents  Public     passwd
38 student@uso:~$ ls ../..
39 bin   cdrom  etc   initrd.img      lib    lib64   lost+found  mnt  proc
   run   snap  swapfile  tmp  var      vmlinuz.old
40 boot  dev    home  initrd.img.old  lib32  libx32  media       opt  root
   sbin  srv   sys       usr  vmlinuz
41 student@uso:~$ ls -a /
42 .   bin   cdrom  etc   initrd.img      lib    lib64   lost+found  mnt
   proc  run   snap  swapfile  tmp  var      vmlinuz.old
43 ..  boot  dev    home  initrd.img.old  lib32  libx32  media       opt
   root  sbin  srv   sys       usr  vmlinuz
```

În exemplu de mai sus

- comanda de la linia 19 (**`ls -l`**) afișează toate informațiile (`-l`) ale tuturor fișierelor, ascunse și neascunse, din directorul curent; dimensiunea fișierelor este afișată în format inteligibil (*human-readable*) (`-h`); observăm că sunt sunt afișate și directoarele standard **`.`** și **`..`**
- comanda de la linia 38 primește o cale relativă pentru a-i afișa conținutul
- comanda de la linia 41 afișează toate fișierele, ascunse și neascunse, din rădăcina ierarhiei de fișiere și directoare (`/`)

O altă opțiune folosită este `-R`, care afișează recursiv directoarele și fișierele ce au ca rădăcină directorul dat ca argument. Prin recursiv înțelegem că trece prin toate directoarele pornind de la directorul dat ca argument.

```
1  student@uso:~$ ls -R /usr/local/lib/
2  /usr/local/lib/:
3  python2.7  python3.6
4
5  /usr/local/lib/python2.7:
6  dist-packages  site-packages
7
8  /usr/local/lib/python2.7/dist-packages:
9
10 /usr/local/lib/python2.7/site-packages:
11
12 /usr/local/lib/python3.6:
13 dist-packages
14
15 /usr/local/lib/python3.6/dist-packages:
```

Dacă dorim să folosim alte opțiuni, putem sa consultăm **`man ls`** sau **`ls --help`**.

2.3.3 Afișarea conținutului fișierelor

Pentru a putea vedea ce conține un fișier, folosim comanda **cat** <**nume fișier**>. Un exemplu de utilizare este:

```
1 student@uso:~$ cat /etc/default/saned
2 # Defaults for the saned initscript, from sane-utils
3
4 # To enable under systemd please read README.Debian
5 # Set to yes to start saned under SysV
6 RUN=no
7
8 # Set to the user saned should run as
9 RUN_AS_USER=saned
```

Comanda va întoarce conținutul întregului fișier, indiferent de lungime. Pentru ca utilizatorul să poată naviga prin tot output-ul comenzii, putem folosi comanda **less**. Comanda **less** are o interfață asemănătoare cu cea a editorului **vim** și permite navigarea în ambele direcții, linie cu linie. O folosim ca mai jos, dându-i ca argument calea către fișierul al cărui conținut dorim să-l afișăm:

```
1 student@uso:~$ less /etc/X11/Xsession
```

2.3.4 Crearea fișierelor/directoarelor

Există doua metode pentru a crea un fișier gol. Prima metoda este utilizarea comenzii **touch** <**nume fișier**>, iar a doua se bazează pe o funcționalitate a shell-ului numită redirectare în fișier:

```
1 student@uso:~/Downloads$ touch cats.txt
2 student@uso:~/Downloads$ > dogs.txt
3 student@uso:~/Downloads$ ls
4 cats.txt  dogs.txt
5 student@uso:~/Downloads$ ls -l
6 total 0
7 -rw-rw-r-- 1 student student 0 sep 30 12:20 cats.txt
8 -rw-rw-r-- 1 student student 0 sep 30 12:20 dogs.txt
```

În exemplul de mai sus:

- comanda de la linia 1 folosește prima metodă; creează un fișier gol cu numele `cats.txt` în directorul curent
- comanda de la linia 2 folosește a doua metodă; redirectează (>) ieșirea unei comenzi nule într-un fișier, ducând la trunchierea fișierului, dacă există, sau, în cazul nostru, ducând la crearea fișierului gol `dogs.txt`

Vedem din rezultatul rulării comenzii `ls -l` că ambele fișiere sunt create cu dimensiune 0, sunt goale.

Comanda **touch** poate fi folosită și pentru a actualiza data ultimei accesări sau modificări asupra fișierului. În exemplu de mai jos folosim comanda touch pentru a schimba data ultimei modificări din `aug 20 20:57` în `sep 30 12:24`.

```
student@uso:~$ ls -l uso.git/README.md
-rw-r--r-- 1 student student 146 aug 20 20:57 uso.git/README.md
student@uso:~$ touch uso.git/README.md
student@uso:~$ ls -l uso.git/README.md
-rw-r--r-- 1 student student 146 sep 30 12:24 uso.git/README.md
```

Exemplu următor prezintă crearea legăturilor simbolice, un mod prin care același fișier poate fi referit din două căi, echivalentul scurtăturilor.

```
student@uso:~$ cat uso.git/README.md
uso
===

   * Directorul ''lab02'' contine toate fisierele si structura de
directoare necesare rezolvarii laboratorului 2 de catre studenti
student@uso:~$ ln -s uso.git/README.md readme_link
student@uso:~$ ls -l readme_link
lrwxrwxrwx 1 student student 17 sep 30 12:28 readme_link -> uso.git/
README.md
student@uso:~$ cat readme_link
uso
===

   * Directorul ''lab02'' contine toate fisierele si structura de
directoare necesare rezolvarii laboratorului 2 de catre studenti
```

În exemplul de mai sus am creat intrarea `readme_link` care este o legătură simbolică spre fișierul **`uso.git/README.md`**. Dacă afișăm, folosind comanda **`cat`**, fișierul sau legătura simbolică, obținem același conținut: legătura simbolică referă același fișier.

Legăturile (*link-urile*) numite și legături simbolice (*symbolic links*) sunt create cu ajutorul comenzii `ln` căreia îi transmitem ca argument opțiunea `-s`.

Pentru a crea un director folosim comanda **`mkdir`** ca mai jos:

```
student@uso:~$ mkdir games
student@uso:~$ ls -d games/
games/
student@uso:~$ ls -ld games/
drwxrwxr-x 2 student student 4096 sep 30 17:28 games/
student@uso:~$ ls games/
student@uso:~$ mkdir games/warcraft
student@uso:~$ mkdir games/lol
student@uso:~$ ls games/
lol  warcraft
student@uso:~$ mkdir -p games/heroes/3/nighon/mutare
student@uso:~$ ls -ld games/heroes/3/nighon/mutare
drwxrwxr-x 2 student student 4096 sep 30 17:32 games/heroes/3/nighon/
mutare
```

În exemplul de mai sus am creat directorul `games/` iar în cadrul său am creat subdirectoarele `warcraft/` și `lol/`. Comanda **`ls`** afișează conținutul unui diretor; dacă dorim să afișăm informații despre director în sine folosim comanda opțiunea `-d` a comenzii **`mkdir`**; o formă frecventă este folosirea opțiunii `-ld` pentru afișarea informațiilor despre director în format lung, cu atribute ale directorului. În partea finală a exemplului am folosit opțiunea `-p` a comenzii **`mkdir`** pentru a crea un director cu toate directoarele părinte necesare chiar dacă acestea nu există.

În Tabelul 2.4 sunt sumarizate comenzile de creare a tipurilor de intrări în sistemul de fișiere.

Tabelul 2.4: Comenzi pentru crearea intrărilor în sistemul de fișere

Tip de intrare	**Comandă**
Fișier normal	**`touch <nume_fișier>`** sau **`> <nume_fișier>`**
Director	**`mkdir <nume_director>`**
Legături (link-uri)	**`ln -s <destinatie> [<nume_legătură>]`**

2.3.5 Copiere / mutare / redenumire / ștergere

Operațiile frecvente cu fișiere sunt copierea, mutarea sau redenumirea și ștergerea. Copierea înseamnă că un fișier sau director este duplicat în alt loc, și se găsește acum în două locuri. Mutarea/Redenumirea înseamnă cu un fișier sau director este acum mutat din locul său inițial în alt loc în sistemul de fișiere; spre deosebire de copiere, acum fișierul/directorul se găsește într-un singur loc. Ștergerea înseamnă eliminarea acelui fișier sau director din sistemul de fișiere; nu se va mai găsi în nici un loc.

Comenzile de copiere și mutare primesc doi parametri:

- primul parametru reprezintă sursa de unde dorim să copiem/mutăm
- al doilea parametru reprezintă destinația unde dorim să copiem/mutăm

> **Observație:** <sursa> comenzilor din acest subcapitol poate fi reprezentată și prin globbing în sistemul de fișiere, despre care vom discuta detaliat în Capitolul 7.

2.3.5.1 Copierea

Comanda folosită pentru a copia un fișier este **`cp`** [**`opțiuni`**] <**`sursă`**> <**`destinație`**>. Mai jos este prezentat un exemplu de copiere a unui fișier și unul de copiere a unui director. Pentru copierea unui director folosim opțiunea `-r` a comenzii **`cp`** pentru copiere recursivă (care trece prin toate subdirectoarele).

```
1  student@uso:~$ cp passwd passwd-copy
2  student@uso:~$ ls -l passwd passwd-copy
3  -rw-r--r-- 1 student student 2506 sep 30 11:28 passwd
4  -rw-r--r-- 1 student student 2506 sep 30 16:48 passwd-copy
5  student@uso:~$ cmp passwd passwd-copy
6  student@uso:~$ cp uso.git/labs/01-fs/wiki/basics.wiki .
7  student@uso:~$ cmp basics.wiki uso.git/labs/01-fs/wiki/basics.wiki
8  student@uso:~$ cp -r uso.git/labs/ .
9  student@uso:~$ diff -r uso.git/labs/ labs/
10 student@uso:~$
```

În exemplu de mai sus, în primă fază facem o copie, în același director a fișierul `passwd` în fișierul `passwd-copy`. Apoi folosim comanda **`ls -l`** pentru a vedea că

cele două fișiere au aceeași dimensiune (2506). Comanda **cmp** compară două fișiere și, dacă sunt identice, nu afișează nimic. Apoi copiem un fișier dintr-o cale relativă (uso.git/labs/01-fs/wiki/basics.wiki) în directorul curent, indicat de construcția . (punct). Folosim comanda **cmp** pentru a valida că cele două fișiere sunt identice. Apoi copiem un director dintr-o cale relativă (uso.git/labs/) în directorul curent. Pentru a compara două directoare și pentru a demonstra că sunt identice, folosim comanda **diff** cu opțiunea -r, pentru parcurgere recursivă a argumentelor de comparat.

2.3.5.2 Mutarea/Redenumirea

Comanda folosită pentru a muta/redenumi un fișier este **mv** [**opțiuni**] <**sursă**> <**destinație**>. Mutarea se va face implicit recursiv și va păstra toate atributele fișierelor. Când se execută comanda **mv**, se schimbă doar părintele fișierului (sau directorului) pe care îl mutăm. Spre deosebire de comanda **cp**, comanda **mv** nu primește opțiunea -r pentru mutarea unui director; o intrare este mutată indiferent dacă este director sau fișier.

Mai jos este un exemplu de folosire a comenzii **mv**:

```
1  student@uso:~$ mv basics.wiki 01-fs-basics.wiki
2  student@uso:~$ ls basics.wiki 01-fs-basics.wiki
3  ls: cannot access 'basics.wiki': No such file or directory
4  01-fs-basics.wiki
5  student@uso:~$ mv passwd-copy Downloads/
6  student@uso:~$ ls passwd-copy
7  ls: cannot access 'passwd-copy': No such file or directory
8  student@uso:~$ ls Downloads/passwd-copy
9  Downloads/passwd-copy
10 student@uso:~$ mv labs/ labs-old
11 student@uso:~$ ls labs
12 ls: cannot access 'labs': No such file or directory
13 student@uso:~$ ls labs-old/
14 00-intro  01-fs  02-process  03-user  04-appdev  05-cli  06-hw-boot  07-
   storage  08-net  09-vm  10-sec  11-ctf
```

În exemplul de mai sus am redenumit fișierul basics.wiki în 02-fs-basics.wiki. Am mutat fișierul passwd-copy în subdirectorul Downloads/, dat prin cale relatativă. Și am redenumit directorul labs/ în labs-old/. La fiecare operație de mutare/redenumire am verificat că vechea intrare nu mai este disponibilă, dar este disponibilă noua intrare.

2.3.5.3 Ștergerea fișierelor/directoarelor

Ștergerea unei intrări înseamnă eliminarea acelei intrări și a conținutului său din sistemul de fișiere.

Comanda cea mai folosită pentru ștergerea fișierelor și directoarelor este **rm** [**opțiuni**] <**cale**>.

Opțiunile cele mai des folosite pentru această comandă sunt:

- -r/-R: se utilizează atunci când se dorește ștergerea recursivă a unui director

- `-f`: se utilizează pentru a șterge forțat fișierele

Este important să fim atenți când folosim această comandă și opțiunile ei, pentru a nu șterge informații utile din greșeală și apoi să nu le mai putem recupera. Una dintre cele mai întâlnite erori este **`rm -rf /`**, care șterge recursiv toate fișierele începând cu rădăcina.

> **Observație:** Dacă dorim să ștergem un director gol, putem să folosim comanda **`rmdir`** `<`**`director_gol`**`>`

Mai jos este un exemplu de folosire a comenzilor **`rm`** și **`rmdir`**.

```
1  student@uso:~$ ls
2  02-fs-basics.wiki  Desktop  Documents  Downloads  Music  Pictures  Public
     Templates  Videos  examples.desktop  labs-old  passwd  readme_link  uso
   .git  vm-actions-log.txt
3  student@uso:~$ rm 02-fs-basics.wiki
4  student@uso:~$ rm passwd
5  student@uso:~$ rm readme_link
6  student@uso:~$ rm -fr labs-old
7  student@uso:~$ rmdir Downloads/
8  rmdir: failed to remove 'Downloads/': Directory not empty
9  student@uso:~$ rm Downloads/*
10 student@uso:~$ rmdir Downloads/
11 student@uso:~$ ls
12 Desktop  Documents  Music  Pictures  Public  Templates  Videos  examples.
   desktop  uso.git  vm-actions-log.txt
```

În exemplul de mai sus am șters, folosind comanda **`rm`** fișierele `basics-wiki` și `passwd` și legătura simbolică `readme_link`. Apoi am șters recursiv (folosind opțiunea `-fr` (de la *force and recursive*) directorul `labs-old/`. Comanda **`rmdir`** poate șterge un director doar dacă acesta este gol; pentru aceasta am șters toate fișierele din directorul `Downloads/` folosind construcția `Downloads/*`; apoi am folosit comanda **`rmdir`** pentru a șterge directorul. Am folosit comanda **`ls`** la început și la sfârșit pentru a demonstra ștergerea fișierelor.

2.3.6 Căutarea fișierelor

Structura ierarhică a sistemului de fișiere permite găsirea rapidă a unui fișier parcurgând pe rând subdirectoarele care-l conțin. Chiar și așa sunt situații în care nu se știe precis unde este localizat un fișier sa un director și este utilă căutarea acestuia. Pentru a căuta un fișier după numele său există două tipuri de căutări:

1. **căutare indexată** în care fișierele sunt inspectate și informațiile despre localizarea lor sunt reținute într-o bază de date cu fișierele indexată;
2. **căutare exhaustivă** în cadrul unei zone din ierarhia sistemului de fișiere, trecând prin toate fișierele

Pentru căutare indexată folosim, în Linux, comanda **`locate`** (și comanda **`updatedb`**). Pentru căutare exhaustivă folosim, în Linux, comanda **`find`**. Le prezentăm în continuare.

2.3.6.1 Comanda locate

Comanda **`locate`** caută în toate fișierele indexate într-o bază de date locală. Avantajul său este căutarea rapidă în sistemul de fișiere. Are două dezavantaje:

1. Baza de date trebuie actualizată periodic pentru a indexa modificările din sistemul de fișiere. Actualizarea se face folosind comanda **`updatedb`**.
2. Căutarea se face strict după nume, nu și după alte atribute ale fișierelor.

Rezultatul rulării comenzii este o listă cu toate fișierele al căror nume conține șirul de caractere trimis ca argument.

În exemplul următor se caută toate fișierele al căror nume conține șirul `pwd`:

```
student@uso:~$ locate pwd
/bin/pwd
/etc/.pwd.lock
/lib/modules/4.15.0-29-generic/kernel/drivers/watchdog/hpwdt.ko
/lib/modules/4.15.0-32-generic/kernel/drivers/watchdog/hpwdt.ko
/lib/modules/4.15.0-34-generic/kernel/drivers/watchdog/hpwdt.ko
/sbin/pam_extrausers_chkpwd
/sbin/unix_chkpwd
/snap/core/4917/bin/pwd
[...]
```

Unele distribuții Linux folosesc comanda **`slocate`** în locul comenzii **`locate`**, care afișează doar fișierele din directoarele în care utilizatorul curent are drepturi de acces.

2.3.6.2 Comanda find

Pentru căutarea în ierarhia de directoare al fișierelor se folosește comanda **`find`**, comandă ce parcurge exhaustiv ierarhia de directoare. Comanda permite căutarea folosind diferite criterii, precum numele fișierului, utilizator, grup, tip, permisiuni, dimensiune, dată și altele. Exemple de opțiuni ale comenzii **`find`** pentru criterii de căutare:

- `-name` - căutare după numele fișierelor
- `-type` - căutare după tipul intrării (fișier, director, legătură simbolică)
- `-size` - căutare după dimensiunea fișierelor

Pentru a căuta în ierarhia **`/usr/`** toate legăturile simbolice, folosim comanda **`find`** astfel:

```
student@uso:~$ find /usr/ -type l
/usr/lib/pm-utils/sleep.d/95hdparm-apm
/usr/lib/libgjs.so.0
/usr/lib/bfd-plugins/liblto_plugin.so
/usr/lib/sudo/libsudo_util.so
/usr/lib/sudo/libsudo_util.so.0
/usr/lib/gold-ld/ld
/usr/lib/systemd/user/dbus-org.bluez.obex.service
/usr/lib/systemd/user/graphical-session.target.wants/update-notifier-
release.path
```

```
10 /usr/lib/systemd/user/graphical-session.target.wants/update-notifier-
   crash.path
11 /usr/lib/systemd/user/graphical-session.target.wants/unicast-local-avahi.
   path
12 [...]
```

Pentru a căuta în ierarhia **/usr/** toate fișierele care au numele `stat` se folosește comanda **find** astfel:

```
1 student@uso:~$ find /usr -name stat
2 /usr/bin/stat
3 /usr/src/linux-headers-4.15.0-29-generic/include/config/f2fs/stat
4 /usr/src/linux-headers-4.15.0-32-generic/include/config/f2fs/stat
5 /usr/src/linux-headers-4.15.0-34-generic/include/config/f2fs/stat
```

Pentru a obține fișierele ce au o mărime mai mare de 500KB, folosim comanda:

```
1 student@uso:~$ find /usr/ -size +500k
2 /usr/lib/gnome-shell/gnome-shell-portal-helper
3 /usr/lib/gnome-shell/libgnome-shell.so
4 /usr/lib/p7zip/7zr
5 /usr/lib/xorg/modules/drivers/intel_drv.so
6 /usr/lib/xorg/Xorg
7 /usr/lib/debug/lib/x86_64-linux-gnu/libnsl-2.27.so
8 /usr/lib/debug/lib/x86_64-linux-gnu/libmvec-2.27.so
9 /usr/lib/debug/lib/x86_64-linux-gnu/ld-2.27.so
10 /usr/lib/debug/lib/x86_64-linux-gnu/libm-2.27.so
11 /usr/lib/debug/lib/x86_64-linux-gnu/libc-2.27.so
12 [...]
```

2.3.6.3 Căutarea comenzilor

Uneori dorim să identificăm o comandă, sau executabilul aferent unei comenzi. Pentru aceasta folosim comenzile **whereis**, **which** și **type**.

Comanda **whereis** este utilizată pentru căutarea locurilor corespunzătoare unei comenzi în sistemul de fișiere De exemplu, pentru a localiza comanda **ls**, folosim următoarea comandă:

```
1 student@uso:~$ whereis ls
2 ls: /bin/ls /usr/share/man/man1/ls.1posix.gz /usr/share/man/man1/ls.1.gz
```

Comanda afișează calea atât către executabil și către pagina de manual a comenzii.

Dacă dorim doar obținerea căii către executabilul aferent comenzii, folosim comanda **which**. De exemplu, comanda următoare afișează calea către executabilul aferent comenzii **chmod**.

```
1 student@uso:~$ which chmod
2 /bin/chmod
```

Utilizarea comenzii **type** duce la determinarea modului de interpretare a altei comenzi, de exemplu comandă integrată în shell (numită și internă), comandă externă sau alias. Mai multe despre comenzi în shell, interne și externe, vor fi prezentate în Capitolul 7.

Un exemplu de comandă integrată in shell este comanda **cd** în vreme ce o comanda externă, care are un executabil asociat, este comanda **cat**, așa cum putem vedea în exemplul de mai jos:

```
student@uso:~$ type cd
cd is a shell builtin
student@uso:~$ type cat
cat is /bin/cat
```

În cazul unui alias se va afișa comanda echivalentă:

```
student@uso:~$ type ls
ls is aliased to 'ls --color=auto'
```

2.3.7 Arhivarea și dezarhivarea fișierelor

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.

În cazul compresiei se creează un dicționar care conține secvențe de octeți care se repetă mai des și codificarea lor pe mai puțini octeți. Pe lângă dicționar, în fișierul rezultat este conținut și cuprinsul fișierelor. Fiind folosite referințele către dicționar a secvențelor lungi, se realizează astfel o economie de spațiu. Cu cât fișierele conțin mai multe date repetitive, cu atât dimensiunile fișierului comprimat rezultat scad.

Utilitare precum **`zip`** realizează atât arhivare cât și compresie; utilitarul **`tar`** realizează doar arhivare, iar utilitarul **`gzip`** realizează doar compresie; de obicei utilitarele **`tar`** și **`gzip`** sunt folosite la comun pentru a realiza și arhivarea și compresia.

Numele utilitarului **`tar`** este acronimul *tape archiver*, provenind de la faptul că, la origine, rezultatul era transferat pe benzi magnetice. Fișierele `.tar` au în componență fișierele inițiale necomprimate precum și informațiile legate de modul lor de extragere (spre exemplu: de unde până unde se găsește un fișier în cadrul arhivei). Din această cauză fișierele `.tar` au o dimensiune mai mare decât suma dimensiunilor fișierelor care îl alcătuiesc.

În continuare sunt prezentate exemple de folosire ale comenzii **`tar`**:

```
student@uso:~$ tar cvf 01-fs.tar uso.git/labs/01-fs/
uso.git/labs/01-fs/
uso.git/labs/01-fs/wiki/
uso.git/labs/01-fs/wiki/basics.wiki
uso.git/labs/01-fs/wiki/concepts.wiki
uso.git/labs/01-fs/wiki/demo.wiki
[...]
student@uso:~$ tar tf 01-fs.tar
uso.git/labs/01-fs/
uso.git/labs/01-fs/wiki/
uso.git/labs/01-fs/wiki/basics.wiki
uso.git/labs/01-fs/wiki/concepts.wiki
uso.git/labs/01-fs/wiki/demo.wiki
[...]
student@uso:~$ cd /tmp/
student@uso:/tmp$ tar xf ~/01-fs.tar
student@uso:/tmp$ ls uso.git/labs/01-fs/
support  wiki
```

Mai sus am creat o arhivă `.tar`, am listat conținutul ei și apoi am dezarhivat-o în directorul `/tmp/`.

Opțiunile comenzii **`tar`**, folosite mai sus, sunt:

- `c` (pentru *create*): pentru a crea arhiva
- `x` (pentru *extract*): dezarhivează
- `t` (pentru *list*): listează conținut
- `v` (pentru *verbose*): arată ce se întâmplă
- `f nume_arhiva.tar` (pentru *file*): numele arhivei

> **Observație:** `f` și `nume_arhiva.tar` reprezintă un singur parametru – astfel, când folosim `f` pentru a indica un fișier, acesta trebuie să se afle ultimul în lista de parametri

Cele mai folosite utilitare în Linux care realizează comprimarea unui fișier sunt:

- **`gzip`**: are o viteză mai ridicată de compresie, dar o rată de mai scăzută (fișiere rezultat mai mari)
- **`bzip2`**: are viteză mai scăzută de compresie, dar o rată de compresie mai ridicată

Comanda **`tar`** poate să utilizeze oricare dintre programele de compresie de mai sus și se folosește de parametrul `z` pentru gzip și `j` pentru bzip2 ca mai jos:

```
1 student@uso:~$ tar czf 01-fs.tar.gz uso.git/labs/01-fs/
2 student@uso:~$ tar cjf 01-fs.tar.bz2 uso.git/labs/01-fs/
3 student@uso:~$ file 01-fs.tar.*
4 01-fs.tar.bz2: bzip2 compressed data, block size = 900k
5 01-fs.tar.gz:  gzip compressed data, last modified: Sun Sep 30 17:02:19
  2018, from Unix
```

În exemplul de mai sus am folosit, respectiv, opțiunea `z` (în cadrul opțiunii `czf`) pentru a comprima folosind **`gzip`**, și apoi opțiunea `j` (în cadrul opțiunii `cjf`) pentru a comprima folosind **`bzip2`**. Apoi am folosit comanda **`file`** pentru a verifica tipul fișierelor și am confirmat tipul de compresie aferent celor două fișiere rezultate.

Există, de asemenea, posibilitatea folosirii și a altor opțiuni, în afară de cele de compresare și arhivare, cele mai utilizare fiind `--preserve`, care poate determina păstrarea drepturilor de acces la arhivare și la dezarhivare.

2.3.8 Backup

Backup-ul este utilizat pentru a păstra într-un loc separat o copie a datelor, copie ce poate fi folosită pentru a le recupera în cazul în care, din diferite motive, suportul original nu mai poate fi folosit. Adesea omitem sau amânăm realizarea unei versiuni de backup a datelor, putând ajunge la pierderea informațiilor în situații de defecțiune. De aceea, în ultimii ani sunt foarte populare soluțiile de backup automat în Cloud. Astfel, utilizatorii pot folosi soluții precum Google Drive, Dropbox, Microsoft OneDrive pentru a sincroniza automat conținutul directoarelor în care lucrează cu un spațiu de stocare

cloud. Dacă utilizatorul lucrează pe mai multe dispozitive (desktop, laptop, mobil etc.) poate fi configurată sincronizarea tuturor acestor dispozitive cu cloud-ul, facilitând astfel transferul documentelor.

Chiar dacă realizăm sincronizarea automată prin Cloud, este util să creăm și backup-uri periodice pe alte suporturi fizice. De exemplu, în cazul în care datele noastre sunt afectate de erori grave, datorită malware sau ștergerii accidentale, aceste erori se pot propaga și în Cloud, dar versiunile salvate anterior pe suporturi fizice distincte vor fi protejate.

Tabelul 2.5 prezintă câteva metode de backup și situațiile când sunt ele potrivite.

Tabelul 2.5: Metode pentru backup

Metoda	**Descriere**
tar + gzip/bzip2	Metodă foarte simplu de aplicat. Devine greu de folosit pentru dimensiuni mari de date. Permite compresia datelor.
dd	Metoda simplu de folosit și independentă de sistemul de fișiere. Permite păstrarea intactă a structurii sistemului de fișiere. Inflexibilă când vine vorba de recuperarea datelor. Utilă pentru cantități mari de date.
rsync	E asemănătoare comenzii cp dar la care s-a adăugat suport de sincronizare intre mai multe computere. Permite replicarea structurii de fișiere (inclusiv permisiuni) între 2 computere.
rdiff-backup	Este un wrapper peste rsync. Adaugă suport pentru backup-uri incrementale, adică: la un moment dat se realizează un backup complet pentru un director (asemănător rsync-ului); backup-urile incrementale salvează doar modificările ce s-au făcut de la ultimul backup pana în prezent, indiferent de tipul backup-ului; în acest fel se poate reveni la orice stare anterioara, în măsura în care s-a realizat cel puțin un backup incremental la acea stare.

Informații detaliate despre backup-uri periodice folosind **`rsync`** vor fi prezentate în Secțiunea 10.5.

2.4 Redirectarea intrării sau ieșirii

Așa cum veți întâlni în Secțiunea 4.4.1, o aplicație care rulează folosește niște intrări speciale (standard); în folosirea acestor intrări și a altor fișiere folosește descriptori de fișiere.

Există 3 **fișiere speciale** utilizate de programele în execuție pentru a interacționa cu utilizatorii:

- intrare standard (*standard input*, *stdin*): acesta reprezintă locul de unde se citesc datele de intrare de către program (de obicei tastatura)

- ieșire standard (*standard output*, *stdout*): acesta este fișierul în care se scriu datele de ieșire (de obicei este consola curentă)
- ieșire de eroare standard (*standard error*, *stderr*): în acest fișier se scriu mesajele de eroare de către program (de obicei tot în consola curentă)

Un descriptor de fișier reprezintă un indice asociat unui fișier deschis de o aplicație. Cele 3 fișiere speciale de mai sus au următorii descriptori de fișier:

- `stdin` are descriptorul cu indexul 0
- `stdout` are descriptorul cu indexul 1
- `stderr` are descriptorul cu indexul 2

Restul fișierelor deschise de aplicații au un descriptor de fișier mai mare sau egal cu 3. În C/C++ există 3 variabile de tipul `FILE *` cu numele `stdin`, `stdout`, `stderr`. Ele au același rol ca cele descrise mai sus și sunt folosite ca orice alte variabilă de tip `FILE *`.

În unele situații, utilizatorul poate dori să modifice intrarea sau ieșirea pentru o aplicație. De exemplu, utilizatorul își poate dori ca în loc să obțină datele de intrare de la tastatură pentru un program, să le obțină dintr-un fișier. Aceste operații se pot realiza doar la nivelul descriptorilor. În shell sunt permise comenzi cu o sintaxă specială asupra descriptorilor standard modificați; pot fi întâlnite cazurile de redirectare din Tabelul 2.6.

Tabelul 2.6: Metode de redirectare

Sursă	**Destinație**	**Exemplu comandă**
intrare (*stdin*)	Fișier	**`./program < fișier_intrare`**
ieșire (*stdout*)	Fișier	**`./program > fișier_ieșire`**
eroare (*stderr*)	Fișier	**`./program 2> fișier_erori`**
eroare (*stderr*)	iesire (stdout)	**`./program 2>&1`**
eroare & ieșire	Fișier	**`./program > fișier_ieșire_și_erori 2>&1`**

Pentru a redirecta ieșirea (către exteriorul programului) se folosește caracterul `>` (semnul *mai mare*), în timp ce pentru a redirecta intrarea se folosește caracterul `<` (semnul *mai mic*) (către program), iar pentru redirectarea ieșirii de erori se folosește descriptorul `2` de fișier urmat de caracterul `>`.

Sintaxa `&1` se folosește atunci când se dorește redirectarea către ieșirea standard (*stdout*), astfel încât construcția `2>&1` redirectează ieșirea de eroare standard către ieșirea standard. Astfel, dacă dorim redirectarea ieșirii de erori și a ieșirii standard către un fișier trebuie să redirectăm mai întâi ieșirea de erori către stdout (`2>&1`) și apoi să redirectăm ieșirea standard într-un fișier (`fișier_erori_și_ieșiri`).

Exemplu de redirectare a intrării:

```
1 student@uso:~$ mail gabriel < message.txt
```

În comanda anterioară intrarea pentru comanda **`mail`** este redirectată din conținutul fișierului `message.txt`. Utilizatorului `gabriel` i se va trimite un email cu informațiile din fișier.

Comanda următoare va scrie în fișierul listare toate fișierele și directoarele ce se află în directorul curent.

```
student@uso:~$ ls > listare
student@uso:~$ cat listare
01-fs.tar
01-fs.tar.bz2
01-fs.tar.gz
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
examples.desktop
games
listare
uso.git
vm-actions-log.txt
```

Pentru a adăuga la sfârșitul fișierului rezultatul unei comenzi folosim operatorul $>>$:

```
student@uso:~$ date >> listare
student@uso:~$ cat listare
[...]
vm-actions-log.txt
duminica 30 septembrie 2018, 20:17:50 +0300
```

Comanda de mai jos încearcă să copieze un fișier fără a spune unde dorește să-l copieze, comandă ce va genera o eroare care va fi redirectată în fișierul `errors.out`.

```
student@uso:~$ cp listare 2> errors.out
student@uso:~$ cat errors.out
cp: missing destination file operand after 'listare'
Try 'cp --help' for more information.
```

Comanda următoare va scrie atât erorile cât și rezultatele comenzii **`strace ls`** în fișierul `strace-all.out`:

```
student@uso:~$ strace ls > strace-all.out 2>&1
student@uso:~$ cat strace-all.out
execve("/bin/ls", ["ls"], 0x7ffd95189350 /* 36 vars */) = 0
brk(NULL)                               = 0x5578b9d96000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=91329, ...}) = 0
[...]
```

Tabelul 2.7 prezintă câteva redirectări folosind fișiere speciale.

Detalii despre internele redirectării și ce se întâmplă la nivelul unei aplicații găsiți în Secțiunea 4.4.2.

Tabelul 2.7: 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 ca `/dev/null`, adică un fișier gol, indentic ca mai sus

2.5 Tipuri de sisteme de fișiere

În timp au apărut mai multe tipuri de sisteme de fișiere, majoritatea dezvoltându-se în mediul open-source. Utilizatorii au opțiunea de a alege sistemul de fișiere preferat cu care doresc să lucreze, mai ales în cazul în care folosesc partiții multiple pentru mai multe sisteme de operare.

Pentru mai multe informații despre crearea, montarea și repararea unui sistem de fișiere, precum și lucrul cu partiții, citiți Capitolul 10. În Tabelul 2.8 de mai jos se găsesc unele dintre cele mai importante sisteme de fișiere utilizate în prezent, alături de sistemele de operare în care operează.

Tabelul 2.8: Sisteme de operare și sisteme de fișiere

SO Sistem fișiere	Windows	Linux	Mac OS
FAT32	Nativ	Nativ	Nativ
NTFS	Nativ (dupa WinNT)	prin ntfs-3g	prin ntfs-3g
Ext2/Ext3/Ext4	Driver third-party	Nativ	-
HFS+	-	Nativ	Nativ
APFS	-	-	Nativ
ISO9660	Nativ	Nativ	Nativ
UDF	Nativ	Nativ	Nativ

Termenul „nativ” semnifică faptul că suportul este oferit de driverele ce însoțesc sistemul de operare.

Sistemele de fișiere pot fi clasificate după locul în care datele sunt stocate. Tabelul 2.9 prezintă succint această clasificare.

Sistemele de fișiere virtuale (precum `procfs`) sunt sisteme care nu au suport fizic pe disc. Accesul prin comenzi la aceste sisteme de fișiere duce de obicei la date care

Tabelul 2.9: Clasificarea sistemelor de fișiere după suportul datelor

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 (informații despre sistem) sau de o altă componentă software (alte surse)
sisteme de fișiere pentru rețea	NFS, SMB	utilizate pentru accesul la fișiere aflate în rețea

se găsesc în memorie. Despre sistemul de fișiere `procfs` vor fi prezentate detalii în Secțiunea 4.8.

2.5.1 Integritatea datelor

Cărțile din bibliotecă se îngălbenesc, caietele se șifonează și scrisul își pierde conturul. Fișierele digitale se pot și ele degrada, datorită unor erori. Sistemele de fișiere au un rol vital în asigurarea integrității datelor (*data integrity*), adică în prevenirea, detectarea și remedierea coruperii datelor. Coruperea datelor (*data corruption*) se referă la degradarea neintenționată a datelor datorită erorilor umane, erorilor de transmisie, defecțiunilor mediului fizic de stocare sau a diferitelor deficiențe în procesare. Securizarea datelor (*data security*) se referă la prevenirea atacurilor intenționate asupra datelor și de asemenea necesită sprijinul sistemului de fișiere.

Un exemplu de metodă prin care sistemele de fișiere pot asigura integritatea datelor se referă la calcularea unei sume de control (*checksum*). Pentru un anumit set de date, o sumă de control este un număr care descrie pe scurt acel set, diferențiindu-l suficient de mult de un set foarte similar dar un pic diferit. De exemplu, o sumă de control poate consta efectiv în însumarea valorilor numerice asociate cu caracterele respective într-un cod, precum valorile lor numerice din codul ASCII. Dacă se modifică o literă din greșeală, suma va diferi și eroarea va fi detectată.

Sistemul de fișiere trebuie să fie capabil să păstreze integritatea datelor și în situația unui eșec (*failure*). În cele mai multe cazuri această funcționalitate se implementează prin **jurnalizare**, care permite și aducerea sistemului la o stare anterioară eșecului. Prin această activitate se creează un jurnal unde sunt păstrate toate modificările efectuate asupra unui sistem de fișiere, fiecare dintre aceste modificări fiind mai întâi scrisă în jurnal și apoi realizată (modificările putând apărea în mai multe locuri din sistemul de fișiere).

Operațiile din jurnal sunt executate la diferite intervale de timp de către driver-ul sistemului pentru actualizarea stării discului. După ce toate modificările asociate unei operații sunt executate, aceasta se șterge din jurnal, astfel că, la apariția unui eșec (cum ar fi întreruperile de curent, defecțiunile fizice ale dispozitivului etc.), vor putea fi executate operațiile care au fost începute, dar s-au întrerupt, deoarece ele sunt încă

prezente în jurnal, urmând să fie reexecutate în întregime.

2.5.2 Alegerea unui sistem de fișiere

Atunci când alegem un sistem de fișiere, criteriile cele mai căutate sunt:

- disponibilitatea: dacă sistemul respectiv poate fi folosit în mai multe sisteme de operare, sau doar în unul singur;
- gradul de siguranță: dacă asigură jurnalizare sau nu – majoritatea sistemelor din prezent folosesc jurnalizarea, alte măsuri luate pentru a permite integritatea datelor;
- restricții speciale date de modul în care datele din sistemul de fișiere sunt organizate;de exemplu: FAT32 folosește doar 32 biți pentru stocarea dimensiunii unui fișier, deci dimensiunea maximă a unui fișier este 4GB (mai puțin decât o imagine de DVD, fapt ce îl face nepractic pentru o parte din aplicațiile multimedia)
- optimizări de performanță, ce sunt realizate în funcție ori de tipul suportului de stocare, ori de dimensiunea fișierelor etc.

Tabelul 2.10 realizează o analiză sumară a caracteristicilor sistemelor de fișiere. Unitatea TiB reprezintă un tebibyte, adică 2^{40} octeți, iar un EiB reprezintă un exbibyte, adică 2^{60} octeți.

2.6 Anexă: Comenzi pentru lucrul cu fișiere în Windows

Tabelul 2.11 prezintă comenzile echivalente în Windows pentru operațiile de lucru cu sistemul de fișiere. Acestea pot fi folosite în prompt-urile Windows precum PowerShell sau Command Prompt.

Tabelul 2.10: Caracteristici ale sistemelor de fișiere mai cunoscute

Tip sistem de fișiere	**Sisteme de operare**	**Dimensiune maximă fișier**	**Jurnalizare**	**Observații**
FAT32	Windows / Linux / Mac OS	4 GB	nu	Cel mai folosit sistem de fișiere - întâlnit în mod special pe USB stick-uri, fără drepturi de acces; Windows-ul limitează la creare dimensiunea unei partiții la 32GB, dar poate citi partiții mai mari realizate și formatate cu aplicații third-party folosit pentru a asigura compatibilitatea cu dispozitive sau sisteme mai vechi
NTFS	Windows / Linux / Mac OS	16TiB	da	Singurul sistem de fișiere pentru Windows recomandat de Microsoft. Singurul sistem de fișiere pentru Windows cu securitate.
Ext2/Ext3	Linux / Windows	16GiB - 64TiB	ext2-nu ext3-da	Sistemul de fișiere considerat cel mai stabil datorită unei istorii de dezvoltare foarte lungi; ext3 e compatibil cu versiunea anterioară ext2, aduce jurnalizare față de ext2
Ext4	Linux / Windows / MacOS	16TiB	da	Succesorul lui Ext3, îmbunătățind performanța, stabilitatea și capacitatea de stocare. Este adecvat și pentru stocarea datelor critice, datorită preciziei ridicate ale marcajelor temporale.
HFS+	Mac OS / Linux	16 EiB	da*	* în Linux, HFS+ este suportat fără jurnalizare
APFS	MacOS, iOS	8 EiB	da	Este optimizat pentru dispozitivele de stocare tip flash și solid drive, și pune accentul pe criptare.
ISO9660	Win / Linux / Mac OS	În funcție de implementare	nu	Sistem de fișiere utilizat în principal pe CD-uri, organizare internă concepută pentru ca datele să fie citibile ușor
UDF	Win / Linux / Mac OS	16 EiB	da	Sistem de fișiere utilizat în principal pe mediile optice, cu suport atât pentru scriere cât și pentru citire

Tabelul 2.11: Echivalențe comenzi Linux și Windows

Comanda Linux	Comanda Windows	Descriere
`comanda --help`	`comanda /?`	afișează informații despre comandă
`cd`	`cd`	schimbă directorul curent
`pwd`	`chdir`	afișează directorul curent
`clear`	`cls`	șterge ecranul consolei curente
`cp`	`copy`	copiază un fișier
`rm`	`del`	șterge un fișier
`ls`	`dir`	afișează conținutul directorului curent
`vim`	`edit`	editează un fișier text
`exit`	`exit`	închide shell-ul curent
`diff`	`fc`	compară două fișiere și afișează diferențele între ele
`find`	`find`	caută fișiere
`mkfs (mke2fs)`	`format`	formatează un disc
`free`	`mem`	afișează informații despre memoria liberă și cea ocupată
`mkdir`	`mkdir`	creează un nou director
`mv`	`move`	mută un fișier
`mv`	`ren`	redenumește un fișier
`date`	`time`	afișează ora sistemului
`diff`	`fc`	afișează diferențele dintre două fișiere