

# Laboratorul 1

## Linia de comandă, Sistemul de fișiere

### 1 Ierarhie

În sistemele de operare de tip UNIX fișierele și directoarele sunt organizate într-o structură arborescentă. Rădăcina este notată cu `/` și se mai numește și *root*. În rădăcină se găsesc mai multe directoare și fișiere. La rândul lor, aceste directoare pot conține alte fișiere și directoare. Directoarele dintr-un director se mai numesc și *subdirectoare*, subliniind relația ierarhică.

De exemplu, mesaje de sistem se găsesc în fișierul `/var/log/messages`. Acesta se află în directorul `log` care se află la rândul său în directorul `var`, care se află în rădăcină. Alt exemplu este directorul `itbi` care conține două directoare, care la rândul lor au câte un subdirector în care se găsesc diferite fișiere.

```
$ tree itbi
itbi
|-- curs
|   '-- 1
|       |-- itbi-curs-1.pdf
|       '-- itbi-curs-1.tex
|-- lab
|   '-- 1
|       |-- itbi-lab-1.aux
|       |-- itbi-lab-1.fdb_latexmk
|       |-- itbi-lab-1.fls
|       |-- itbi-lab-1.log
|       |-- itbi-lab-1.pdf
|       |-- itbi-lab-1.synctex.gz
|       '-- itbi-lab-1.tex
```

### 2 Linia de comandă

Majoritatea comenzilor vor fi executate în cadrul unui terminal (ex. `xterm(1)`, GNOME Terminal etc.). Deși dificil și aparent mai complicat la început, folosirea terminalului oferă multe avantaje precum flexibilitate, automatizarea sarcinilor și control la distanță al altor mașini.

Un terminal tipic are la bază un program de tip shell (ex. `bash(1)`, `ksh`, `zsh`) care gestionează și execută comenzi secvențial sau în paralel. Promptul unui terminal indică

faptul că se așteaptă o comandă de la utilizator și este în cea mai simplă formă constituit din simbolul `$` sau `%` pentru utilizatorii comuni și `#` pentru administrator (denumit *root* în UNIX). În Linux se practică un prompt mai elaborat care poate conține numele utilizatorului, numele mașinii, și/sau directorul curent. De exemplu, promptul `root@lab uso#` indică faptul că utilizatorul `root` este logat pe terminalul de pe mașina `lab` și se află în directorul `uso`.

Tot ce este scris în dreapta promptului reprezintă comanda utilizatorului către mașină.

```
$ echo "Hello, World!"
Hello, World!
$
```

În exemplul de mai sus a fost executată comanda `echo` cu parametrul `"Hello, World!"`. Rezultatul comenzii, dacă există, este afișat fără a fi prefixat cu prompt. Încheierea comenzii este semnalată prin reapariția promptului.

De fiecare dată când vedeți o comandă necunoscută citiți manualul pentru a afla ce face:

```
$ man echo
```

Din motive istorice, manualul sistemului de operare (ce include manualele comenzilor) este împărțit în secțiuni. Astfel pot exista mai multe intrări cu același nume dar în secțiuni diferite. Vezi cunoscuta funcție `printf`.

```
$ man printf
$ man 3 printf
```

Prima instrucțiune s-ar putea să vă surprindă afișând manualul comenzii `printf` nu a funcției C `printf`. Comenzile de terminal se află de regulă în secțiunea 1, pe când funcțiile se află în secțiunea 3. Pentru a accesa manualul funcției trebuie să specificăm un argument în plus comenzii `man(1)` care specifică secțiunea explicit. Din această cauză când ne referim la o comandă sau o funcție punem la sfârșit și secțiunea din manual în care este documentată: `printf(1)` versus `printf(3)`. Este bine de știut că există manual și pentru comanda de citit manuale:

```
$ man man
```

## 3 Navigare

În general, fiecare utilizator are un spațiu de lucru propriu în care își poate desfășura activitatea. Acest spațiu este găzduit într-un director, de regulă `/home/username`, care este memorat în variabila `$HOME`.

```
$ echo $HOME
/home/horatiu
```

Acest tip de variabilă se mai numește și variabilă de mediu. Ele sunt definite dinamic de sistem sau utilizator pentru a fi folosite de programe la execuție. Variabilele sunt în general scrise cu litere mari și precedate de simbolul `$`.

Când este pornit un terminal, acesta de regulă vă plasează în directorul `$HOME`. Folosiți comanda `pwd(1)` pentru a verifica în orice moment unde vă aflați și comanda `ls(1)` pentru a lista conținutul directorului curent.

```
$ pwd
/home/horatiu/Documents/itbi/lab1
$ ls
itbi-lab-1.aux  itbi-lab-1.fdb_latexmk  itbi-lab-1.flx
itbi-lab-1.log  itbi-lab-1.pdf           itbi-lab-1.tex
```

Toate comenzile se execută relativ la directorul curent: `ls(1)` verifică implicit directorul curent și listează conținutul său.

Dacă doriți să schimbați directorul curent folosiți comanda `cd(1)`. Aceasta primește ca parametru viitorul director curent. El poate fi dat relativ la directorul curent sau în formă absolută pornind de la rădăcină. În aproape toate sistemele de operare `.` simbolizează directorul curent și `..` directorul părinte. Deci dacă vrem din exemplul anterior să ajungem acasă putem folosi oricare dintre următoarele comenzi:

```
$ cd ../../../../
$ cd /home/horatiu
$ cd $HOME
$ cd
```

Implicit `cd(1)` fără argumente schimbă directorul în directorul `$HOME`.

Pentru a afla unde se află executabilul aferent unei comenzi folosiți comanda `which(1)`:

```
$ which ls
/usr/bin/ls
```

O altă variabilă importantă este cea în care sunt memorate căile din sistemul de fișiere în care se găsesc executabilele.

```
$ echo $PATH
/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:/usr/local/bin:/usr/local/sbin
```

În general, comanda pentru a rula un executabil este dată de calea (fie absolută, fie relativă la directorul curent) către acesta.

Calea absolută:

```
$ /path/to/executable
```

Calea relativă:

```
$ ./path/to/executable
```

Observați că `ls(1)` se află într-unul din directoarele conținute în `$PATH`, motiv pentru care nu este nevoie să specificăm întreaga cale. Pentru a adăuga directoarele la `$PATH` se poate folosi comanda

```
$ export $PATH=$PATH:/path/to/directory
```

pentru adăugare la final, sau

```
$ export $PATH=/path/to/directory:$PATH
```

pentru adăugare la început.

Spre exemplu, după

```
$ export $PATH=$PATH:/home/horatiu/itbi/bin
```

toate executabilele din /home/horatiu/itbi/bin vor putea fi apelate drept comenzi cu numele lor.

## 4 Editarea textului în linia de comandă

Pentru editarea fișierelor din linia de comandă sunt disponibile diverse editoare fără interfață grafică, precum `nano(1)` sau `vi(1)`.

```
$ nano main.cpp
```

Un avantaj al acestora este că pot fi utilizate inclusiv pe sisteme care nu pun la dispoziție o interfață grafică. Desigur, multe distribuții vin cu editoare grafice instalate, precum `gedit` sau `kate`, care pot fi apelate într-o manieră similară.

```
$ gedit main.cpp
```

## 5 Citire și scriere

Pentru a crea fișiere noi text se poate folosi `echo(1)`:

```
$ echo "lorem ipsum" > foo
```

unde operatorul `>` redirecționează ieșirea comenzii către fișierul `foo`. Dacă `foo` există, va fi suprascris. Pentru a adăuga la sfârșitul unui fișier existent folosiți `»`. Fișierele text scurte pot fi rapid afișate în terminal cu ajutorul comenzii `cat(1)`.

```
$ cat foo
lorem ipsum
```

Deși se poate aplica aceeași comandă asupra fișierelor binare, precum executabilele, nu este recomandat deoarece anumite “caractere” rezultate pot fi interpretate de shell drept caractere de control care vor da peste cap funcționarea normală a terminalului. De aceea se folosește utilitarul `hexdump(1)`:

```
$ hexdump -C /bin/ls
```

unde opțiunea `-C` indică modul canonic de afișare a binarelor: în stânga octeții în format hexadecimal și în dreapta octeții în format ASCII. Rezultatul este destul de lung și depășește lungimea terminalului. Pentru a parcurge toată informația se recomandă folosirea unui *pager* precum `less(1)`:

```
$ hexdump -C /bin/ls | less
```

unde operatorul `|` se numește *pipe*. Un *pipe* transformă ieșirea programului din stânga în intrarea celui din dreapta. Pentru a ieși din `less(1)` apăsați tasta `q`. Pentru a căuta un text folosiți comanda `/`. De exemplu `/print` va căuta șirul de caractere `print`. Evident, `less(1)` poate fi folosit direct pentru a inspecta fișiere și este util mai ales pentru fișiere text mari:

```
$ less /etc/passwd
```

## 6 Manipulare

Directoarele sunt create cu comanda `mkdir(1)`:

```
$ pwd
/home/horatiu/Documents/itbi
$ mkdir tmp
$ cd tmp
```

Pentru a crea fişiere noi lipsite de conţinut folosiţi comanda `touch(1)`.

```
$ touch foo
$ ls
foo
```

Operarea de copiere se face cu comanda `cp(1)`:

```
$ cp foo bar
$ ls
bar  foo
```

iar cea de mutare cu comanda `mv(1)`:

```
$ mv bar baz
$ ls
baz  foo
```

Fişierele se şterg cu comanda `rm(1)`, iar directoarele goale cu comanda `rmdir(1)`:

```
$ pwd
/home/horatiu/wrk/ub/itbi/lab/1/tmp
$ ls
baz  foo
$ rm baz foo
$ ls
$ cd ..
$ cd ../
$ rmdir tmp
```

## 7 Sarcini de laborator

1. Rulaţi toate comenzile descrise în laborator
2. Refaceţi ierarhia directorului `uso` din Secţiunea 1 folosind comenzile `mkdir(1)` şi `touch(1)`.
3. Creaţi un director şi câteva fişiere în acesta. Încercaţi să ştergeţi directorul creat folosind `rm` sau `rmdir`. Ce observaţi? Căutaţi în manualul `rm(1)` cum să ştergeţi recursiv şi folosiţi informaţia pentru a şterge într-o singură instrucţiune directorul creat devreme.
4. Creaţi un director `bin` în `$HOME` şi adăugaţi-l în `$PATH`. Copiaţi un executabil existent (de exemplu `gcc`) şi observaţi cum/dacă se modifică ieşirea comenzii `which(1)` când întrebaţi de executabilul copiat. Dacă nu se schimbă, de ce?