Considerații teoretice

Formatori:
Tutor: Stângaciu Valentin
Tutor: Belu Claudiu-Marcel
+3

Data de începere a cursului:
25.09.2023
Utilizatori înscriși
Calendar
Note

Considerații teoretice

Sistemul de operare Linux

În general comenzile Linux sunt implementate prin intermediul unor programe. Când se execută o comandă, de fapt se execută programul cu numele acelei comenzi. Spre deosebire de Windows, în Linux nu este necesar ca un program executabil să aibă extensia *.exe*. În Linux contează ce fel de litere folosim, literele mari fiind considerate diferite de cele mici (Linux are nume de fișiere/directoare *case sensitive*).

Execuția comenzilor se face prin intermediul unui program numit *terminal* sau *consolă*. Rolul terminalului este simplu: afișează un prompter, așteaptă ca utilizatorul să tasteze o comandă (care este o linie de text terminată cu ENTER), caută programul corespunzător acelei comenzi, îl execută și afișează rezultatul programului. Acest ciclu se repetă până când se închide terminalul. Pentru a porni un terminal se poate accesa meniul *Application -> Accessories -> Terminal*.

Prompterul afișat de terminal este de forma:

```
murphy@murphy-laptop:~/tp$
```

Prompterul se compune din trei părți separate de **@ : $**

```
murphy   @   murphy-laptop   :   ~/tp  $
```

Prima parte, înainte de **@**, indică utilizatorul curent (contul cu care v-ați autentificat). Toate comenzile se vor executa în numele acestui utilizator. În exemplul nostru utilizatorul este *murphy*. A doua parte, între @ și :, indică numele calculatorului pe care lucrați (*murphy-laptop* în exemplu). A treia parte indică directorul curent în care lucrați. În exemplu este *~/tp*. ~ este o scurtătură către directorul *home* al utilizatorului. În general fiecare utilizator are un director propriu, localizat în directorul *home*. În exemplul de mai sus, directorul home al utilizatorului *murphy* va fi de fapt */home/murphy*. Astfel, ~/tp este directorul */home/murphy/tp*.

În sistemele UNIX datele sunt organizate pe disc sub formă de fișiere. Acestea sunt simple fluxuri de octeți, nemaiexistând alte forme de organizare a informației. Există totuși posibilitatea de organizare a fișierelor în directoare, dar acestea sunt la rândul lor tot fișiere ce nu pot fi scrise de utilizator. UNIX are un sistem de fișiere arborescent. Spre deosebire de sistemul de fisiere DOS/Windows, separatorul între componentele numelui de fişier (directoare şi numele propriu-zis) este caracterul / (slash) şi nu \ (backslash). De asemenea, numele de fişiere nu conţin un identificator al discului fizic (A:, C:, etc.), ci întreaga ierarhie de fişiere porneşte de la o rădăcină unică, notată cu '/' (slash). Un exemplu de organizare a sistemului de fisier UNIX poate fi și următorul (extras de pe un sistem Linux Debian 9.8 - folosind comanda *tree*):

```
/
|-- bin
|-- boot
|-- dev
|-- etc
|-- home
|   |-- student
|   |-- student1
|-- lib
|-- lib32
|-- lib64
|-- media
|-- mnt
|-- opt
|-- proc
|-- root
|-- run
|-- sbin
|-- srv
|-- sys
|-- tmp
|-- usr
|   |-- bin
|   |-- games
|   |-- include
|   |-- lib
|   |-- lib32
|   |-- local
|   |-- sbin
|   |-- share
|   |-- src
|-- var|--
initrd.img -> boot/initrd.img-4.9.0-8-amd64
|--
initrd.img.old -> boot/initrd.img-4.9.0-7-amd64
|--
vmlinuz -> boot/vmlinuz-4.9.0-8-amd64
|--
vmlinuz.old -> boot/vmlinuz-4.9.0-7-amd64
```

- /bin - prescurtarea de la "*binaries*" - acest director conține unele programe utilitare fundamentale (cum ar fi ls, cp,...)
- /boot - conține fișierele necesare pentru a porni cu succes un sistem UNIX: fișierul ce reprezintă kernel-ul (nucleul), fișierul initrd (initial ramdisk) ce reprezintă sistemul de fișiere inițial folosit la pornirea kernel-ului, fișierul cu opțiunile de configurare cu care a fost compilat kernel-ul, fisșiere caracteristice bootloader-ului
- /dev - numele provine de la "*devices*"; conține fișiere speciale ce reprezintă echipementele hardware ale sistemului (ex: fisierul /dev/mem reprezintă întreaga memorie - spațiu de adrese - a sistemului, /dev/sda - reprezintă primul hard-disk al sistemului)
- /etc - este un director dedicat stocării fișierelor de configurare ale sistemului și ale programelor și serviciilor ce rulează în sistem
- /home - conține directoarele "personale" ale utilizatorilor. Fiecare utilizator are asignat un director personal în care are drepturi depline (poate să creeze, să șteargă, să editeze și eventual să execute fișiere), dar nu are dreptul să-și șteargă acest director. Această organizare a fost preluată ulterior de către sistemele Microsoft Windows (Windows 7, Windows 8, Windows 10 ...) prin directorul C:\Users
- /lib, /lib32, /lib64 - numele provine de la *libraries*; conține bibliotecile de bază ale sistemului, în principal cele folosite de programele din /bin. Variantele /lib32, /lib64 sunt prezente pentru a oferi suport pentru arhitecturi diferite. Bibliotecile conținute în acest director sunt de obicei biblioteci link-editate dinamic (shared libraries - cu extensia .so)
- /media - reprezinta un director folosit ca punct de montare implicit al dispozitivelor de stocare temporare (removable devices) precum stick-uri USB, HDD externe, etc.
- /mnt - numele provine de la *mount* și reprezinta directorul folosit în mod comun de administratorii sistemului pentru a monta diferite dispozitive de stocare, în principal hard-disk-urile
- /opt - numele provine de la *optional* și reprezintă un director folosit pentru a instala unele aplicații.
- /proc - acest director există pe disc, dar conținutul lui nu reprezintă fișiere propriu-zise. În acest director se montează sistemul de fișiere *procfs* ce conține, tot sub formă de fișiere, informații despre procesele ce rulează în sistem. Aceste fișiere nu ocupa spațiu efectiv pe disc ci reprezintă doar o interfață
- /root - directorul de home al utilizatorului *root*. Acest utilizator are drepturi depline în sistem fiind numit și *superuser*.
- /sbin - numele directorului provine de la "*system binaries*" și conține utilitare fundamentale necesare pornirii sistemului
- /srv - denumit și "*server data*"; conține anumite date ale unor servicii ce pot rula în sistem
- /sys - acest director există pe disc, dar conținutul lui nu reprezintă fișiere propriu-zise (analog cu /proc). În acest director se montează sistemul de fișiere *sysfs* ce conține fișiere pentru a accesa mai ușor anumite dispozitive din sistem
- /tmp - director folosit pentru fișiere temporare. Nu este definit dacă fișierele stocate aici rămân și după repornirea sistemului. În majoritatea cazurilor, conținutul acestui director este șters la pornirea sistemului. În principiu, orice utilizator are dreptul să scrie fișiere în acest director, dar nu are voie să execute fișiere din acest director.

- /usr - denumit și "*user filesystem*" și contine programe (fișiere binare - în /usr/bin), biblioteci statice sau dinamice (/usr/lib), fișiere header (/usr/include) ce nu sunt critice pentru sistem.
- /var - provine de la cuvântul "*variable*" și conține fișiere ce se modifică mai des față de restul sistemului. În această categorie pot intra fișiere de log, fișiere temporare create de server-ul de mail, etc.

Este important de menționat faptul că sistemele UNIX/Linux nu folosesc extensia la fișiere, aceasta fiind folosită doar pentru a facilita utilizarea sistemului. Un fișier poate fi executabil indiferent de extensia lui (poate fi executabil un fișier fără extensie, cu extensia .jpg sau .txt). Extensiile sunt folosite de către utilizator pentru a identifica cu ușurință tipul de fișier. De asemenea, unele programe mai pot folosi extensia pentru a se adapta mai ușor. De exemplu, un editor de texte dacă deschide un fișier .c, .html, .java ar putea activa facilitatea de syntax highlight corespunzătoare.

Comenzile Linux au următorul format:

```
nume_comandă opțiuni fișiere_sau_directoare
```

Numele comenzii este programul care va fi executat. Acest program este căutat în sistem și executat. Opțiunile încep cu - (minus) și diferă în funcție de program. Ele se folosesc pentru a modifica felul în care se execută acea comandă.

Exemplu: dacă vom tasta în consolă comanda:

```
murphy@murphy-laptop:~/tp$ ls -l /usr
```

Linux va lansa în execuție programul *ls* cu 2 argumente: *-l*, care este o opțiune și */usr*, care este un director. Se poate face o analogie cu o funcție: programul este numele funcției, iar opțiunile, fișierele și directoarele sunt argumentele funcției.

Consola menține istoria comenzilor tastate, astfel încât putem apăsa tastele sus/jos pentru a readuce în linia de editare o comandă anterioară. Această facilitate ne ajută să nu trebuiască să rescriem complet o comandă, ci doar să modificăm una anterioară. În plus, consola știe să completeze automat numele unui fișier/director. De exemplu, dacă avem un fișier numit "exemplu.c", putem să scriem doar "ex" și să apăsăm TAB, restul numelui fiind completat automat. Completarea se face doar dacă există un singur fișier cu prefixul scris de noi. Dacă există mai multe, nu se va face completarea, fiindcă nu se știe ce nume să se completeze. În această situație puteți apăsa TAB de 2 ori, pentru a vedea toate fișierele cu acel prefix.

Comenzi de bază Linux

- **ls** *director* - (list) afișează conținutul directorului specificat (fișiere și subdirectoare). Dacă nu se specifică un director, se va afișa conținutul directorului curent. *ls* are multe opțiuni, prin care se poate controla formatul afișării, ce anume să se afișeze, ordinea, etc.

ls - afișează conținutul directorului curent
ls *.c - afișează toate fișierele din directorul curent care se termină cu *.c*. Steluța este interpretată ca fiind orice succesiune de caractere, inclusiv nulă (ex: tema* - toate fișierele care încep cu literele *tema*)
ls -l /usr - afișează conținutul directorului */usr* în format lung (cu mai multe informații, printre care drepturi de folosire, deținător fișier, dimensiune, dată de modificare).

- **pwd** - (print working directory) afișează directorul curent

- **cd** *director* - (change directory) ne duce în directorul specificat, care va deveni directorul curent. Există două metode de a specifica un director:

Dacă numele directorului începe cu / (slash), ceea ce semnifică directorul rădăcină (root, părintele tuturor subdirectoarelor) sau cu ~ (tilda), directorul *home* al utilizatorului curent, specificarea directorului se numește **absolută**. În acest fel, putem specifica un anumit director din oricare alt director ne-am afla.
Dacă numele directorului începe cu alte caractere (ex: cu litere sau punct), specificarea se numește **relativă** la directorul curent. În acest fel, putem să specificăm directoare care sunt într-o anumită relație cu directorul curent (ex: subdirectoare, directoare părinți).

cd ~/tema1 - de oriunde am fi (specificare absolută), ne duce în subdirectorul *tema1* al directorului *home*
cd .. - ne duce în directorul părinte (**..** (două puncte) înseamnă directorul părinte, iar **.** (un punct) înseamnă directorul curent)
cd ../.. - ne duce în părintele de ordinul 2 al directorului curent
cd p5 - ne duce în subdirectorul *p5* al directorului curent

- **mkdir** *director* - (make directory) creează un nou director. Se poate specifica opțional și unde anume să fie creat noul director.

mkdir tema2 - creează subdirectorul *tema2* în directorul curent
mkdir tema2/p1 - creează subdirectorul *p1* în directorul *tema2*
mkdir ~/tema3 - creează subdirectorul *tema3* în directorul *home*

- **rmdir** *director* - (remove directory) șterge un director. Directorul trebuie să fie gol (să nu conțină fișiere sau alte subdirectoare). Dacă se dorește ștergerea unui director care nu este gol, se va folosi comanda "**rm -r**" (remove recursive).

rmdir tema1 - șterge subdirectorul *tema1* din directorul curent
rm -r ~/p2 - șterge subdirectorul p2 din directorul home, împreună cu tot conținutul său (fișiere și subdirectoare)
rm *fișier_sau_director* - (remove) șterge un fișier sau un director. Directoarele se șterg doar dacă se folosește și opțiunea **-r**, și atunci se șterg în

mod recursiv, chiar dacă nu sunt goale.
rm *.o - șterge toate fișierele cu extensia *.o* din directorul curent
rm -r test1 - șterge subdirectorul *test1* din directorul curent, împreună cu tot conținutul său

- **cat** *fișier1 fișier2 ... fișierN* - (concatenate) afișează pe ecran, în ordinea dată, conținutul fișierelor specificate. Deoarece în Linux afișarea se poate redirecționa către un fișier, *cat* se poate folosi pentru a concatena mai multe fișiere text într-unul singur. Redirecționarea se face punând "> *nume_fișer_destinație*" la finalul liniei de comandă.

cat 1.c - afișează conținutul fișierului *1.c*
cat pers1.txt pers2.txt > pers.txt - concatenează conținutul fișierelor *pers1.txt* și *pers2.txt* în fișierul *pers.txt*
cp *sursă destinație* - (copy) copiază fișierul sau directorul sursă în destinație. Pentru a se copia directoare, trebuie specificată opțiunea **-r**.
cp 1.c 2.c - crează o copie denumită **2.c** a fișierului **1.c**
cp -r test1 backup - crează o copie denumită *backup* a subdirectorului *test1* și a întregului său conținut
cp tema/*.c src - copiază toate fișierele cu extensia *.c* din subdirectorul tema în directorul *src*

- **mv** *sursă destinație* - (move) funcționează la fel ca și *cp* (copy), dar mută/redenumește sursa în destinație
- **touch** *fișier* - dacă fișierul specificat nu există, creează un fișier vid având numele specificat. Dacă fișierul există, îi setează data ultimei accesări ca fiind data execuției comenzii *touch*, lăsând conținutul neatins.

touch 1.txt - creează un fișier vid având numele *1.txt* (dacă acesta nu există deja)

- Comanda **man -** Din punct de vedere didactic, probabil una dintre cele mai importante comenzi este cea care afișează paginile de manual atât pentru comenzi de shell script și programe lansate de interpretorul de comenzi, cât și pentru prototipuri de funcții C. Se va discuta în continuare despre comanda **man**. Cea mai simplă formă de apel ar fi următoarea:

man [opțiuni] [secțiune] comandă

Argumentele dintre parantezele drepte [...] sunt opționale.Paginile de manual se împart pe secțiuni. În momentul de față, în sistemele Linux există 9 secțiuni:

1. Secțiunea 1 - descrie comenzile standard (programe executabile și comenzi shell script)
2. Secțiunea 2 - apeluri sistem UNIX apelabile în limbajul C
3. Secțiunea 3 - funcțiile de bibliotecă C
4. Secțiunea 4 - informații despre fișierele speciale (în principal cele din /dev)
5. Secțiunea 5 - informații despre convențiile și formatele anumitor fișiere specifice sistemului
6. Secțiunea 6 - manuale de la jocurile din Linux
7. Secțiunea 7 - informații despre diverse teme ce nu pot fi incluse în alte secțiuni (spre exemplu man 7 signal)
8. Secțiunea 8 - comenzi de administrare a sistemului (de obicei doar pentru userul root)
9. Secțiunea 9 - rutine kernel

Argumentul *secțiune* este opțional. Astfel, în situația în care nu se specifică secțiunea pentru comanda/funcția pentru care se cere pagina de manual, programul man va căuta comanda/funcția în secțiuni în ordine crescătoare începând cu secțiunea 1. Programul man va afișa informațiile din prima secțiune în care funcția/comanda a fost găsită. În cazul în care există funcții/comenzi diferite, dar cu același nume și sunt în secțiuni diferite este necesar să se specifice secțiunile, în caz contrar se va furniza doar prima apariție.

Exemple:
· Afisarea paginii de manual pentru functia printf

```
man printf
```

· Afisarea paginii de manual pentru comanda cat

```
man cat
```

Editarea de fișiere text

În decursul acestui laborator se încurajează cât mai mult utilizarea sistemului in linie de comandă chiar și cu existența mediului grafic activat. Așadar se va descuraja deschiderea editorului de texte din meniurile interfeței grafice. Nu se vor impune la laborator editoarele de texte, studenții având libertatea să folosească editoarele preferate (dacă sunt instalate).

Este important de menționat diferența dintre un editor de documente și un editor de text. Editorul de documente nu este un editor de text, acesta salvând informația in fișiere binare (sau arhive, xml, etc). Exemple de editoare de documente: Microsoft Word, Libre Office Writer.

Un editor de texte va edita fișierul în mod text, iar rezultatul va fi un fișier text fără alte informații adiționale. Exemple de editoare de texte:

- Windows: Notepad, Notepad++
- Linux: Emacs, Gedit, vi, vim, mcedit, kate, etc.

Pentru apelarea corectă a unui editor de texte din linia de comandă în Linux se va folosi următoarea sintaxă:

```
<editor_de_texte> fișier_text &
```

Exemplu

```
emacs fisier.c &
```

Semnul '&' (ampersand) are rolul de a "trece" editorul de texte invocat din terminal în background-ul acestuia. În această situație, după execuție, terminalul va rămâne "liber" și va mai putea fi folosit fără a se închide editorul de texte. În cazul în care nu se folosește semnul '&' editorul de texte va bloca terminalul respectiv și acesta nu va mai putea fi folosit decât după închiderea editorului.

Atenție! închiderea terminalului va duce la închiderea forțată a editorului ceea ce poate duce la pierderea datelor (dacă nu s-a salvat fișierul). Această metodă se aplică doar atunci când se folosește un mediu grafic... în mod de text absolut, în Linux, nu se recomandă această utilizare. La laborator se va folosi mediul grafic.

Pentru afișarea unui fișier text de mici dimensiuni se recomandș folosirea comenzii cat în loc de deschiderea acestuia într-un terminal.

Primul program C

Codul de mai jos prezintă un prim program C, care afișează pe ecran cuvântul *salut*.

```
// primul program C
#include <stdio.h>

int main() {
    printf("salut");
    return 0;
}
```

Acest cod este compus din următoarele elemente:

- **// text** - orice text care apare pe linie după **//** este considerat ca fiind comentariu și el nu contează pentru programul în sine. Comentariile sunt foarte utile pentru programator pentru a-și documenta codul.

- **#include <stdio.h>** - copiază în cod conținutul fișierului specificat. În acest caz, fișierul stdio.h (*standard input/output . header*) conține declarații pentru funcțiile de intrare (de la tastatură sau din fișier) și cele de ieșire (afișare pe ecran sau scriere în fișier). Programul a avut nevoie de stdio.h pentru că în el se află declarația funcției *printf*.

- **int main() {...}** - între aceste linii se definește funcția *main*. Tot ceea ce se află între **{...}** reprezintă corpul funcției. Funcția *main* este apelată automat la execuția programului, astfel încât tot ceea ce se află în interiorul ei va fi executat. Mai multe despre funcții se vor discuta în laboratoarele ulterioare. Deocamdată reținem faptul că tot codul nostru va fi inclus în această funcție.

- **printf("salut");** - reprezintă apelul funcției *printf* (print formatted). Această funcție afișează pe ecran argumentele ei. În acest caz *printf* are un singur argument, *"salut"*. Acest argument este un *șir de caractere*, care în C se pune între ghilimele. În general, în C instrucțiunile trebuie încheiate cu *punct și virgulă*. După acoladele închise nu este necesar să se pună punct și virgulă, deoarece chiar acoladele în sine au rol de separator.

- **return 0;** - încheie funcția și returnează către apelantul funcției valoarea specificată. Pentru funcția *main*, valoarea 0 semnifică faptul că funcția s-a încheiat cu succes. În caz de eroare, funcția *main* ar fi returnat un cod de eroare, diferit de 0.

Codul descris mai sus se va implementa prin scrierea lui într-un fișier text după instrucțiunile anterioare, folosind un editor de texte. Este foarte important ca noul fișier text să aibă extensia ".c" astfel ca editorul de texte să recunoască limbajul și să activeze evidențierea sintaxei (syntax highlight)

Compilarea unui program C

Compilarea este procesul prin care instrucțiunile dintr-un fișier C sunt translatate în instrucțiuni cod mașină, instrucțiuni care pot fi executate de microprocesorul gazdă (Intel, AMD, ARM, etc). Aplicațiile IDE (Integrated Development Environment), ca de exemplu *Code::Blocks* sau *Visual Studio*, de fapt constă dintr-o suită de programe, cum ar fi editoare de cod, depanatoare, managere de proiecte, compilatoare și altele. La comanda de compilare sau de execuție, aceste aplicații apelează compilatorul și acesta din urmă realizează translatarea efectivă a codului sursă în cod mașină.

Există mai multe compilatoare de C, dintre care cel mai răspândit în Linux este **gcc** (GNU C Compiler). Acesta are o serie întreagă de opțiuni, dar o modalitate simplă de a-l folosi este următoarea:

```
gcc -Wall -o nume_executabil nume_fișier_C
```

În această formă, *gcc* compilează fișierul C specificat pentru a rezulta la ieșire (output, **-o**) fișierul executabil cu numele dat. Opțiunea **-Wall** se folosește pentru a se afișa toate atenționările (warnings, **-W**, de exemplu variabile neinițializate), ceea ce ajută mult în faza de dezvoltare a unui program.

Dacă compilarea a avut loc cu succes, *gcc* nu afișează nimic, iar pe disc va apărea fișierul executabil specificat, care va putea fi apoi executat. Dacă au avut loc erori la compilare, acestea vor fi afișate pe ecran și nu se va produce fișierul executabil.

Exemplu: să presupunem că în fișierul *test.c* avem următorul cod:

```
#include <stdio.h>
int main(void)
{
    printf("Oare s-a compilat?\n");
    return 0;
}
```

Conținutul directorului curent este:

```
murphy@murphy-laptop:~/tp$ ls -l
total 8
drwxr-xr-x 2 murphy murphy 4096 2009-02-17 23:42 laborator01
-rw-r--r-- 1 murphy murphy   85 2009-02-18 00:22 test.c
```

Compilăm fișierul:

```
murphy@murphy-laptop:~/tp$ gcc -Wall -o test test.c
```

Dacă în program ar fi erori, ele ar fi afișate pe ecran. Dacă nu se afișează nimic, înseamnă că programul a fost compilat cu succes. Acum directorul curent conține:

```
murphy@murphy-laptop:~/tp$ ls -l
total 16
drwxr-xr-x 2 murphy murphy 4096 2009-02-17 23:42 laborator01
-rwxr-xr-x 1 murphy murphy 6369 2009-02-18 00:24 test
-rw-r--r-- 1 murphy murphy   85 2009-02-18 00:22 test.c
```

A apărut fișierul *test*, care este rezultatul compilării de către *gcc* a programului sursă *test.c*. *test* este un fișier *binar* (nu este direct citibil de către oameni, în opoziție cu fișierele *text*, care se pot citi/modifica folosind un editor de text). Putem verifica aceasta listând conținutul său pe ecran (*cat test*), pentru a constata astfel că în general este neinteligibil. Pentru a executa programul pe care tocmai l-am compilat, tastăm:

```
murphy@murphy-laptop:~/tp$ ./test
```

În urma acestei comenzi, în consolă va trebui să apară mesajul din programul sursă. Când tastăm o comandă în Linux, sistemul de operare (SO) o va căuta în anumite directoare prestabilite. Directorul curent nu face parte dintre aceste directoare, de aceea trebuie să specificăm faptul că *test* se află în directorul curent (**.**), altfel SO nu-l va găsi.

◄ Examen Prezentarea 3 - Cod 2

Sari la...

Teme și aplicații ►

Contactați serviciul de asistență

Sunteți conectat în calitate de [redacted] (Delogare)
S1-L-AC-CTIRO1-PC

Meniul meu
Profil
Preferinte
Calendar
ZOOM
Română (ro)
English (en)
Română (ro)

Rezumatul păstrării datelor
Politici utilizare site