

SD 2019 - 2020  
Tema 1 : Filesystem Minishell

09.03.2020

**Deadline hard: 29.03.2020**

Responsabili temă: Ștefan Laurențiu [steflaurentiu@gmail.com](mailto:steflaurentiu@gmail.com)  
Nițu Ioan-Florin-Cătălin [nitu.catalin1998@gmail.com](mailto:nitu.catalin1998@gmail.com)

Ultima modificare: **08.03.2020**

## 0 Schimbări

## 1 Introducere

În știința calculatoarelor, un sistem de fișiere (în eng. filesystem sau file system, adesea prescurtat prin fs), controlează modul în care datele sunt stocate și preluate. Fără un sistem de fișiere, datele situate într-un mediu de stocare ar fi un corp mare de date, fără niciun fel de a preciza unde se oprește o bucată de date și unde începe următoarea. Prin separarea datelor în bucăți și acordarea unui nume fiecărei părți, datele sunt ușor izolate și identificate. Luându-și numele din modul în care este denumit sistemul de gestionare a datelor pe bază de hârtie, fiecare grup de date este denumit „fișier”. Normele de structură și logică utilizate pentru gestionarea grupurilor de date și numele acestora alcătuiesc un „sistem de fișiere”.

Utilitatea sistemului de fișiere este una destul de clară, utilizatorii din ziua de azi fiind obișnuiți cu structura acestuia, precum și să navigheze în acesta prin diferite modalități: prin GUI (ex. File Explorer pe Windows) sau prin CLI (ex. terminal pe Linux).

**Scopul acestei teme** este de a implementa un sistem de fișiere minimal, cu o mulțime de operații asociate de CLI, foarte similare cu cele de pe terminalul unui sistem UNIX, pentru a manipula respectivul sistem de fișiere. Structura acestuia va fi similară cu cea de pe un sistem UNIX-based, prin împărțirea în fișiere și directoare, plecându-se de la directorul rădăcină **[0]** (mai multe detalii la secțiunea **2**). Implementarea acestui sistem de fișiere se va face folosind cunoștințele pe care le-ați dobândit până acum la disciplina Structuri de Date.

## 2 Structură

Sistemul de fișiere pe care dorim să-l implementăm este alcătuit din două tipuri de elemente: fișier **[1]** și director **[2]**.

Un **fișier** este unitatea de bază a sistemului de fișiere și are ca rol să rețină date într-un anumit format (text sau binar).

Un **director** este o structură de catalogare a unui sistem de fișiere, care conține referințe la alte fișiere din calculator și, eventual, la alte directoare. Prin urmare, un sistem de fișiere are o structură arborescentă, care pleacă de la directorul rădăcină (root filesystem, '/').

Ne propunem să implementăm acest sistem de fișiere folosind listele înlănțuite. Pentru reprezentarea unui fișier, respectiv a unui director, se vor pune în schelet (în fișierul **file.h**) definițiile structurilor aferente (în cadrul acestei teme, structurile de fișier, respectiv de director vor avea o reprezentare mai simplă, care conține numai metadatele strict necesare pentru a implementa operațiile de shell dorite, ce vor fi enunțate în continuare; de reținut faptul că, într-un sistem de operare modern, structura de fișier conține mult mai multe metadate: mai multe detalii la **[3]**).

Vom referi în continuare structurile de fișier și director prin **File**, respectiv **Directory**.

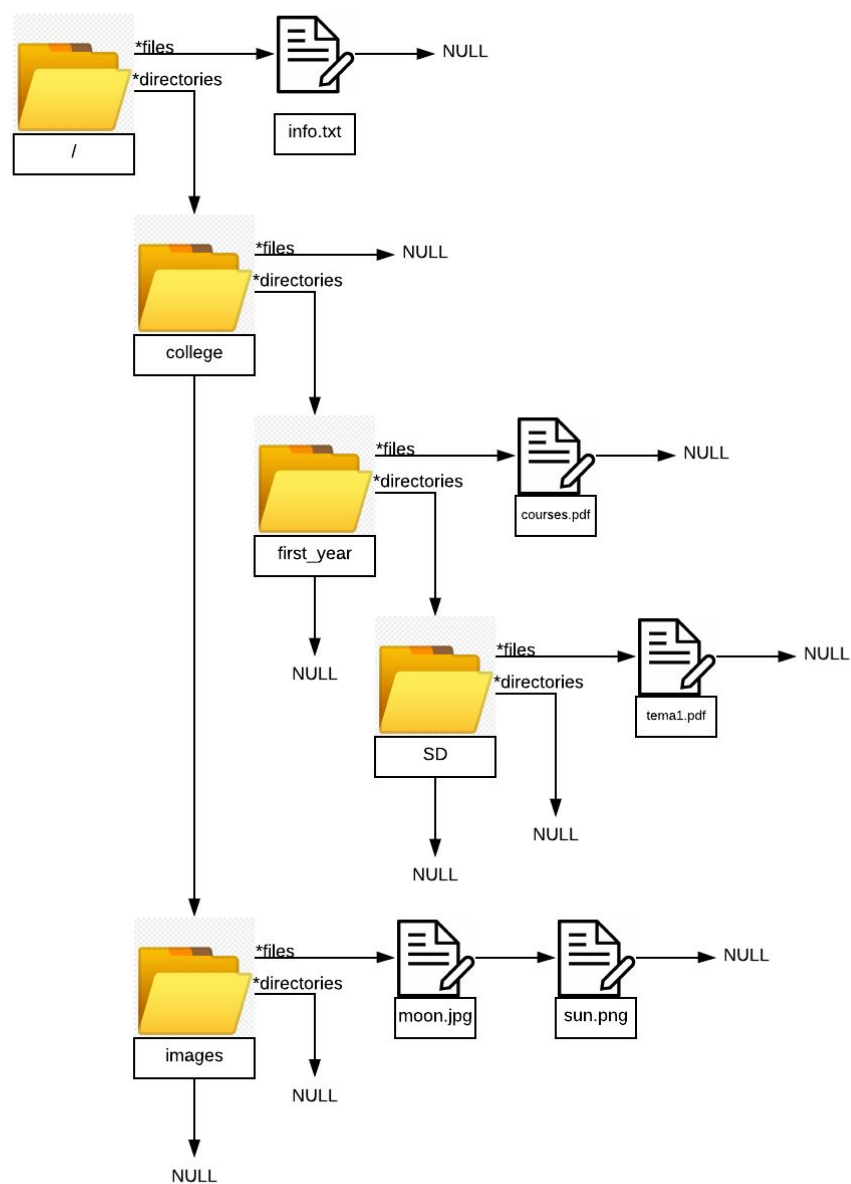
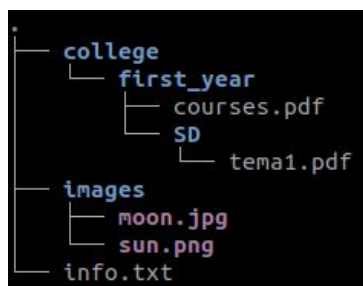
**Structura Directory** va avea în componență **o referință la o listă înlănțuită de fișiere și o referință la o listă înlănțuită de directoare**. Semnificația acestor liste este următoarea:

- Lista înlănțuită de fișiere conține numai elemente de tipul **File**, acestea reprezentând fișierele accesibile din directorul curent
- Lista înlănțuită de directoare conține numai elemente de tipul **Directory**, acestea reprezentând directoarele accesibile din directorul curent

### Observații:

- Întrucât în reprezentarea precizată mai sus avem două tipuri de liste (cu elemente de tipul **File** și cu elemente de tipul **Directory**), o implementare folosind liste generice **[4][5][6]** ar fi una mai elegantă decât două implementări foarte similare de listă (pentru modularizare, evitare cod duplicat etc.). Totuși, nu o să fie restricții de implementări cât timp se folosesc listele înlănțuite
- Dacă într-un director anume nu se află niciun fișier, lista de fișiere va fi vidă
- Dacă într-un director anume nu se află niciun alt director, lista de directoare va fi vidă

Mai jos găsiți un exemplu de ierarhie de fișiere și directoare (după rularea utilitarului **tree** pe Linux) și imediat dedesubt o schemă ce evidențiază cum arată reprezentarea internă a sistemului de fișiere ce respectă structura descrisă mai sus.



### 3 Cerințe

Ne propunem să implementăm o aplicație de CLI care să gestioneze sistemul de fișiere. Pentru aceasta, ne dorim să implementăm o serie de operații asemănătoare cu CLI-ul din Linux, de manipulare a sistemului de fișiere, însă într-o formă mai simplificată, pentru a reduce complexitatea implementării. Operațiile sunt descrise în subsecțiunile următoare.

Observație: Operațiile de mai jos se realizează interactiv, la orice moment de timp existând un director curent ("unde ne aflăm"). La început, se va apela operația "create fs", care are în același timp rolul de a seta directorul curent la rădăcina "/".

#### 3.1 Operații de inițializare și ștergere ale sistemului de fișiere

##### **\$ create fs**

- creează rădăcina sistemului de fișiere (directorul cu numele "/")

##### **\$ delete fs**

- șterge DOAR directorul rădăcină al sistemului de fișiere (nu șterge recursiv conținutul subdirectoarelor acestuia sau fișierele acestuia; considerăm că acestea au fost deja șterse prin operații rmdir și rm realizate anterior)

#### 3.2 Operație de creare a unui fișier

##### **\$ touch <filename> <content>**

- adaugă un fișier nou cu numele <filename> și conținutul <content> în directorul curent (trebuie completată și lungimea fișierului în structura File)
- pentru simplitate, content va fi un șir de caractere care nu conține spații

Ex. touch test.in thisisthecontent

- Se creează în directorul curent un fișier cu numele "test.in" și cu conținutul "thisisthecontent"

#### 3.3 Operație de creare a unui director

##### **\$ mkdir <directoryname>**

- adaugă un director nou cu numele <directoryname> în directorul curent

Ex. mkdir my\_dir

- Se creează în directorul curent un director cu numele "my\_dir"

### 3.4 Operație de listare a conținutului unui director

#### \$ ls

- afișează în ordine lexicografică numele fișierelor ce se află în directorul curent, urmate de numele directoarelor ce se află în directorul curent

**Atenție! Pentru o afișare lexicografică a numelor fișierelor și a directoarelor, inserările în lista de fișiere sau în lista de directoare trebuie realizată astfel încât tot timpul acestea să aibă intrările în ordine lexicografică după nume! Se va scădea din punctaj pentru implementările care sortează listele la fiecare operație de tip "ls".**

Această funcție este folosită pentru a valida implementarea funcțiilor de creare/ștergere a fișierelor/directoarelor. Fără implementarea acestora nu se vor putea testa implementările respective.

Ex. ne aflăm în rădăcina sistemului de fișiere din schema de mai sus

```
ls
info.txt college images
```

### 3.5 Operație de afișare a căii directorului curent

#### \$ pwd

- afișează calea absolută a directorului curent, sub o formă similară celei din Linux (folosind delimitatorul '/')

Ex. ne aflăm în directorul "SD" din schema de mai sus

```
pwd
/college/first_year/SD
```

### 3.6 Operație de schimbare a directorului

#### \$ cd <directoryname>

- schimbă directorul curent în directorul <directoryname> (dacă acesta există)
- dacă în directorul curent nu există directorul cu numele <directoryname>, se va afișa mesajul de eroare "Cannot move to '<directoryname>': No such directory!"
- <directoryname> poate fi și "." cu aceeași semnificație ca în Linux, se va schimba directorul cu cel părinte al celui curent (se asigură că mereu există un părinte dacă este dat "..")

Ex. ne aflăm în rădăcina sistemului de fișiere din schema de mai sus

```
pwd
/  
cd images  
pwd  
/images  
cd test  
Cannot move to 'test': No such directory!
```

### 3.7 Operație de afișare a conținutului sistemului de fișiere

#### \$ tree

- afișează, începând de la directorul curent, sub o formă arborescentă, conținutul subsistemului de fișiere astfel: se parcurg mai întâi fișierele din director și se afișează numele acestora, după care, pentru fiecare director, se afișează numele acestuia și se reia procedeul (vezi utilitarul tree pe Linux)
- la fel ca la operația ls, numele fișierelor și directoarelor trebuie afișate în ordine lexicografică după nume
- se va folosi un nivel de indentare de 4 spații la fiecare nivel de adâncime în sistemul de fișiere

Ex. ne aflăm în rădăcina sistemului de fișiere din schema de mai sus

```
tree  
/  
  info.txt  
  college  
    first_year  
      courses.pdf  
      SD  
        tema1.pdf  
  images  
    moon.jpg  
    sun.png
```

### 3.8 Operație de ștergere a unui fișier

#### \$ rm <filename>

- șterge fișierul cu numele <filename> din directorul curent
- dacă în directorul curent nu există fișierul cu numele <filename>, se va afișa mesajul de eroare "Cannot remove '<filename>': No such file!"

Ex. ne aflăm în rădăcina sistemului de fișiere din schema de mai sus

```
ls
info.txt college images
rm info.txt
ls
college images
rm test
Cannot remove 'test': No such file!
```

### 3.9 Operație de ștergere a unui director

#### \$ rmdir <directoryname>

- șterge directorul cu numele <directoryname> din directorul curent (dacă acesta există) și tot conținutul acestuia într-o manieră recursivă (trebuie eliberat tot subsistemul de fișiere care pleacă de la directorul <directoryname>)

- dacă în directorul curent nu există directorul cu numele <directoryname>, se va afișa mesajul de eroare "Cannot remove '<directoryname>': No such directory!"

Ex. ne aflăm în rădăcina sistemului de fișiere din schema de mai sus

```
ls
info.txt college images
rmdir college
tree
/
  images
    moon.jpg
    sun.png
rmdir test
Cannot remove 'test': No such directory!
```

### 3.10 Operație de căutare a unui fișier [BONUS]

#### \$ find <max\_depth> <min\_size> <max\_size> <subcontent>

- caută, începând de la directorul curent, cu o adâncime maximă de <max\_depth>, fișierele care au o dimensiune în intervalul [min\_size, max\_size] și care conțin textul <subcontent> (este inclus în conținutul fișierului), iar apoi afișează numele fișierelor găsite.

Ex. în acest exemplu avem doar rădăcina sistemului de fișiere creată

- căutarea se face urmând o parcurgere în adâncime (nodul inițial se află la adâncime 0)

```
touch a.in ana
touch b.in mere
mkdir subdir1
cd subdir1
touch c.in anaaremere
touch d.in anaa
```

```
mkdir subdir11
cd subdir11
touch e.in ana
cd ..
cd ..
find 1 3 5 ana
a.in d.in
# b.in nu a fost afișat deoarece nu conține subșirul "ana"
# c.in nu a fost afișat deoarece nu are dimensiunea în intervalul [3, 5]
# e.in nu a fost afișat deoarece se ajunge la adâncime 2 în parcurgerea directoarelor
```

**Observații:**

- Se asigură corectitudinea operațiilor la testare (cu excepția operațiilor touch, mkdir sau cd unde pot fi erori și acestea trebuie semnalate)!
- NU se pot primi căi relative/absolute ca în terminalul de Linux!
- Operațiile sunt incrementale; prin urmare, la testarea unei operații cel mai probabil se folosesc o parte din operațiile anterioare; se recomandă implementarea acestora în ordinea de mai sus!

## 4 Format input/output

Aplicația rezultată se va comporta ca un prompt interactiv, ce așteaptă tot timpul comenzi de la **stdin** și afișează rezultatul acestora (dacă este cazul) la **stdout**. Se asigură că prima operație va fi întotdeauna "create fs" iar ultima operație va fi "delete fs", care are și rolul de a opri execuția aplicației. Operațiile vor fi delimitate printr-un newline. De asemenea, după fiecare output al unei operații trebuie afișat un newline.

Mai jos se găsește un exemplu integral de comenzi, care creează structura prezentată în schema de mai sus și în final dealocă întreaga memorie. Ce este boldat este afișat la stdout.

```
$ ./fsminishell
create fs
mkdir images
touch info.txt
mkdir college
ls
info.txt college images
cd images
touch sun.png
touch moon.jpg
cd ..
cd college
mkdir first_year
```



```

touch courses.pdf
mkdir SD
cd SD
pwd
/college/first_year/SD
touch tema1.pdf
cd ..
tree
first_year
  courses.pdf
  SD
    Tema1.pdf
cd ..
cd ..
rmdir college
rmdir images
rm info.txt
delete fs

```

## 5 Punctaj

Cerință	Punctaj	Observații
3.1 create fs, delete fs	30p	6 teste de 5 puncte cu dificultate incrementală
3.2 touch		
3.3 mkdir		
3.4 ls		
3.5 pwd	20p	4 teste de 5 puncte cu dificultate incrementală
3.6 cd		
3.7 tree	20p	2 teste de 10 puncte
3.8 rm	20p	10p corectitudine structură fișiere (2 teste de 5 puncte) 10p lipsă leak-uri de memorie (1 test rulat cu utilitarul <b>valgrind</b> )
3.9 rmdir		
README și Coding Style	10p	Sunt verificate parțial de checker (se vor verifica și manual la corectare)
3.10 find	20p	BONUS (2 teste de 10 puncte)

## 6 Arhiva temei

Temele trebuie să fie încărcate pe vmchecker [7]. NU se acceptă teme trimise pe email sau altfel decât prin intermediul vmchecker-ului. O rezolvare constă într-o **arhivă de tip zip** care conține **toate fișierele sursă** alături de un **Makefile** ce va fi folosit pentru compilare și un fișier **README** cu detaliile implementării.

Trebuie respectate următoarele reguli:

- **Makefile-ul** trebuie să aibă obligatoriu regulile pentru **build** și **clean**
- **Regula build** trebuie să aibă ca efect compilarea surselor și crearea binarului cu numele **fsminishell**. Fișierele sursă trebuie să fie în rădăcina arhivei. Mai exact, nu faceți un director în care se află toate sursele și arhivați directorul, ci selectați toate sursele și faceți arhivă din ele: în directorul cu sursele dați comanda  

```
zip NumeArhiva.zip sursa1 sursa2 ... sursan Makefile README
```

Arhiva trebuie să aibă tipul **zip** și trebuie să fie numită astfel:

**Nume\_ToatePrenumele\_GrupaSeria\_Tema1SD.zip**

De exemplu, Ioniță Radu Constantin, grupa 311CC va trimite o arhivă cu numele "Ionita\_RaduConstantin\_311CC\_Tema1SD.zip".

**NU se vor puncta temele care nu respectă specificațiile anterioare.**

## 7 Precizări

- Tema trebuie predată cel târziu pe **29 martie 2020, ora 23:55**. Deadline-ul temei este **HARD**.
- Tema este **INDIVIDUALĂ**. Orice tentativă de copiere va fi sancționată conform regulamentului.
- Este **obligatorie** implementarea folosind **liste înlănțuite**! O implementare care **nu respectă** această restricție **nu va fi punctată**.
- Citiți cu atenție observațiile din cadrul enunțului. Pentru anumite nerespectări ale cerinței se acordă depunctări.
- Se asigură că numele fișierelor și al directoarelor va fi unic în cadrul aceluiasi director.
- În arhiva checker-ului se află și fișierul "**instructions.txt**" în care aveți mai multe detalii despre cum să rulați checker-ul.
- Checker-ul testează prezența fișierului README (nu se acordă 10 puncte dacă acesta nu există) cât și coding style-ul prin intermediul script-ului **style.pl**; dacă aveți erori de coding style, se vor scădea 10 puncte din nota temei; erorile de coding style le veți putea vedea în fișierul **output/styleErrors.txt**.
- Scheletul temei vine cu fișierul header **file.h** în care se află definițiile structurilor de fișier și director care trebuie folosite în rezolvarea temei.

- Recomandarea noastră este să **izolați implementarea listei/listelor** de implementarea **operațiilor pe fișiere** (preferabil în fișiere diferite: de exemplu în list.h și list.c implementarea de listă, în file.h și file.c implementarea sistemului de fișiere și în main.c promptul interactiv) pentru a putea testa în primă fază funcționarea corectă a operațiilor pe listă, și de abia după să implementați operațiile pe fișiere.
- Puteți considera dimensiunea maximă a unei comenzi fixă (ex. 100) și să alocați static buffer-ul în care citiți de la stdin comanda.

[0]: [http://www.linfo.org/root\\_directory.html](http://www.linfo.org/root_directory.html)

[1]: [https://en.wikipedia.org/wiki/Computer\\_file](https://en.wikipedia.org/wiki/Computer_file)

[2]: [https://en.wikipedia.org/wiki/Directory\\_\(computing\)](https://en.wikipedia.org/wiki/Directory_(computing))

[3]: [https://en.wikipedia.org/wiki/File\\_system#Metadata](https://en.wikipedia.org/wiki/File_system#Metadata)

[4]: <https://www.geeksforgeeks.org/generic-linked-list-in-c-2/>

[5]: <https://aticleworld.com/generic-linked-list-in-c/>

[6]: <https://pseudomuto.com/2013/05/implementing-a-generic-linked-list-in-c/>

[7]: <https://vmchecker.cs.pub.ro/ui/#SDCC>