SD 2019 - 2020

Tema 1 : Filesystem Minishell

09.03.2020

Deadline hard: 29.03.2020

Responsabili temă: Ştefan Laurenţiu steflaurentiu@gmail.com
Niţu Ioan-Florin-Cătălin nitu.catalin1998@gmail.com

Ultima modificare: **08.03.2020**

0 Schimbări

11.03.2020 - adăugat precizare în legătură cu modificarea structurilor din schelet

1 Introducere

În ştiinţa 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 fişiere este una destul de clară, utilizatorii din ziua de azi fiind obişnuiți cu structura acestuia, precum şi să navigheze în acesta prin diferite modalităţi: prin GUI (ex. File Explorer pe Windows) sau prin CLI (ex. terminal pe Linux).

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

2 Structură

Sistemul de fişiere pe care dorim să-l implementăm este alcătuit din două tipuri de elemente: fişier **[1]** şi director **[2]**.

Un **fişier** este unitatea de bază a sistemului de fişiere şi are ca rol să reţină 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 fişiere are o structură arborescentă, care pleacă de la directorul rădăcină (root filesystem, '/').

Ne propunem să implementăm acest sistem de fişiere folosind listele înlănţuite. Pentru reprezentarea unui fişier, respectiv a unui director, se vor pune în schelet (în fişierul **file.h**) definiţiile structurilor aferente (în cadrul acestei teme, structurile de fişier, respectiv de director vor avea o reprezentare mai simplă, care conţine numai metadatele strict necesare pentru a implementa operaţiile de shell dorite, ce vor fi enunţate în continuare; de reţinut faptul că, într-un sistem de operare modern, structura de fişier conţine mult mai multe metadate: mai multe detalii la **[3]**).
Vom referi în continuare structurile de fişier şi director prin **File**, respectiv **Directory**.

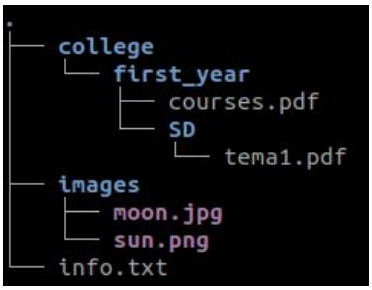
Structura Directory va avea în componenţă **o referinţă la o listă înlănţuită de fişiere** şi **o referinţă la o listă înlănţuită de directoare;**. Semnificaţia acestor liste este următoarea:

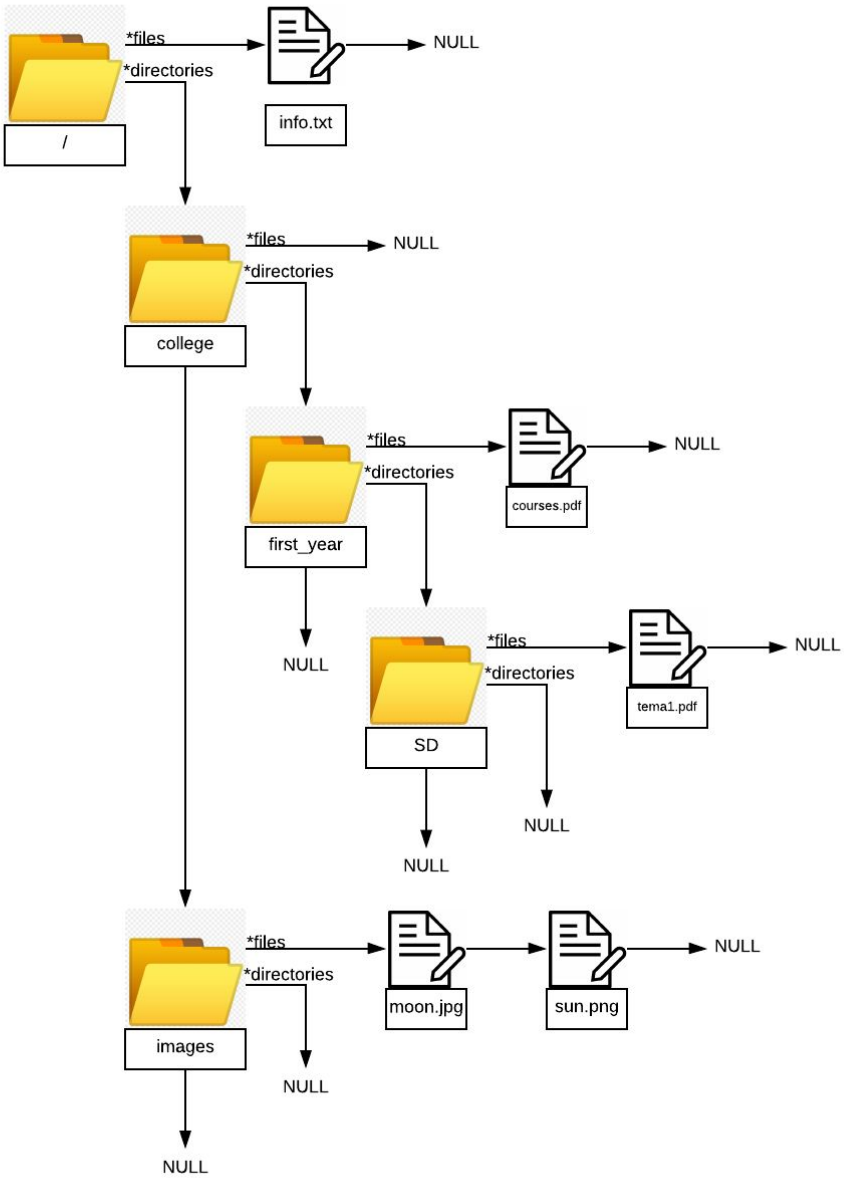
- Lista înlănţuită de fişiere conţine numai elemente de tipul **File**, acestea reprezentând fişierele accesibile din directorul curent
- Lista înlănţuită de directoare conţine numai elemente de tipul **Directory**, acestea reprezentând directoarele accesibile din directorul curent

Observaţii:

- **Întrucât în reprezentarea precizată mai sus avem două tipuri de liste (cu elemente de tipul File şi 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.). Totuşi, nu o să fie restricţii de implementări cât timp se folosesc listele înlănţuite**
- Dacă într-un director anume nu se află niciun fişier, lista de fişiere va fi vidă
- Dacă într-un director anume nu se află niciun alt director, lista de directoare va fi vidă

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


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


*files
*directories
NULL
/
info.txt
college
first_year
courses.pdf
SD
tema1.pdf
images
moon.jpg
sun.png

3 Cerinţe

Ne propunem să implementăm o aplicaţie de CLI care să gestioneze sistemul de fişiere. Pentru aceasta, ne dorim să implementăm o serie de operaţii asemănătoare cu CLI-ul din Linux, de manipulare a sistemului de fişiere, însă într-o formă mai simplificată, pentru a reduce complexitatea implementării. Operaţiile sunt descrise în subsecţiunile următoare.

Observaţie: Operaţiile de mai jos se realizează interactiv, la orice moment de timp existând un director curent (“unde ne aflăm”). La început, se va apela operaţia “create fs”, care are în acelaşi timp rolul de a seta directorul curent la rădăcina “/”.

3.1 Operaţii de iniţializare şi ștergere ale sistemului de fişiere

$ create fs

- creează rădăcina sistemului de fişiere (directorul cu numele “/”)

$ delete fs

- şterge DOAR directorul rădăcină al sistemului de fişiere (nu şterge recursiv conţinutul subdirectoarelor acestuia sau fişierele acestuia; considerăm că acestea au fost deja şterse prin operaţii rmdir şi rm realizate anterior)

3.2 Operaţie de creare a unui fişier

$ touch <filename> <content>

- adaugă un fişier nou cu numele <filename> şi conţinutul <content> în directorul curent (trebuie completată şi lungimea fişierului în structura File)
- pentru simplitate, content va fi un şir de caractere care nu conţine spaţii

Ex. touch test.in thisisthecontent

- Se creează în directorul curent un fişier cu numele “test.in” şi cu conţinutul “thisisthecontent”

3.3 Operaţie 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 Operaţie de listare a conţinutului unui director

$ ls

- afişează în ordine lexicografică numele fişierelor ce se află în directorul curent, urmate de numele directoarelor ce se află în directorul curent

Atenţie! Pentru o afişare lexicografică a numelor fişierelor şi a directoarelor, inserările în lista de fişiere 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 operaţie de tip "ls".

Această funcţie este folosită pentru a valida implementarea funcţiilor de creare/ştergere a fişierelor/directoarelor. Fără implementarea acesteia nu se vor putea testa implementările respective.

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

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

3.5 Operaţie de afişare a căii directorului curent

$ pwd

- afişează 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 Operaţie 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 afişa mesajul de eroare "Cannot move to '<directoryname>': No such directory!"
- <directoryname> poate fi şi ".." cu aceeaşi semnificaţie 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 fişiere din schema de mai sus

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

3.7 Operaţie de afişare a conţinutului sistemului de fişiere

$ tree

- afişează, începând de la directorul curent, sub o formă arborescentă, conţinutul subsistemului de fişiere astfel: se parcurg mai întâi fişierele din director şi se afişează numele acestora, după care, pentru fiecare director, se afişează numele acestuia şi se reia procedeul (vezi utilitarul tree pe Linux)
- la fel ca la operaţia ls, numele fişierelor şi directoarelor trebuie afişate în ordine lexicografică după nume
- se va folosi un nivel de indentare de 4 spaţii la fiecare nivel de adâncime în sistemul de fişiere

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

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

3.8 Operaţie de ştergere a unui fişier

$ rm <filename>

- şterge fişierul cu numele <filename> din directorul curent
- dacă în directorul curent nu există fişierul cu numele <filename>, se va afişa mesajul de eroare “Cannot remove ‘<filename>’: No such file!”

Ex. ne aflăm în rădăcina sistemului de fişiere 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 Operaţie de ştergere a unui director

$ rmdir <directoryname>

- şterge directorul cu numele <directoryname> din directorul curent (dacă acesta există) şi tot conţinutul acestuia într-o manieră recursivă (trebuie eliberat tot subsistemul de fişiere care pleacă de la directorul <directoryname>)
- dacă în directorul curent nu există directorul cu numele <directoryname>, se va afişa mesajul de eroare “Cannot remove ‘<directoryname>’: No such directory!”

Ex. ne aflăm în rădăcina sistemului de fişiere 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 Operaţie de căutare a unui fişier [BONUS]

$ find <max_depth> <min_size> <max_size> <subcontent>

- caută, începând de la directorul curent, cu o adâncime maximă de <max_depth>, fişierele care au o dimensiune în intervalul [min_size, max_size] şi care conţin textul <subcontent> (este inclus în conţinutul fişierului), iar apoi afişează numele fişierelor găsite.

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

- căutarea se face urmând o parcurgere în adâncime (nodul iniţial 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
```

Observaţii:

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

4 Format input/output

Aplicaţia rezultată se va comporta ca un prompt interactiv, ce aşteaptă tot timpul comenzi de la **stdin** şi afişează rezultatul acestora (dacă este cazul) la **stdout**. Se asigură că prima operaţie va fi întotdeauna “create fs” iar ultima operaţie va fi “delete fs”, care are şi rolul de a opri execuţia aplicaţiei. Operaţiile vor fi delimitate printr-un newline. De asemenea, după fiecare output al unei operaţii trebuie afişat un newline.

Mai jos se găseşte un exemplu integral de comenzi, care creează structura prezentată în schema de mai sus şi 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

Cerinţă	Punctaj	Observaţii
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 fişiere (2 teste de 5 puncte) 10p lipsă leak-uri de memorie (1 test rulat cu utilitarul **valgrind**)
3.9 rmdir		
README şi Coding Style	10p	Sunt verificate parţial de checker (se vor verifica şi 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 ı̂ntr-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** şi **clean**
- **Regula build** trebuie să aibă ca efect compilarea surselor şi crearea binarului cu numele **fsminishell**. Fișierele sursă trebuie să fie în rădăcina arhivei. Mai exact, nu faceţi un director în care se află toate sursele şi arhivaţi directorul, ci selectaţi toate sursele şi faceţi arhivă din ele: în directorul cu sursele daţi comanda
  ```
  zip NumeArhiva.zip sursa1 sursa2 ... sursan Makefile README
  ```

Arhiva trebuie să aibă tipul **zip** şi trebuie să fie numită astfel:
Nume_ToatePrenumele_GrupaSeria_Tema1SD.zip

De exemplu, Ioniţă 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 sancţionată conform regulamentului.
- Este **obligatorie** implementarea folosind **liste înlănţuite**! O implementare care **nu respectă** această restricţie **nu va fi punctată**.
- Citiţi cu atenţie observaţiile din cadrul enunţului. Pentru anumite nerespectări ale cerinţei se acordă depunctări.
- Se asigură că numele fişierelor şi al directoarelor va fi unic în cadrul aceluiaşi director.
- În arhiva checker-ului se află şi fişierul “**instructions.txt**” în care aveţi mai multe detalii despre cum să rulaţi checker-ul.
- Checker-ul testează prezenţa fişierului README (nu se acordă 10 puncte dacă acesta nu există) cât şi coding style-ul prin intermediul script-ului **style.pl**; dacă aveţi erori de coding style, se vor scădea 10 puncte din nota temei; erorile de coding style le veţi putea vedea în fişierul **output/styleErrors.txt**.
- Scheletul temei vine cu fişierul header **file.h** în care se află definiţiile structurilor de fişier şi director care trebuie folosite în rezolvarea temei.

- **Este interzisă modificarea structurii File, iar în structura Directory este permisă doar adăugarea celor 2 pointeri spre liste (marcat cu TODO în schelet)**
- Recomandarea noastră este să **izolaţi implementarea listei/listelor** de implementarea **operaţiilor pe fişiere** (preferabil în fişiere diferite: de exemplu în list.h şi list.c implementarea de listă, în file.h şi file.c implementarea sistemului de fişiere şi în main.c promptul interactiv) pentru a putea testa în primă fază funcţionarea corectă a operaţiilor pe listă, şi de abia după să implementaţi operaţiile pe fişiere.
- Puteţi considera dimensiunea maximă a unei comenzi fixă (ex. 100) şi să alocaţi static buffer-ul în care citiţi de la stdin comanda.

[0]: http://www.linfo.org/root_directory.html
[1]: https://en.wikipedia.org/wiki/Computer_file
[2]: https://en.wikipedia.org/wiki/Directory_(computing)
[3]: 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