# Tema 3 - Sistem de Fișiere

## Structuri de Date si Algoritmi (Seria CB)

| | |
|---|---|
| **Responsabili Temă** | Butnariu Bogdan, Cîrpici Sorin |
| **Data publicării** | 9.05.2022 |
| **Termen predare** | Deadline 27.05.2022 (ora 23:59)<br>Se acceptă teme trimise cu penalizare de 10 puncte/zi (din maxim 150 puncte) până la data de 29.05.2022 (ora 23:59) |
| **Versiune document** | 1.0 |
| **Versiune checker** | 1.0 |

## 1. Introducere

După aproape un semestru de PCLP2 si SDA unde am învățat despre limbajul C, dorim să simulăm un sistem de fișiere bazat pe Arbori Binari de Căutare. Programul va rula în terminal si va folosi câteva comenzi învățate la USO.

Pentru simplificare, vom avea în vedere doar legăturile dintre directoare si fișiere, precum și ierarhia lor.

Fiecare director va avea următoarea structură:

- nume (șir de caractere)
- părinte (pointer către directorul părinte)
- fișier (pointer către rădăcina arborelui de fișiere)
- directories (pointer către rădăcina arborelui de subdirectoare)
- st (pointer către următorul director cu nume mai mic lexicografic decât el)
- dr (pointer către următorul director cu nume mai mare lexicografic decât el)

Fiecare fișier va avea următoarea structură:

- nume (șir de caractere)
- părinte (pointer către directorul de care aparține)
- st (pointer către următorul fișier cu nume mai mic lexicografic decât el)
- dr (pointer către următorul fișier cu nume mai mare lexicografic decât el)

## 2. Implementare

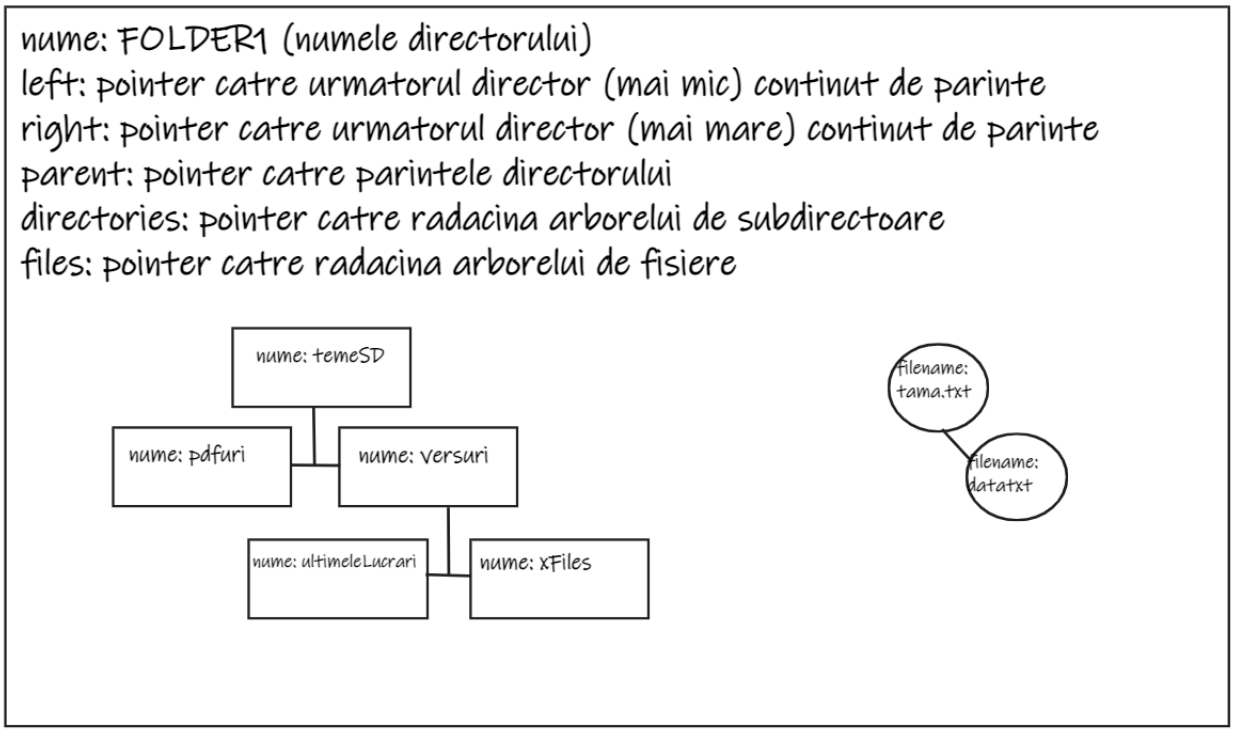
Pentru gestionarea sistemului de fișiere, vom folosi următoarele structuri de date:

1. **Director root**: un director inițializat înainte de efectuarea comenzilor.
   name = „root”; files = NULL;
   parent = NULL; left = NULL;
   directories = NULL; right = NULL;

2. **Arbori Binari de Căutare:** fișierele si subdirectoarele unui directoare vor fi stocate fiecare în câte un Arbore Binar de Căutare. Câmpurile directorului root, „directories” si „files”, vor reprezenta rădăcinile Arborelui corespunzător.

Exemplu:

- Un director (FOLDER1) ce conține 5 subdirectoare (temeSD, pdfuri, versuri, ultimeleLucrari, xFiles) și 2 fișiere (tema.txt, date.txt)




- Atenție! **Urmăriți cu grijă** diagrama de mai sus înainte să vă apucați de cerințe.

# 3. Cerință

## 3.1. Adăugare fișier 12p

**Sintaxa:** touch <nume_fișier>

**Mod de funcționare:** Se verifică câmpul „files” al directorului curent. Dacă acest câmp este NULL, se inițializează Arborele Binar de Căutare corespunzător cu rădăcina <nume_fișier>. În caz contrar, se compară numele rădăcinii cu <nume_fișier>. Se parcurge în funcție de comparare pe câmpurile „left” sau „right”. Se repetă algoritmul. Dacă există un fișier sau un director cu același nume, se va afișa mesajul „File <nume_fișier> already exists!” sau „Directory <nume_fișier> already exists!”.

## 3.2. Adăugare subdirector 12p

**Sintaxa:** mkdir <nume_director>

**Mod de funcționare:** Se verifică câmpul „directories” al directorului curent. Dacă acest câmp este NULL, se inițializează Arborele Binar de Căutare corespunzător cu rădăcina <nume_director>. În caz contrar, se compară numele rădăcinii cu <nume_director>. Se parcurge în funcție de comparare pe câmpurile „left” sau „right”. Se repetă algoritmul. Dacă există un fișier sau un director cu același nume, se va afișa mesajul „File <nume_director> already exists!” sau „Directory <nume_director> already exists!”.

## 3.3. Afișare conținut 17p

**Sintaxa:** ls

**Mod de funcționare:** Se parcurge câmpul „directories” al directorului curent si se printează în ordine lexicografică numele directoarelor, fiecare nume este urmat de un spațiu. Se parcurge câmpul „files” al directorului curent si se printează în ordine lexicografică numele fișierelor, fiecare nume este urmat de un spațiu. În cazul în care nu avem directoare sau fișiere, programul afișează o linie goală.

### 3.4. Ștergere fișier 17p

**Sintaxa:** rm <nume_fișier>

**Mod de funcționare:** Se parcurge câmpul „files” si se șterge fișierul <nume_fișier>. Se va avea în vedere păstrarea Arborelui Binar de Căutare. În cazul în care nu există un fișier cu numele <nume_fișier> se va afișa mesajul „File <nume_fișier> doesn't exist!”.

### 3.5. Ștergere director 17p

**Sintaxa:** rmdir <nume_director>

**Mod de funcționare:** Se parcurge câmpul „directories” si se sterge directorul <nume_director>. Se va avea în vedere păstrarea Arborelui Binar de Căutare. Ștergerea directorului va avea în vedere si ștergerea fișierelor si subdirectoarelor. În cazul în care nu există un director cu numele <nume_director> se va afișa mesajul „Directory <nume_director> doesn't exist!”.

### 3.6. Schimbare director curent 20p

**Sintaxa:** cd <nume_director>

**Mod de funcționare:** Se parcurge câmpul „directories” si se selectează directorul cu numele potrivit. Dacă directorul nu este găsit, se va afișa mesajul „Directory not found!”. În cazul în care <nume_director> este „..”, ne vom deplasa pe directorul părinte.

### 3.7 Afișare cale de lucru 20p

**Sintaxa:** pwd

**Mod de funcționare:** Se parcurg părinții directoarelor si se afiseaza numele directorului până in momentul în care ajungem în directorul „root”.

### 3.7. Găsire director, fișier 20p

**Sintaxa:** find -d/-f <nume>

**Mod de funcționare:** Se găsește directorul/fișierul si se afișează mesajul „Directory/File <nume> found!” urmat de apelul comenzii **pwd**. În cazul în care directorul/fișierul nu există, se afișează mesajul „Directory/File <nume> not found!”.

## 4. Restricții si precizări

- Numele unui fișier sau a unui director este un șir de maxim 50 de caractere.
- Nu există directoare sau fișiere cu același nume.
- Programul va fi rulat astfel: „./tema3”. Comenzile vor fi date în terminal

## 5. Exemplu

| Intrare | Ieșire |
|---|---|
| mkdir d1<br>mkdir d2<br>touch f1<br>ls<br>touch d1<br>rm d3<br>rm d1<br>rm f1<br>ls<br>cd d2<br>mkdir d3<br>cd d3<br>touch file<br>cd ..<br>cd ..<br>pwd<br>ls<br>find -f file<br>quit | *//s-a creat directorul d1 in directorul root*<br>*//s-a creat directorul d2 in directorul root*<br>*//s-a creat fisierul f1 in directorul root*<br>d1 d2 f1 *//se afiseaza (**in order**) intai numele din arborele de directoare si apoi (**in Order**) numele din arborele de fisiere*<br>Director d1 already exists! *//nu se mai creeaza d1 caci el exisat deja IN root*<br>File d3 doesn't exist! *//directorul d3 care trebuie sters nu exista (exista doar d1 si d2)*<br>File d1 doesn't exist! *//fisierul d1 care trebuie sters nu exista, exista directorul d1*<br>d1 d2 *//acum exista doar derectoarele d1 si d2 caci fisierul f1 a fost sters in urma comenzii „rm f1”*<br>/root<br>d1 d2<br>/root/d2/d3 |

## 6. Utilizare checker

- Comandă pentru drept de execuție: chmod +x checker.sh
- Verificare temă: ./checker.sh
- Verificare test: ./checker.sh <numărul testului>

## 7. Restricții și precizări

- **135 de puncte** obținute pe testele de pe vmchecker. **Observatie:** Se pot obtinue punctaje parțiale fără a implementa toate comenzile.
    - Testele 1-10 folosesc doar comenzile
        - ✓ *touch <nume_fișier>*
        - ✓ *mkdir <nume_director>*
        - ✓ *ls*
    - Testele 11-14 folosesc în plus comanda
        - ✓ *rm <nume_fișier>*
    - Testele 15-18 folosesc in plus comanda
        - ✓ *rmdir <nume_director>*
    - Testele 19-22 folosesc in plus comanda
        - ✓ *cd <nume_director>*
    - Testele 23-26 folosesc in plus comanda
        - ✓ *pwd*
    - Testele 27-30 folosesc in plus comanda
        - ✓ *find -d/-f <nume_director>*
- **10 de puncte** coding style.
- **5 puncte** README - va conține detaliile de implementare a temei, precum și punctajul obținut la teste (la rularea pe calculatorul propriu).
- **20 puncte bonus** pentru soluțiile care nu au memory leak-uri (bonusul se acordă doar dacă testul a trecut cu succes).
- temele care nu compilează, nu rulează sau obțin punctaj 0 pe vmchecker, vor primi punctaj 0.
- se depunctează pentru:
    - warninguri la compilare (trebuie compilat cu -Wall)
    - linii mai lungi de 80 de caractere
    - folosirea incorectă de pointeri
    - **altă implementare** față de Arbori Binari de Căutare

## 8. Reguli de trimitere a temelor.

- temele vor trebui încărcate atât pe vmchecker (în **secțiunea Structuri de Date Seria CB**) cât și pe curs.upb.ro, în secțiunea aferentă temei 3.
- arhiva cu rezolvarea temei trebuie să fie .zip și să conțină:
    - fișiere sursă (fiecare fișier sursă va trebui să înceapă cu un comentariu de forma: /* NUME Prenume - grupa */),
    - fișier **README**, denumit obligatoriu astfel, care să conțină detalii despre implementarea temei,
    - fișier **Makefile**, denumit obligatoriu astfel, cu două reguli:
        - **build**, care va compila sursele și va obține executabilul cu numele **tema3**
        - **clean**, care va șterge executabilele și alte fișier obiect generate
- arhiva trebuie să conțină doar fișierele sursă (inclusiv Makefile și README), nu se acceptă fișiere executabile sau obiect.
- dacă arhiva nu respectă specificațiile de mai sus nu va fi acceptată la upload și tema nu va fi luată în considerare.

## 9. Reguli împotriva copierii temelor

- se consideră copiate doua teme care seamănă suficient de mult pentru a putea trage aceasta concluzie;
- modificarea unei alte teme, asemănarea mai mult sau mai puțin evidentă a implementării, bucăți de cod identice etc. duc la considerarea temelor în cauză ca fiind copiate;
- pentru prima temă copiată: atât sursei cât și destinației li se va anula punctajul pentru tema respectivă şi ambii studenţi vor primi mustrare scrisă, fără discuții relative la cine a copiat de la cine și a cui e vina;
- la a doua temă copiată: atât sursei cât și destinației li se va anula punctajul pentru toate temele (ceea ce va duce la repetarea materiei) şi ambii studenţi vor primi mustrare scrisă, fără discuții relative la cine a copiat de la cine și a cui e vina.