

Tema 1 - Sistemul lui Biju

- Deadline: 21.11.2021
- Data publicării: 01.11.2021
- Ultima actualizare a checker-ului
- Responsabili:
 - Tudor Hodoboc-Velescu [mailto:tudorhodoboc@yahoo.com]
 - Catalin-Gabriel Popa [mailto:catalingabrielpopa847@gmail.com]

Enunt

Biju si-a stricat linuxul dand comanda `sudo deluser username` sudo recomandata de prieteni. Asa ca si-a facut ca scop sa le strice tuturor sistemul de operare, dar ca sa faca asta trebuie sa inteleaga ce este un sistem de operare. Deoarece un sistem de operare este exagerat de complex, si se face in anul 3 *wenk* *wenk*, doreste sa faca un sistem mult mai simplu. Ajutati-l pe Biju sa se razbune facandu-i urmatoarea simulare.

Deoarece chiar si un model simplu de sistem de operare este foarte complex, dorim sa simulam doar sistemul de fisiere. Orice sistem de operare lucreaza cu fisiere si directoare, asa ca ne propunem sa simulam operatii simple pe aceste doua structuri, ca de exemplu: touch, mkdir, rm, rmdir..

Implementare

Ne propunem să implementăm acest sistem de fisiere folosind liste simple înlănțuite. In 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.

Urmatoarele doua reprezinta implementarea structurilor de date:

```
typedef struct Dir{
    char *name; // numele directorului
    struct Dir* parent; // pointer catre parintele directorului(null pentru radacina)
    struct File* head_children_files; // pointer catre primul element de tip File din interiorul directorului
    struct Dir* head_children_dirs; // pointer catre primul element de tip Dir din interiorul directorului
    struct Dir* next; // pointer catre urmatorul element din lista in care se afla directorul
}Dir; // structura de tip director

typedef struct File {
    char *name; // numele fisierului
    struct Dir* parent; // pointer catre directorul pe
    struct File* next; // pointer catre urmatorul element din lista de fisiere
}File; // structura de tip fisier
```

Ne dorim ca cel mai de baza nivel al sistemului de fisiere sa fie un director numit home in interiorul caruia o sa ne desfasuram toate task-urile.

In momentul rularii aplicatiei, aceasta va astepta inputuri de la stdin. Fiecare comanda va fi apelata sub forma <nume_comanda> + argumente, argumentele variind in functie de comanda data. Exemple de comenzi: touch f1, mkdir d1, rm f2, rmdir d2, ls, mv nume_vechi nume_nou etc. Aplicatia se va opri la comanda stop. Comenzile vor fi explicate in cele ce urmeaza.

Recomandam citirea punctului 10. *memory management* inainte de a va apuca de scris cod.

1. touch

```
void touch (Dir* parent, char* name);
```

Functia creaza un fisier in directorul curent. Acesta va fi adaugat la finalul listei de fisiere. Aceasta primeste ca argument numele fisierului. Numele fisierului nu contine "/"(nu va reprezenta o cale). In cazul in care exista deja un fisier/director cu acest nume, se va afisa urmatorul mesaj de eroare: File already exists\n.

2. mkdir

```
void mkdir (Dir* parent, char* name);
```

Functia creaza un director in directorul curent. Acesta va fi adaugat la finalul listei de fisiere. Aceasta primeste ca argument numele directorului. Numele directorului nu contine "/"(nu va reprezenta o cale). In cazul in care exista deja un fisier/director cu acest nume, se va afisa urmatorul mesaj de eroare: Directory already exists\n.

3. ls - 30p

```
void ls (Dir* parent);
```

Functia afiseaza toate directoarele, respectiv fisierele din directorul curent. Mai intai vor fi afisate numele directoarelor, apoi a fisierele, fiecare pe un rand nou.

Deoarece primele doua functii nu pot fi testate fara ls, acestea vor fi testate impreuna.

4. rm - 5p

```
void rm (Dir* parent, char* name);
```

Functia elimina fisierul cu numele name din directorul curent. Daca nu a fost gasit niciun fisier cu numele dat, se va afisa mesajul de eroare Could not find the file\n. Numele fisierului nu contine "/"(nu va reprezenta o cale).

Exemplu de rulare:

```
touch f1
touch f2
```

```
ls
rm f1
ls
stop
```

Rezultat:

```
f1
f2
f2
```

5. rmdir - 10p

```
void rmdir (Dir* parent, char* name);
```

Funcția elimină directorul cu numele `name` din directorul curent și toate fișierele care se găsesc în acesta. Dacă nu a fost găsit niciun director cu numele dat, se va afișa mesajul de eroare `Could not find the dir\n`. Numele directorului nu conține `"/` (nu va reprezenta o cale).

Exemplu de rulare:

```
mkdir d1
mkdir d2
ls
rmdir d2
ls
stop
```

Rezultat:

```
d1
d2
d1
```

Comportamentul funcției `rmdir` diferă de comportamentul utilitarului `rmdir`. Funcționalitatea acestei funcții este echivalentă cu folosirea comenzii `rm -r <director>`

6. cd - 5p

```
void cd(Dir** target, char* name);
```

Funcția schimbă directorul curent în directorul cu numele `name`, din lista de directoare a directorului curent. Dacă nu este găsit, se va afișa mesajul de eroare `No directories found!\n`. Pentru a schimba directorul curent în directorul părinte al celui curent se va folosi sirul de caractere special `..`. Dacă directorul curent nu are părinte, atunci nu se va schimba directorul curent. Numele directorului nu conține `"/` (nu va reprezenta o cale).

7. tree - 10p

```
void tree(Dir* target, int level);
```

Funcția `tree` afișează interiorul directorului curent într-o formă arborescentă și al fiecărui director din interior.

Formatul de output este următorul: 4 spații(" ") x `level` + nume fișier/director. Antetul funcției este scris pentru a permite o rezolvare recursivă, însă cerința poate fi rezolvată iterativ.

Ordinea de afișare este următoarea:

- Se vor afișa mai întâi numele directoarele și apoi numele fișierele
- Numele directoarelor și al fișierelor vor fi afișate în ordinea în care au fost create
- După afișarea numelui unui director, se va afișa tot ceea ce conține acel director

Exemplu de rulare:

```
mkdir d1
mkdir d2
cd d1
touch f1
mkdir d3
cd d3
touch f2
mkdir d4
cd ..
cd ..
tree
stop
```

Rezultat:

```
d1
  d3
    d4
    f2
  f1
d2
```

8. pwd - 10p

```
char *pwd (Dir* target);
```

Funcția întoarce un sir de caractere ce reprezintă calea de la directorul rădăcină(home) până la directorul curent, cu numele `target`. Formatul aplicat pentru afișare este: `"/home/(parent_name)/..../(target-name)"`. Rezultatul funcției trebuie afișat la `stdout`, în afara funcției `pwd`.

Exemplu de rulare:

```
mkdir d1
cd d1
pwd
stop
```

Rezultat:

```
/home/d1
```

9. stop

```
void stop (Dir* target);
```

Aceasta functie opreste aplicatia.

10. memory management - 20p

O alta parte foarte importanta a temei este intelegerea si lucrul cu memoria. Pentru asta, acest task va consta din doua parti:

1) Alocarea corecta de memorie: memoria va fi alocata dinamic pentru troate string-urile si structurile folosite. Pentru un fisier cu numele "Ana", se vor alocata dinamic 4 octeti (3 pentru fiecare litera + unul pentru terminatorul de sir). Declaratiile statice de tipul `char a[50]` trebuie modificate in `char *a = malloc(size_of_string);`.

2) Dezalocarea corecta a memoriei: memoria va fi dezalocata corect si complet. Astfel, in functie de task, vor trebui sa se faca urmatoarele operatii:

- `rm`: se va dezaloca structura in sine
- `rmdir`: se va dezaloca atat structura in sine, cat si fiecare structura din interiorul directorului (atat directoarele cat si fisierele)
- `pwd`: se va dezaloca sirul intors de acesta, in afara functiei `pwd` (dupa ce a fost afisat)
- `stop`: la inchiderea aplicatiei, memoria alocata pentru directoarele si fisierele va fi eliberata

Pentru testare, vom folosi `valgrind` si ca punct de referinta, programul nu trebuie sa afiseze niciun `read invalid` sau orice `leak` de memorie.

`valgrind -leak-check=full -show-leak-kinds=all -track-origins=yes ./tema.`

Punctajul pe aceasta cerinta va fi oferit doar daca cel putin 50p au fost obtinute din alte cerinte.

11. (Bonus) mv - 20p

```
void mv(Dir* parent, char* oldname, char* newname);
```

Functia va schimba numele directorului/fisierului `oldname` din directorul curent in `newname`. Se va verifica, mai intai, daca numele `oldname` exista deja, in caz contrar se va afisa mesajul `File/Director not found`, iar apoi daca numele `newname` nu este folosit deja, in caz contrar se va afisa mesajul de eroare `File/Director already exists`. Schimbarea se va face prin readaugare directorului/fisierului in lista de directoare/fisiere. In cazul directoarelor, interiorul va ramane neschimbat. Numele directorului/fisierului (atat cel vechi, cat si cel nou) nu contine "/" (nu va reprezenta o cale).

Exemplu de rulare 1:

```
mkdir d1
ls
mv d2 d3
ls
stop
```

Rezultat 1:

```
d1
File/Director not found
d1
```

Exemplu de rulare 2:

```
touch f1
touch f2
ls
mv f1 f3
ls
stop
```

Rezultat 2:

```
f1
f2
f2
f3
```

Trimitere și notare

Temele vor trebui încărcate pe platforma `vmchecker` [<https://vmchecker.cs.pub.ro/ui/#IOCLA>] (în secțiunea `IOCLA`) și vor fi testate automat. Arhiva încărcată trebuie să fie o arhivă `.zip` care să conțină:

- fișierul sursă ce conține implementarea temei, denumit `tema1.c`
- fișier `README` ce conține descrierea implementării

Punctajul final acordat pe o temă este compus din:

- punctajul obținut prin testarea automată de pe vmchecker - 90%
- fișier README + coding style - 10%
- bonus - 20%

Precizări suplimentare

- Checkerul se va rula folosind comanda `make checker` după ce ai dat drepturi de execuție fișierului.
- Fișierul sursa se va numi `tema1.c`.
- Arhiva va trebui încărcată pe vmchecker. Pe lângă codul sursa, va trebui să fie adăugat și un fișier README. Acesta va reprezenta 10p din nota finală.
- Nu se acordă puncte pentru coding style, dar încurajăm scrierea unui cod lizibil și îngrijit, și un cod cât mai modularizat.
- Dimensiunea maximă a unei linii citite este definită în scheletul de cod cu numele `MAX_INPUT_LINE_SIZE`
- Recomandăm parcurgerea regulamentului [https://ocw.cs.pub.ro/courses/iocla/reguli-notare#realizarea_temelor] dacă nu ai făcut-o deja

Resurse

- Scheletul de cod, Makefile-ul și scriptul de testare se găsesc pe repository-ul public IOCLA [<https://github.com/systems-cs-pub-ro/iocla>] în folderul `teme/tema-1/`.

iocla/teme/tema-1.txt · Last modified: 2021/11/01 22:50 by tudor.hodoboc