This is an old revision of the document!

Tema 1 - Sistemul lui Biju

- Deadline: 21.11.2021
- Data publicării: 01.11.2021
- Ultima actualizare a enuntului:
  - 02.11.2021: Update coding style [https://ocw.cs.pub.ro/courses/iocla/teme/tema-1#precizari_suplimentare]
- 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 fişiere folosind liste simple înlănţuite. In 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.

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 fisierelor, 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);
```

Functia elimina directorul cu numele `name` din directorul curent si toate fisierele care se gasesc in acesta. Daca nu a fost gasit niciun director cu numele dat, se va afisa mesajul de eroare `Could not find the dir\n`. Numele directorului nu contine "/"(nu va reprezenta o cale).

Exemplu de rulare:

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

Rezultat:

```
d1
d2
d1
```

Comportamentul functiei `rmdir` difera de comportamentul utilitarului `rmdir`. Functionalitatea acestei functii este echivalenta cu folosirea comenzii `rm -r <director>`

6. cd - 5p

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

Functia schimba directorul curent in directorul cu numele name, din lista de directoare a directorului curent. Daca nu este gasit, se va afisa mesajul de eroare `No directories found!\n`. Pentru a schimba directorul curent in directorul parinte al celui curent se va folosi sirul de caractere special "..". Daca directorul curent nu are parinte, ,atunci nu se va schimba directorul curent. Numele directorului nu contine "/"(nu va reprezenta o cale).

7. tree - 10p

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

Functia tree afiseaza interiorul directorului curent intr-o forma arborescenta si al fiecarului director din interior.

Formatul de output este urmatorul: 4 spatii(" ") x level + nume file/director. Antetul functiei estescris pentru a permite o rezolvare recursiva, insa cerinta poate fi rezolvata iterativ.

Ordinea de afisare este urmtoarea:

- Se vor afisa mai intai numele directoarele si apoi numele fisierele
- Numele directoarelor si al fisierelor vor fi afisate in ordinea in care au fost create
- Dupa afisarea numelui unui director, se va afisa tot ceea ce contine 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);
```

Functia intoarce un sir de caractere ce reprezinta calea de la directorul radacina(home) pana la directorul curent, cu numele target. Formatul aplicat pentru afisare este: "/home/(parent_name)/…./(target-name)". Rezultatul functiei trebuie afisat la stdout, in afara functiei 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 aloca 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\n`, iar apoi daca numele newname nu este folosit deja, in caz contrar se va afisa mesajul de eroare `File/Director already exists\n`. 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 - 10%
- bonus - 20%

Precizări suplimentare

- Checkerul se va rula folosind comanda `make checker` după ce ați dat drepturi de execuție fișierului.
- Fisierul sursa se va numi tema1.c.
- Arhiva va trebui incarcata pe vmchecker. Pe langa codul sursa, va trebui sa fie adaugat si un fisier README. Acesta va reprezenta 10p din nota finala.
- Dimensiunea maxima a unei linii citite este definita in scheletul de cod cu numele `MAX_INPUT_LINE_SIZE`
- Recomandam parcurgerea regulamentului [https://ocw.cs.pub.ro/courses/iocla/reguli-notare#realizarea_temelor] daca nu ati facut-o deja
- EDIT 2.11.2021: Puteti folosi orice coding style, cat timp sunteti consecventi. Vom depuncta situatii in care coding style-ul ingreuneaza citirea codului.

Resurse

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

iocla/teme/tema-1.1635870319.txt.gz · Last modified: 2021/11/02 18:25 by radu.nicolau