Enunt

Sa se implementeze un protocol simplu client - server folosind mecanisme IPC. Serverul intretine un arbore binar de cautare in care se fac inserari si stergeri comandate de mesajele primite de la clienti.

Clientii sunt comandati prin argumentele primite in linia de comanda la lansarea in executie:

```
./client a 1 a 2 r 1
```

va trimite serverului, in ordine, mesajele: adauga in arbore valoarea 1, adauga valoarea 2, elimina din arbore valoarea 1.

Lista completa de comenzi pe care le poate primi clientul este:

- "a n" : trimite serverului mesajul de adaugare in arbore a valorii n;
- "r n" : trimite serverului mesajul de eliminare din arbore a primului nod gasit ce contine valoarea n;
- "c" : trimite serverului mesajul de golire a continutului arborelui (clear);
- "s m" : clientul face sleep pentru m milisecunde;
- "p" : clientul afiseaza la standard output arborele (formatul este precizat mai jos);
- "e" : clientul ii spune serverului sa isi incheie executia.

Exemple:

```
./client a 3 a 1 a 4 r 1 r 3 s 1000 p
./client a 5 a 4 s 1000 e
```

Afisarea arborelui se va face pe niveluri, cate un nivel pe linie, pornind de la radacina. De exemplu pentru arborele:

se va afisa:

```
3
1 5
-1 * 4 6
* * * * 5 *
* *
```

unde * semnifica faptul ca nodul respectiv nu exista (fiul unui nod de pe nivelul superior, fiu ce trebuia afisat acum, este nul). Se observa ca nu se continua afisarea de * ca fii ai nodurilor *. Alte exemple:

arbore

output

```
4
1 5
* * * *
```

arbore

output

```
4
* 5
* 6
* *
```

arbore

```
    4
   /
  1
```

output

```
4
1 *
* *
```

arborele vid

output

```
*
```

Simbolurile de pe o linie (valori de chei sau *) sunt separate prin spatii.

Precizari Generale

- Valorile introduse in arbore sunt numere intregi.
- Subarborele stang al unui nod va contine valori < decat valoarea din nod, iar subarborele drept va contine valori >= cu valoarea din nod (sunt permise duplicatele).
- Pentru stergerea unui nod din arbore se va folosi varianta de algoritm care inlocuieste nodul sters cu **SUCCESORUL** sau, daca nodul are ambii fii.
- Pot exista mai multi clienti care comunica simultan cu serverul. Operatiile pe care le face serverul trebuie sa respecte ordinea temporala a trimiterii mesajelor clientilor.
- Clientii trebuie sa execute comenzile in ordine, asa cum au fost primite in linia de comanda.
- Pentru comunicatia client-server se vor folosi cozi de mesaje (MailSlots in Windows).

- Din motive de eficienta, este obligatoriu sa se foloseasca memorie partajata pentru operatia de print: serverul va mentine arborele in memoria partajata iar clientul va citi aceasta aceasta zona de memorie de fiecare data cand are nevoie sa faca print.
  - **În memoria partajată se menține arborele serializat, nu un șir de caractere reprezentând afișarea lui. Clientul va de-serializa arborele și va face, în acest timp, afișarea.**

- Pentru sincronizare se vor folosi numai semafoare, atat pe Linux cat si pe Windows.

- Dimensiunea memoriei partajate poate fi fixa (nu este nevoie sa fie extinsa). Se recomanda valoarea de 4kB.

Testare

Pentru simplificarea procesului de corectare al temelor, dar si pentru a reduce greselile temelor trimise, corectarea temelor se va face automat cu ajutorul unor teste publice (linux, windows).

Formatul afisarii, asa cum a fost descris mai sus, este obligatoriu.

In urma compilarii temei trebuie sa rezulte doua executabile: `server` (respectiv `server.exe` pentru Windows) si `client` (respectiv `client.exe` pentru Windows). Numele acestor executabile trebuie sa fie respectat.

Se vor acorda cate 1.25 puncte pentru fiecare dintre cele 8 teste. Nota mai poate fi modificata prin depunctari suplimentare:

-0.4 neeliberarea resurselor asociate cu noduri sterse din arbore

-0.6 sincronizare incorecta pe resursele partajate de mai multe procese

-0.1 pentru makefile incorect

-0.2 pentru README necorespunzator

-0.2 pentru surse prost comentate

-0.4 neverificarea conditiilor de eroare sau/si neeliberarea de resurse

-0.1 diverse alte probleme constatate in implementare

Precizari Windows

Tema se va rezolva folosind doar functii Win32. Se pot folosi de asemenea si functiile de formatare `printf`, `scanf`, functiile de alocare de memorie `malloc`, `free` si functiile de manipulare a sirurilor de caractere (`strcat`, `strdup`, etc.)

Pentru partea de I/O si procese se vor folosi doar functii Win32. De exemplu, functiile `open`, `read`, `write`, `close` nu trebuie folosite, in locul acestor trebuind sa folositi `CreateFile`,`ReadFile`, `WriteFile`, `CloseHandle`.

Precizari Linux

Procesul server trebuie sa fie implementat ca daemon.

Tema se va rezolva folosind doar functii POSIX. Se pot folosi de asemenea si functiile de formatare `printf`, `scanf`, functiile de alocare de memorie `malloc`, `free` si functiile de manipulare a sirurilor de caractere (`strcat`, `strdup`, etc.)

Pentru partea de I/O si procese se vor folosi doar functii POSIX. De exemplu, functiile `fopen`, `fread`, `fwrite`, `fclose` nu trebuie folosite, in locul acestor trebuind sa folositi `open`,`read`, `write`, `close`.