Test practic la SO – varianta nr. 3

Realizați o aplicație formată din următoarele trei programe cooperante, plus un script de execuție a lor și un fișier cu date de intrare în format text, care vor fi plasate conform ierarhiei de directoare de mai jos:

```
.
├── app
│   ├── analyzers
│   │   └── analyzer.c
│   └── services
│       ├── service.c
│       └── worker.c
├── data
│   └── input.txt
└── starter.sh
```

1. Scriptul "starter.sh" va implementa funcționalitatea descrisă în specificația următoare:

i) Scriptul va primi în linia de comandă un parametru *p*, având valoarea 0 sau 1, iar în caz contrar va afișa un mesaj de eroare adecvat și se va termina.
ii) Dacă *p*=0, atunci scriptul va porni mai întâi execuția programului **analyzer** (din subdirectorul corespunzător) în *background*, iar după o pauză de 1 secundă, va porni și execuția programului **service** (din subdirectorul corespunzător).
iii) Iar dacă *p*=1, atunci scriptul va porni mai întâi execuția programului **service** (din subdirectorul corespunzător) în *background*, iar după o pauză de 2 secunde, va porni și execuția programului **analyzer** (din subdirectorul corespunzător).
iv) În ambele situații, programul **analyzer** va fi pornit cu un argument în linia de comandă: calea (absolută sau relativă) către fișierul de intrare "input.txt" (pe care-l veți crea anterior, în orice editor de texte doriți, cu formatul specificat mai jos, în enunțul programului **analyzer**).
v) Apoi scriptul va aștepta terminarea execuției celor trei programe și va afișa conținutul mapării nepersistente cu nume, după care va face *curățenie*: va "distruge" maparea nepersistentă cu nume și canalul fifo.

2. Programul "analyzer.c" va implementa funcționalitatea descrisă în specificația următoare:

Programul va primi un argument în linia de comandă, ce reprezintă calea (absolută sau relativă) către un fișier existent pe disc, numit "input.txt". Acest fișier conține diverse caractere ASCII, fără un anumit format.
i) În funcția main, programul va verifica dacă s-a primit un argument, numele fișierului de intrare, și dacă are drepturi de citire asupra sa. Și tot în funcția main se vor face inițializările necesare pentru a putea schimba informații cu procesul **service**, în ambele sensuri, printr-un singur obiect de memorie partajată – o mapare nepersistentă cu nume, creată cu funcția *shm_open*.
ii) Într-o funcție separată, apelată din funcția main, programul va citi fiecare cuvânt (prin *cuvânt* înțelegem o secvență de caractere consecutive formată doar din litere, mari sau/și mici; e.g: pentru textul "abc 1 DefG" avem 2 cuvinte "abc", respectiv "DefG") din fișierul input.txt și va transmite cuvintele astfel extrase către procesul **service** prin intermediul acelei mapări nepersistente cu nume. (*Notă*: așadar, separatorul între două cuvinte din fișier este format din unul sau mai multe caractere care nu sunt litere mari sau mici.)
iii) Într-o altă funcție separată, apelată din funcția main, programul va citi rezultatele (cele două trigrame și un număr) transmise acestuia de către procesul **service** prin intermediul acelei mapări nepersistente cu nume și le va afișa pe ecran.

3. Programul "service.c" va implementa funcționalitatea descrisă în specificația următoare:

i) În funcția principală a programului, acesta va crea un proces fiu, iar în fiul creat va starta, printr-un apel exec adecvat, programul **worker**.
ii) De asemenea, tot în funcția main, va face inițializările necesare pentru a putea trimite informații la procesul **worker** printr-un canal anonim (prin comunicații unu-la-unu) și, respectiv, pentru a putea primi

rezultate înapoi printr-un canal fifo (prin comunicații unu-la-unu).
iii) De asemenea, tot în funcția main, va face inițializările necesare pentru a putea schimba informații cu procesul **analyzer**, în ambele sensuri, prin acea mapare nepersistentă cu nume descrisă în specificația programului **analyzer**.
iv) Într-o funcție separată, apelată din funcția main, programul va citi din acea mapare nepersistentă cu nume, cuvintele transmise lui de către procesul **analyzer,** și pentru fiecare cuvânt citit va genera lista de trigrame asociată lui (numim *trigramă* o secvență consecutivă de trei caractere din cadrul unui șir de caractere), fără a elimina eventualele duplicate (e.g., pentru cuvântul “abcabc” se vor genera 4 trigrame: “abc” “bca” “cab” “abc”). Fiecare trigramă va fi transmisă mai departe către procesul **worker,** prin intermediul canalului anonim, folosind apeluri POSIX.
v) Într-o altă funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, secvența de trigrame transmise lui de către procesul **worker,** prin intermediul canalului fifo, și le va procesa în felul următor: va reține doar prima și ultima trigramă primite, și va contoriza numărul total de trigrame primite. Apoi va transmite rezultatele procesării (i.e., prima trigramă, ultima trigramă, și numărul total) către procesul **analyzer,** prin acea mapare nepersistentă cu nume.

4. Programul "worker.c" va implementa funcționalitatea descrisă în specificația următoare:

i) În funcția main, va face inițializările necesare pentru a putea primi informații de la procesul **service** printr-un canal anonim (prin comunicații unu-la-unu) și, respectiv, pentru a-i putea trimite rezultate înapoi printr-un canal fifo (prin comunicații unu-la-unu).
ii) Într-o funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, rând pe rând, fiecare trigramă transmisă lui de către procesul **service** prin intermediul canalului anonim și va selecta doar trigramele de forma “consoană vocală consoană”, fără a stoca în memoria proprie întreaga secvență de informații primite. Fiecare trigramă selectată va fi transmisă către procesul **service**, prin intermediul canalului fifo, folosind apeluri POSIX.

Recomandare:

Mai întâi, desenați de mână pe o foaie de hârtie o schemă cu **arhitectura aplicației**: ierarhia de procese, mijloacele de comunicație între procese și sensul de transmitere a informației prin aceste mijloace de comunicație (precum ați văzut în exemplele de diagrame realizate de mână, atașate la unele exerciții rezolvate în suporturile online de laborator).
Schema vă va ajuta să vă clarificați mai bine cerințele specificate în enunțul problemei și, de asemenea, vă va ajuta la partea de implementare!
De asemenea, citiți mai întâi și baremul asociat acestui subiect, pentru a vă clarifica mai bine cerințele specificate în enunțul problemei și modul în care trebuie să fie implementate.

Submitere:

La finalul testului, pentru a submite rezolvarea dvs. printr-un formular Google ce vă va fi indicat, veți face o arhivă (în format .zip) care să conțină cele trei programe sursă C (plus eventuale fișiere header .h proprii, în cazul în care veți defini și folosi astfel de fișiere), scriptul starter.sh și fișierul cu datele de intrare, **cu numele lor și poziția în ierarhia de directoare exact precum au fost specificate** în enunțul de mai sus. Iar arhiva o veți denumi în felul următor: TP2_NumePrenume.zip