Test practic la SO – varianta nr. 4

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:

```
.
├── starter.sh
├── data
│   └── input.txt
├── main_app
    └── supervisor.c
    └── procesators
        └── worker1.c
        └── worker2.c
```

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 **worker2** (din subdirectorul corespunzător) în *background*, iar după o pauză de 1 secundă, va porni și execuția programului **supervisor** (din subdirectorul corespunzător).
iii) Iar dacă *p*=1, atunci scriptul va porni mai întâi execuția programului **supervisor** (din subdirectorul corespunzător) în *background*, iar după o pauză de 2 secunde, va porni și execuția programului **worker2** (din subdirectorul corespunzător).
iv) În ambele situații, programul supervisor 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 supervisor).
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 "supervisor.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 un text în limba engleză, ce conține litere mari, litere mici, cifre și simboluri.
i) În funcția main, programul va verifica primirea argumentului la linia de comandă și va face inițializările necesare pentru a putea schimba informații cu procesul **worker1**, 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 pe rând, una câte una, fiecare linie din fișierul input.txt, va selecta din linia respectivă doar caracterele ce sunt litere mari, litere mici și cifre, și le va transmite către procesul **worker1** prin intermediul acelei mapări nepersistente cu nume.
iii) Într-o altă funcție separată, apelată din funcția main, programul va citi două numere întregi transmise lui de către procesul **worker1** prin intermediul acelei mapări nepersistente cu nume și va calcula suma acestor numere, afișând-o la final pe ecran. Dacă suma calculată este număr par, programul se va termina cu codul de exit 0, iar dacă este număr impar, se va termina cu codul de exit 1.

3. Programul "worker1.c" va implementa funcționalitatea descrisă în specificația următoare:

i) În funcția main, va face inițializările necesare pentru a putea trimite informații la procesul **worker2** printr-un canal fifo (prin comunicații unu-la-unu) și, respectiv, pentru a putea primi rezultate înapoi printr-un canal anonim (prin comunicații unu-la-unu).
ii) De asemenea, tot în funcția main, va face inițializările necesare pentru a putea schimba informații cu procesul **supervisor**, în ambele sensuri, prin acea mapare nepersistentă cu nume descrisă în specificația programului **supervisor**.
iii) Într-o funcție separată, apelată din funcția main, programul va citi din acea mapare nepersistentă cu

nume, toate caracterele (litere mari, litere mici și cifre) transmise lui de către procesul **supervisor,** și va procesa informatia citită astfel: calculează suma cifrelor primite și o reține într-o variabilă *sum_digits*, iar fiecare caracter ce nu este cifră (așadar, doar literele mari și literele mici) va fi transmis mai departe către procesul **worker2,** prin intermediul canalului fifo, folosind apeluri POSIX.
iv) Într-o altă funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, toate caracterele și un număr întreg (reprezentând numărul de transformări realizate în worker2) transmise lui de către procesul **worker2,** prin intermediul canalului anonim**,** și va calcula suma codurilor ASCII (i.e., codul ASCII = int(caracter)) ale tuturor caracterelor primite; iar la final, va scădea din suma astfel calculată numărul întreg primit de la worker2 (i.e., numărul de transformări realizate). Rezultatul astfel obținut, precum și valoarea reținută în variabila *sum_digits* (i.e., suma cifrelor primite calculată la punctul iii) de mai sus) vor fi transmise mai departe către procesul **supervisor,** prin acea mapare nepersistentă cu nume.

4. Programul "worker2.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 **worker1**.
ii) Tot în funcția main, va face inițializările necesare pentru a putea primi informații de la procesul **worker1** printr-un canal fifo (prin comunicații unu-la-unu) și, respectiv, pentru a-i putea trimite rezultate înapoi printr-un canal anonim (prin comunicații unu-la-unu).
iii) Într-o funcție separată, apelată din funcția main, programul va citi, folosind apeluri POSIX, rând pe rând, fiecare caracter transmis lui de către procesul **worker1** prin intermediul canalului fifo și va transforma fiecare literă mare primită în litera mică corespondentă, iar literele mici le păstrează neschimbate. Totodată, va contoriza câte transformări majusculă → minusculă a realizat. Toate literele astfel procesate, precum și valoarea întreagă reprezentând numărul de transformări realizate, vor fi transmise către procesul **worker1**, prin intermediul canalului anonim, 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