Fisiere – Exercitii

1. Scrieti o functie "creaza" care primeste ca parametri un specificator de fisier si un string continand o bara de drepturi, de exemplu:

```
creaza("f","rwxr_xr_x")
```

si creaza fisierul respectiv, cu drepturile respective, returnand 0. Daca fisierul exista sau nu se poate crea, va returna un cod nenul. Functia va folosi doar functii de la nivelul inferior de prelucrare fisierelor. Scrieti un program ilustrativ pentru aceasta functie - programul va primi ca argumente in linia de comanda fisierul si bara de drepturile si va crea fisierul respectiv; in caz ca nu reuseste, va afisa un mesaj de eroare.

2. Scrieti o functie "getint" care primeste ca argument un descriptor de fisier (presupus a fi deschis in prealabil pentru citire) si citeste cu format zecimal un intreg din fisierul respectiv, returnand acel intreg. Functia nu va folosi decat functii de la nivelul inferior de prelucrare a fisierelor. Deci, daca se poate citi intregul, apelul:

```
int x; x=getint(d);
```

este echivalent cu:

```
int x; FILE *f;
f=fdopen(d,"r");
fscanf(f,"%d",&x);
```

(deci functia "getint" va citi caractere cat timp ele se potrivesc cu o reprezentare externa de intreg zecimal si la sfarsit va returna intregul respectiv; cand un caracter nu se mai potriveste, indicatorul in fisier este mutat inapoi cu lseek). In citirea intregului, functia "getint" va ignora eventualele caractere albe (blank, tab, cap de linie) prealabile. Daca de la bun inceput intregul nu se poate citi (primul caracter nealb nu poate face parte dintr-o reprezentare de intreg sau dupa eventual caractere albe s-a intalnit sfarsitul fisierului) functia va returna 0 si va seta adecvat errno.
Scrieti un program ilustrativ pentru aceasta functie.

3. Scrieti un program care primeste ca argument in linia de comanda un fisier si inlocuieste in el orice succesiune de caractere albe (blank, tab, cap de linie) cu un singur blank. Se vor folosi doar functii de la nivelul inferior de prelucrare a fisierelor, fara lseek si nu se vor folosi fisiere auxiliare. Indicatie: fisierul va fi accesat prin doi descriptori, unul in citire, altul in scriere.

4. Scrieti un program care sorteaza caracterele dintr-un fisier al carui specificator este dat ca argument in linia de comanda. Nu se vor folosi fisiere auxiliare.

5. Scrieti un program care numara aparitiile unui string intr-un fisier dat. Stringul si specificatorul fisierului sunt dati ca argumente in linia de comanda.

6. Scrieti un program care inverseaza ordinea caracterelor intr-un fisier al carui specificator este dat ca argument in linia de comanda. Nu se vor folosi fisiere auxiliare.

7. Scrieti un program care inlocuieste intr-un fisier toate aparitile lui string1 cu string2. Fisierul si cele doua stringuri sunt date ca argumente in linia de comanda.

8. Comanda **tee** citeste de la standard input si afiseaza la standard output si in fisierele precizate ca argument in linia de comanda. Prin default, **tee** suprascrie continutul fisierelor (daca acestea exista). Implementati optiunea **-a** ce concateneaza la sfarsitul fisierelor deja existente in loc sa le suprascrie.