Sisteme de operare
Laborator 2
File I/O

1. Creati un nou subdirector *lab2/* in structura de directoare a laboratorului creata anterior (*SO/laborator*) si subdirectoarele aferente *doc src* si *bin*. Nu uitati sa actualizati variabila de mediu PATH pentru a include directorul *SO/laborator/lab2/bin*.

2. Scrieti un program C **mycp.c** care copiaza in bucla informatia citita de la standard input (file descriptor 0) la standard output (file descriptor 1). Cum puteti folosi acest program pentru a simula efectul comenzii *cp*, care copiaza un fisier sursa intr-un fisier destinatie?

3. Scrieti un program C **hole.c** care creaza un fisier cu "goluri/gauri" in el. De pilda, dupa ce ati creat fisierul, scrieti un sir de caractere la inceputul fisierului, mutati offsetul curent din fisier la valoarea 100 si scrieti un alt sir de caractere acolo. Care este dimensiunea fisierului nou creat in octeti afisata de comenzi cum ar fi *ls -l <fisier_cu_goluri>* sau *du -sb <fisier_cu_goluri>*?

Dar comanda *od -c <fisier_cu_gauri>* ce va arata?

4. Scrieti un program C **true-du.c** care primeste un fisier ca parametru si afiseaza dimensiunea lui in octeti fara goluri (adica dimensiunea sa reala, fara goluri, avand in vedere ca acestea nu se stocheaza pe disc).

5. Scrieti un program C **myfile.c** care emuleaza comanda *file*. Mai exact, programul primeste o serie de fisiere ca argumente in linie de comanda si afiseaza ce tip de fisiere sunt. De exempu puteti folosi urmatoarea linie de comanda:

```
$ myfile /home myfile.c /dev/tty1 /dev/sda1 my_simbolic_link myfifo
```

Indicatie: folositi apelul sistem *lstat* si comenzile *ln* (pentru a crea cu flag-ul *-s* un link simbolic in directorul curent, de pilda) si *mknod* (pentru a crea in directorul curent, de pilda, un fisier special de tip pipe/FIFO).

6. Scrieti un program C **temp.c** care simuleaza felul in care programele complexe gestioneaza fisierele temporare. In acest sens, mai intai creati un fisier "tempfile" cu urmatoarea comanda:

```
$ echo "this is a temporary file" > tempfile
$ cat tempfile
$ ls -l tempfile
```

Programul deschide fisierul "tempfile" folosind apelul sistem *open* apoi sterge fisierul folosind apelul sistem *unlink* si afiseaza pe ecran un mesaj din care sa reiasa ca fisierul "tempfile" a fost sters. Apoi simuleaza executarea unui task de 15 secunde apeland *sleep*. Dupa ce programul iese din *sleep*, citeste continutul fisierului "tempfile", il afiseaza pe ecran si termina executia.

Pentru a intelege mai bine ce se intampla, rulati programul in background (folosind "&"). De exemplu, daca programul se numeste *temp.c* si va aflati in subdirectorul *src*:

```
$ gcc -o ../bin/temp temp.c
$ temp &
```

Dupa ce programul a afisat pe ecran notificarea stergerii fisierului “tempfile”, rulati comanda

```
$ ls -l tempfile
```

pentru a va convinge ca programul a fost intr-adevar sters. Asteptati 15 secunde. Ce se intampla? Cum va explicati efectul executiei programului daca reflectati la suita de operatii executate?

7. Scrieti un program C **myls.c** care afiseaza continutul unui director furnizat ca parametru al programului, simuland functionarea simpla, neparametrizata, a comenzii *ls*.

Indicatie: Folositi apelurile sistem *opendir*/*readdir*/*closedir*.

8. Scrieti un program C **mycd.c** care primeste un director ca parametru si schimba directorul curent la valoarea parametrului primit folosind *chdir*. Apoi, foloseste apelul sistem *getcwd* pentru a afla directorul curent si il tipareste pe ecran.

Testati programul utilizand urmatoarea secventa de comenzi:

```
$ pwd
$ mycd /etc
$ pwd
```

Ce observati? De ce e *cd* comanda interna a shell-ului?