

## Sisteme de operare

### Laborator 3

### File I/O (cont.)

1. Creati un nou subdirector *lab3/* 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/lab3/bin*.

2. 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 exemplu puteti folosi urmatoarea linie de comanda:

```
$ myfile /home myfile.c /dev/tty1 /dev/sda1 my_symbolic_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/mkfifo* (pentru a crea in directorul curent, de pilda, un fisier special de tip FIFO).

3. 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 *tempfile* 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 functia de biblioteca *sleep* (*man 3 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?

4. 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*.

Folositi ulterior apelurile sistem *stat/fstat/lstat* pentru a implementa formatul lung al comenzii:

*\$ myls -l <dirname/filename>*

5. 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?

6. Scrieti doua programe **ping.c** si **pong.c** care comunica printr-un fisier FIFO. Programele deschid fisierul FIFO atat pentru scriere cat si pentru citire. Programul **ping.c** scrie mesajul "*ping\n*" in fisier, doarme 5 secunde folosind apelul de biblioteca *sleep (man 3 sleep)*, si apoi citeste din FIFO raspunsul programului **pong.c** pe care il afiseaza pe ecran. Programul **pong.c** citeste un mesaj din FIFO pe care il afiseaza pe ecran iar apoi raspunde cu mesajul "*pong\n*".

Indicatie: Pentru a crea fisierul FIFO folositi comenzile *mknod/mkfifo*.