Laborator 1 – Retele de calculatoare
Recapitulare – IPC, procese/thread-uri, semnale

1. Scrieti un program care creaza un **pipe** intre un proces parinte si un copil. Parintele scrie un mesaj in pipe si asteapta sa se termine procesul copil. Copilul citeste mesajul trimis de parinte in pipe si il afiseaza pe ecran. Primul lucru pe care il fac atat parintele cat si copilul dupa crearea pipe-ului este sa inchida partea de pipe pe care nu o folosesc la comunicarea dintre ei.

2. Cititi paginile de manual ale apelului de sistem **pipe** (man 2 pipe, man 7 pipe) si aflati ce se intampla daca un proces scrie intr-un pipe catre un alt proces care a inchis capatul sau de citire, respectiv daca un proces citeste dintr-un pipe in care nu mai scrie nici un proces (i.e, procesele care scriau in pipe au inchis capatul lor de scriere). Modificati programul de la punctul 1 a.i. :
- procesul copil nu inchide capatul sau de scriere in pipe
- dupa ce a citit din pipe mesajul scris de procesul parinte, copilul scrie inapoi in pipe mesajul “done\n” si se termina
- dupa ce a scris in pipe primul mesaj pentru copil, parintele asteapta sa citeasca din pipe mesajul “done\n” scris de copil si apoi il afiseaza pe ecran.

Ce se intampla? Cum explicati situatia?

3. Scrieti un program in care un proces parinte creeaza un proces copil si apoi se blocheaza asteptand ca utilizatorul sa apeze o tasta (*Indicatie*: puteti folosi getchar();). Procesul copil afiseaza un mesaj care contine propriul process ID (PID) ca si pe cel al procesului parinte, dupa care termina executia cu un cod de eroare (sa zicem 44). Cum poate procesul parinte sa afle in cadrul programului ca procesul copil s-a terminat si codul sau de terminare fara sa apeleze blocant **wait**? *Indicatie*: scrieti un handler de semnal pt. SIGCHLD.

4. Scrieti un program care creeaza un proces copil cu ajutorul apelului sistem **vfork**. Procesul copil incrementeaza o variabila globala **global** initializata cu valoarea 6 si o variabila automatica **local** initializata de procesul parinte cu valoarea 10, dupa care se termina cu cod de iesire 0 (executie cu succes). Imediat dupa apelul vfork, procesul parinte afiseaza valorile variabilelor **global** si **local** si termina executia cu cod zero.

Ce se intampla? Cum explicati situatia?

5. Scrieti un program care creeaza un proces copil cu ajutorul apelului sistem **vfork** si creeaza un **pipe** intre cele doua procese. Procesul copil scrie in pipe caracterul ‘a’ si asteapta sa citeasca un caracter din pipe. Apoi afiseaza caracterul pe care l-a citit din pipe si incheie executia cu succes (cod zero). Procesul parinte asteapta sa citeasca un caracter din pipe si apoi scrie caracterul ‘b’ in pipe dupa care afiseaza pe ecran caracterul pe care l-a citit din pipe.

Ce se intampla? Cum explicati situatia?

6. Scrieti codul pentru thread-urile producator si consumator in solutia cu thread-uri pentru problema producator-consumator schitata in codul furnizat pe site.