Sisteme de operare
Laborator 13
Servere concurente si multiprotocol TCP/UDP

1. Creati un nou subdirector *lab13/* 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/lab13/bin*.

2. Modificati serverul **timed-udp.c** pt serviciul *time* din laboratorul trecut pentru a deveni un *server multiprotocol*, care raspunde atat la cereri TCP cat si UDP din partea clientilor.

Indicatie: folositi *select*.

3. Modificati serverul **echod-tcp.c** pentru serviciul *echo* TCP scris in laboratorul trecut a.i. sa devina un *server concurent* (i.e., sa fie capabil sa raspunda la cereri concurente ale clientilor). In acest scop, folositi apelul sistem *select*.

4. Scrieti un server TCP concurent **echod-tcp.c** pentru serviciul de *echo* care sa raspunda la cererile clientului *echo* TCP din laboratoarele trecute. Pentru fiecare cerere client, serverul (proces master) creeaza un nou proces (copil, slave) folosind apelul sistem *fork* care va trata separat cererea client, in vreme ce serverul parinte (master) revine la activitatea de acceptare a cererilor clientilor.

Obs: Pentru a opera in mod concurent, serverul parinte nu poate astepta in mod blocant terminarea proceselor (copii) care trateaza cererile client. Cum procedati pentru ca procesele copil sa nu ramana in starea de zombie?

5. Folositi shell script-ul din laboratorul trecut pentru a verifica functionalitatea concurenta a serverelor **echod** de mai sus (exercitiile **3** si **4**).

6. Implementati propria versiune a apelului de biblioteca *usleep*, care suspenda executia programului pentru un numar de microsecunde dat:

int my-usleep(unsigned int usecs);

Indicatie: folositi *select*.

7. Scrieti un program **mmap-cp.c** care copiaza un fisier sursa intr-un fisier destinatie folosind apelul sistem *mmap*. Numele fisierelor sunt furnizate in linie de comanda ca mai jos:

$ mmap-cp <fisier-sursa> <fisier-destinatie>

si programul le mapeaza in memorie cu ajutorul *mmap* inainte de a face copierea datelor.

Dupa ce ati testat corectitudinea programului, folositi un fisier de dimensiune relativ mare (eg, cca 100MB) drept fisier sursa si comparati performanta **mmap-cp** cu cea a programului **mycp** din laboratorul 2 folosind comanda *time* in felul urmator:

$ time mmap-cp fisier-100MB copie-mmap
$ time mycp < fisier-100MB > copie-mycp

Cum interpretati rezultatele comenzii *time*? Ce va spune breakdown-ul timpilor de executie?