Sisteme de operare
Laborator 6
Controlul proceselor

1. Creati un nou subdirector *lab6/* 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/lab6/bin*.

2. Scrieti un program C **my-exec-engine.c** care primeste ca parametru in linie de comanda un program executabil si apeleaza *vfork* pentru a crea un nou proces. Procesul copil executa comanda primita ca argument folosind apelul sistem *execvp*. Parintele asteapta terminarea executiei copilului folosind apelul sistem *wait*. Ca exemplu, comanda trebuie sa functioneze in felul urmator:

$ my-exec-engine ls -l -a -R

3. Scrieti un program C **my-script-engine.c** care primeste ca parametru in linie de comanda un shell script. Programul parseaza prima linie a scriptului in cautarea interpretorului care trebuie lansat in executie. Odata identificat interpretorul, programul apeleaza *fork* pentru a crea un nou proces si apoi lanseaza in executie interpretorul (folosind *execv*) in contextul procesului copil pentru a executa scriptul. Parintele asteapta terminarea executiei copilului folosind apelul sistem *wait*.

Puteti folosi ca exemple scripturile de pe slide-urile de laborator, stocate in fisierele executabile *myscript*, respectiv *myawk* ca mai jos:

$ my-script-engine myscript
$ my-script-engine myawk
$ my-script-engine myawk 1 2 3 4 5

Obs: nu uitati ca interpretoarele de pe prima linie a unui shell script pot primi si parametri, asa cum este cazul scriptului *myawk* care apeleaza programul *awk* cu flagul *-f*.

4. Scrieti un program C **system.c** care simuleaza comportamentul programului de la punctul 2 folosind functia de biblioteca *system*. Cum ati scrie varianta proprie a acestui program folosind *fork*/*exec*?

Indicatie: pentru a folosi functia *system* e nevoie sa concatenati string-urile furnizate ca parametri de apel ai programului (argv[i]). In acest sens, puteti folosi functia de biblioteca *strcat*.

5. Folositi experienta dobandita in exercitiile de mai sus pentru a scrie un program C **my-shell.c** care citeste in bucla comenzi de la tastatura si le executa. Mai exact, programul ruleaza intr-o bucla infinita in care afiseaza pe ecran promptul “my-shell> “, citeste datele introduse la tastatura si foloseste o functionalitate similara celei *my-exec-engine* pentru a executa comanda introdusa de utilizator. Programul se termina cand utilizatorul tasteaza Ctrl-d (adica EOF).

Pentru exemplificare, urmatoarele comenzi trebuie sa poate fi executate de my-shell:

$ my-shell
my-shell> ls -l
my-shell> my-exec-engine ls -l
my-shell ./myawk 1 2 3 4 5
my-shell> my-script-engine awk 1 2 3

my-shell> Ctrl-d
$

6. Scrieti un program C **times.c** care masoara timpii de executie ai unuia dintre programele de mai sus folosind apelul sistem *times*.