Sisteme de operare
Laborator 1
Interpretorul de comenzi, procesul de login

1. Creati o structura de directoare pe care sa o folositi pentru programele de la laborator, dupa cum urmeaza:

```
$ cd                          # schimbati directorul in home directory
                              # echivalent cu “cd $HOME”

$ pwd                         # afisati valoarea directorului curent
                              # echivalent cu “echo $PWD”

$ mkdir -p SO/laborator/lab1  # creati directorul SO cu subdirectoarele sale

$ cd SO/laborator/lab1        # schimbati directorul
$ pwd
$ mkdir doc src bin           # creati subdirectoarele doc, src, bin in directorul curent
$ ls -l                       # afisati continutul directorului curent
```

Creati apoi un fisier *.myrc* (un fisier de tip *run commands*, asa cum e si fisierul *.bashrc* din home directory) in directorul SO care sa creeze alias-uri pt comenzile *rm* si *ls* si sa extinda variabila de mediu PATH dupa cum urmeaza:

```
$ cd ../../                   # reveniti in directorul SO
$ pwd
$ echo alias rm=\’rm -i\’ >> .myrc   # adaugati la continutul fisierului .myrc
                              # comanda care stabileste ca stringul “rm”
                              # este un shortcut pentru comanda “rm -i”

$ echo alias l=\’ls -l\’ >> .myrc    # similar pentru “ls -l” abreviat cu “l”

$ ls -la                      # flagul -a permite afisarea fisierelor al
                              # caror nume incepe cu ., cum e si .myrc, care
                              # altfel nu sunt vizibile la un simplu “ls”

$ cat .myrc                   # afiseaza la stdout continutul fisierului .myrc

$ env                         # afisati continutul variabilelor de mediu

$ env | grep PATH             # string matching, grep selecteaza doar
                              # continutul variabilei PATH dintre toate
                              # variabilele de mediu afisate de env
                              # comanda de tip pipeline, iesirea comenzii env
                              # e redirectata catre intrarea comenzii grep

$ env | grep HOME             # comenzile env si grep se lanseaza
                              # concurent/paralel si se sincronizeaza prin pipe
                              # (canal de comunicatie  unidirectional)

$ echo export PATH=$PATH:$HOME/SO/laborator/lab1/bin >> .myrc
```

```
                                    # extinde continutul variabilei PATH cu
                                    # stringul “$HOME/SO/laborator/lab1/bin”
                                    # comanda interna export face variabila PATH
                                    # disponibila tuturor programelor care se
                                    # executa din shell
                                    # orice executabil care se gaseste intr-un
                                    # director din PATH poate fi executat din
                                    # orice alt director fara a i se furniza calea
                                    # completa
                                    # executabilele care nu se gasesc intr-un director
                                    # din PATH trebuie specificate cu cale completa
                                    # pentru a putea fi executate

$ echo $PATH ; echo $HOME           # afisati valorile variabilelor PATH si HOME
                                    # comanda compusa, lista de comenzi care se
                                    # executa secvential
$ cat .myrc
$ source .myrc                      # executa toate comenzile stocate in .myrc
                                    # si pastreaza in shell valorile modificate
                                    # echivalent cu comanda “.  .myrc”

$ rm .myrc                          # ce se intampla daca stergeti fisierul .myrc?
                                    # dispare efectul comenzilor pe care le-ati
                                    # executat mai sus cu comanda source ?

$ cd -                              # schimbati directorul catre directorul anterior
$ pwd
$ l -a                              # l este ls -l dupa source-ul de mai sus, iar
                                    # flagul -a se adauga si comanda devine ls -la
                                    # mai exact ls -l -a
```

Pentru a intelege mai bine ce face fiecare comanda, cititi paginile de manual (eg, *man mkdir*). In subdirectorul *doc* veti salva documentatia de laborator, in subdirectorul *src* veti crea programele sursa C (folosind editorul text preferat), iar in subdirectorul *bin* veti pune programele executabile rezultate ca urmare a compilarii programelor sursa C.

Compilarea programelor C se va face cu comanda *gcc*, compilatorul C, dupa cum urmeaza:

```
$ cd  $HOME/SO/laborator/lab1/src       # acelasi lucru cu
                                        # cd ~/SO/laborator/lab1/src
                                        # ~ este echivalent cu home directory
$ pwd
$ gcc -o ../bin/fisier_executabil fisier_sursa.c
                                    # compileaza fisierul sursa C din directorul
                                    # curent, i.e. lab1/src, si plaseaza executabilul
                                    # nou creat in directorul lab1/bin
```

unde *fisier_executabil* si *fisier_sursa* sunt nume de fisiere pe care le alegeti dupa dorinta fiecaruia.

Apoi puteti executa programul executabil rezultat, din orice director, cu comanda

```
$ fisier_executabil
```

De ce e posibil asa ceva? In fond, *fisier_executabil* se afla in subdirectorul *lab1/bin* si in principiu ca sa poate fi executat trebuie furnizata calea completa a executabilului sau trebuie schimbat directorul in *lab1/bin* inainte de a executa programul. Compilati programul urmator, de la punctul 2, si vedeti ce se intampla.

2. Scrieti un program C *pwnam.c* in directorul *lab1/src* care primeste ca parametru in linia de comanda un nume de utilizator si afiseaza informatii despre utilizator cum ar fi: user si group ID-ul, home directory, shell. Compilati executabilul cu numele *pwnam* in directorul *lab1/bin* cf. comenzilor de mai sus si executati programul:

```
$ gcc -o ../bin/pwnam pwnam.c
$ pwnam
```

Dupa compilarea si executarea programului, stergeti executabilul si apelati comanda *gcc* fara flag-ul *-o*, ca mai jos:

```
$ rm ../bin/pwnam
$ pwnam
$ gcc pwnam.c
$ l                    # sau ls -l daca nu va functioneaza alias-ul de mai sus
```

Cine e fisierul *a.out*? Ce se intampla daca il executati ca mai jos?

```
$ ./a.out
```

Dar daca incercati sa-l executati astfel:

```
$ a.out
```

De ce nu mai functioneaza? Ce legatura are cu variabila de mediu PATH?

Cum puteti sa recreati comanda *pwnam* (rezultatul compilarii de mai sus) folosind fisierul *a.out* , fara sa recompilati programul *pwnam.c*?

<u>*Indicatie*</u>: a. Folositi functia *getpwnam* (*man getpwnam*). Numele de utilizator il puteti lua din fisierul */etc/passwd* pe care il puteti vizualiza cu comenzile de tip pager *more/less* (*man more*, *man less*).
b. Cautati in paginile de manual ce fac comenzile *cp* si *mv*:

```
$ man cp
$ man mv
```

3. Scrieti un program C *getids.c* in directorul *lab1/src* care afiseaza PID-ul, UID-ul si GID-ul procesului curent. Nu uitati ca desi este doar un program, el apartine unui utilizator (cel care il ruleaza) si atunci pe langa PID are si UID si GID. Compilati executabilul cu numele *getids* in directorul *lab1/bin* cf. comenzilor de mai sus.

<u>*Indicatie*</u>: Folositi functiile *getpid*, *getuid*, *getgid*. Folositi comanda *man* pentru a afla cum sa le utilizati.

4. Exersati comezile shell din prezentarea laboratorului, deopotriva comenzi interne si externe. Pentru a afla calea absoluta a unui executabil (comanda externa) folositi comanda

$ which <cmd>

Pentru a cauta un fisier oarecare dupa un substring din numele sau folositi comanda

$ locate <string>

Daca *locate* nu functioneaza, instalati comanda folosind package managerul sistemului Linux pe care il folositi si apoi apelati *updatedb* in calitate de root (indicatie valabila doar daca lucrati pe laptopul personal, nu si pe calculatoarele din laborator).