Sisteme de operare
Laborator 2
Interpretorul de comenzi, procesul de login (cont.)

1. Creati un nou subdirector *lab2/* 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/lab2/bin*.

2. Scrieti un program C *getenv.c* care listeaza continutul variabilelor de mediu folosind apelul de biblioteca *getenv* (*man getenv*). Programul primeste ca parametru numele unei variabile de mediu (eg, HOME) si tipareste pe ecran continutul variabilei respective. In absenta oricarui parametru tipareste toate variabilele de mediu si continutul lor.

Indicatie: Parcurgeti vectorul de string-uri *envp* (al treilea argument de apel al functiei *main*) in cautarea numelui variabilei de mediu. Pentru comparatia de string-uri folositi apelurile de biblioteca *strcmp*/*strncmp* (*man strcmp*, *man strncmp*).

3. Scrieti un program C *myecho.c* care primeste ca parametru in linie de comanda un string si il tipareste pe ecran (*stdout*, file descriptor 1). Modificati programul a.i. sa citeasca stringul de la tastatura (*stdin*, file descriptor 0) dupa care il afiseaza pe ecran.

Indicatie: Pentru a citi/scrie date de pe un file descriptor folositi apelurile sistem *read/write* (*man 2 read*, *man 2 write*).

Obs: atunci cand citeste date de la tastatura, apelul sistem *read* asteapta tasta ENTER pentru a returna datele introduse de la tastatura.

Rulati prima versiune a programului dupa cum urmeaza:

```
$ gcc -o ../bin/myecho myecho.c
$ myecho “hello world”
```

Ce se intampla daca rulati comanda *myecho* astfel?

```
$ myecho hello world
```

Cum va explicati rezultatul?

Rulati cea de-a doua versiune a programului in mai multe feluri, de ex:

```
$ gcc -o ../bin/myecho myecho.c
$ myecho                                                    # pp. ca “lab2/bin” e in $PATH
$ echo “alt fel de a rula comanda myecho” | myecho
$ echo “si inca un alt fel de a rula comanda myecho” > out; myecho < out
```

Care e diferenta intre diferitele rulari? Odata ce s-a creat fisierul *out* ca mai sus, de ce nu functioneaza urmatoarele comenzi spre deosebire de versiunea a doua a programului *myecho* pe care ati scris-o?

```
$ echo < out
$ echo “hello world”  | echo
```

4. Scrieti un program C *mycp.c* care copiaza in bucla informatia citita de la standard input (file descriptor 0) la standard output (file descriptor 1) pana cand se ajunge la EOF (*end of file*). Cum puteti folosi acest program a copia un fisier sursa intr-un fisier destinatie?

Obs: pentru detectia EOF la *read* puteti testa valoarea de return a apelului sistem *read*. Pentru detalii cititi cu atentia pagina de manual (*man 2 read*).

Indicatie: intai exersati comanda *cat* (*man cat*) din shell pentru a afisa continutul unui fisier oarecare, de ex:

```
$ cat getenv.c
```

Apoi folositi comanda *cat* astfel:

```
$ cat  -  > out
```

Ce se intampla?

Obs1: spatiile suplimentare sunt introduse intentionat pentru a nu induce ideea ca e vorba de o sageata formata din doua caractere (“→”).

Obs2: Combinatia de taste *Ctrl^d* reprezinta caracterul EOF (*end of file*). Cand introduceti un text de la tastatura puteti incheia introducerea datelor cu *Ctrl^d* .

In particular, shell-ul este un program care asteapta in permanenta date de la tastatura. Ce se intampla cand tastati *Ctrl^d* la promptul shell-ului?