ARGUMENTELE LINIEI DE COMANDĂ.

1. Argumentele liniei de comandă şi parametrii funcţiei main()

Orice program **C** trebuie să aibă o funcţie **main**. Rutina de startup transmite funcţiei **main** trei parametrii (argumente): argc, argv şi env.

– **argc** este de tip int şi reprezintă numărul argumentelor din linia de comanda transmise funcţiei main;

– **argv** este un tablou de pointeri la caracter (char *[ ]). Fiecare element din acest tablou conţine adresa de început a unui şir de caractere, astfel: argv[0] conţine adresa de început a şirului de caractere care reprezintă numele complet al programului în curs de execuţie (include calea către directorul care conţine programul executabil); argv[1] conţine adresa de început a primului şir de caractere tastat în linia de comandă după numele programului (de care este separat printr-un spaţiu); argv[argc-1] conţine adresa de început a ultimului argument al liniei de comandă transmis funcţiei **main**; argv[argc] este NULL.

Aceste argumente (ce sunt şiruri de caractere separate prin spaţii) se specifică în linia de comandă sub forma:

nume_program arg_1 arg_2 ...arg_n

- **env** este tot un tablou de pointeri la caracter şi fiecare element din tablou conţine adresa unui șir de caractere care conține o caracteristică a sistemului pe care rulează programul. Nu se poate şti în avans (ca la argv) numărul de elemente din tablou, dar se ştie că ultimul element din tablou are valoarea 0 (NULL). Astfel explorarea tabloului **env** se continuă până când se întâlneşte un element al tabloului egal cu 0 (zero).

Cei trei parametri ai funcţiei **main** trebuie declaraţi exact în ordinea menţionată (chiar dacă se pot folosi şi alte nume). Sunt deci posibile 4 forme ale acestei funcţiei:

a) int main(void)

b) int main(int argc)

c) int main(int argc, char *argv[ ])

d) int main(int argc, char *argv[ ], char *env[ ])

În următorul exemplu, programul afişează pe monitor argumentele liniei de comandă şi variabilele de mediu specifice:

```
#include <stdio.h>
int main(int argc, char *argv[ ], char *env[ ])
{
```

```
        int i;
        for (i=0; i<argc; i++)
        {
            printf("argv[%d] : %s\n", i, argv[i]);
        }

        for (i=0; env[i]!=NULL; i++)
        {
            printf("env[%d] : %s\n", i, env[i]);
        }
        return 0;
    }
```

Observaţie:

Argumentele liniei de comandă pot conţine caractere albe dacă acestea sunt încadrate între ghilimele (care vor dispărea la prelucrarea argumentelor). Lungimea maximă a argumentelor liniei de comandă transmise funcţiei main (inclusiv numele programului şi spaţiile dintre argumente) este specifică fiecărui sistem de operare.

Argumentele liniei de comandă pot fi și șiruri care conţin caracterele joker (wildcard) (*, ?). Aceste argumente se comportă ca nişte "modele" ale numelor fişierelor care trebuie transmise către program. Caracterul ***** înlocuieşte un şir de caractere din nume, iar caracterul **?** înlocuieşte un singur caracter. În acest caz tabloul **argv** va fi completat cu adresele de început ale unor şiruri de caractere care conţin numele tuturor fişierelor care corespund modelului dat în linia de comandă. Acest proces se numeşte expandarea liniei de comandă (pentru că, în general, există mai multe nume de fişiere care corespund aceluiaşi "model"). Dimensiunea tabloului **argv** depinde de mărimea memoriei disponibile. Dacă nu se găseşte nici un fişier al cărui nume să se potrivească modelului din linia de comandă, argumentul este transmis neschimbat (nu se expandează). Argumentele încadrate între ghilimele nu sunt expandate. Deci dacă se vrea ca tabloul **argv** să conţină anumite fişiere, să spunem toate fişierele cu extensia **.c** din directorul de lucru, linia de comandă este (**p1** este numele programului):

p1 *.c

Dacă **p1** este programul din exemplul de mai sus, pe monitor se vor afişa toate fişierele cu extensia **.c** din directorul de lucru.

TEME

Problema nr. 1

Să se scrie un program care afișează pe monitor conținutul unui fișier al cărui nume este dat ca argument al liniei de comandă. Dacă nu sunt argumente în linia de comandă

se va afișa un mesaj în care se va specifica forma corectă a liniei de comandă și apoi programul se va încheia.

Problema nr. 2

Să se scrie un program care concatenează toate fişierele de tip **C** din directorul curent într-un fişier numit **final.c**. Linia de comandă este de forma:

p2 *.c

în care **p2** este numele programului.

Problema nr. 3

Scrieţi un program numit **tail** care tipăreşte un set de linii specificate într-o gamă dată ca argument în linia de comandă. Liniile se citesc de la tastatură (sau prin indirectare dintr-un fişier text) şi se depun într-un tablou de şiruri.

Exemplu:

```
tail 5-  /* tipăreşte toate liniile, începând cu linia 5 */
tail -10 /* tipăreşte primele 10 linii */
tail 2-8 /* tipăreşte 7 linii (inclusiv liniile 2 şi 8) */
tail     /* tipăreşte toate liniile */
```

Problema nr. 4

Să se scrie un program care citeşte de la tastatură un şir de numere până întâlneşte sfârșitul de fișier. Numerele pare vor fi scrise într-un fişier, iar cele impare în altul. Numele celor două fişiere vor fi date ca argumente în linia de comandă. Numerele citite pot fi formate din una sau mai multe cifre.

Problema nr. 5

Să se scrie un program care face analiza următoarei linii de comandă:

p5 [-p][-s][-m][-h] [fisier_intrare] [fisier_iesire]

unde opţiunile liniei de comandă au următoarele semnificaţii (parantezele pătrate indică faptul că parametrii respectivi sunt opţionali):

-p → prelucrarea 1

-s → prelucrarea 2

-m → prelucrarea 3

-h → afişarea unui mesaj de help (care descrie ce trebuie să facă programul şi forma liniei de comandă)

Opţiunea implicită de prelucrare este prelucrarea 3, iar în linia de comandă nu poate apare decât o singură opţiune.

În cazul în care în linia de comandă nu există numele nici unui fişier, citirea datelor se face de la tastatură, iar afişarea se face pe monitor. În cazul în care avem numele unui singur fişier, acesta se consideră a fi fişierul de intrare din care se citesc datele de intrare, iar rezultatele se afişează pe monitor.

Programul va scrie în fişierul de ieşire doar un mesaj care să indice numele fişierului de intrare şi prelucrarea care urmează a fi făcută.