



Programarea Calculatoarelor

Cursul 11: Fișiere text. Fișiere binare. Funcții de prelucrare a fișierelor. Argumente la execuția programelor

Ion Giosan

Universitatea Tehnică din Cluj-Napoca
Departamentul Calculatoare



- 2



Fișiere

- Tipuri de fișiere
 - Text – conținutul lui este reprezentarea text (scrisă) a datelor
 - Binare – conținutul lui este reprezentarea din memorie a datelor
- Procesarea conținutului unui fișier presupune
 - Deschiderea fișierului
 - Prelucrarea conținutului acestuia (citiri, scrieri)
 - Închiderea fișierului
- La cererea de deschidere a unui fișier
 - Fișierul a putut fi deschis
 - Pointer-ul la structura **FILE** nu este NULL
 - Se poate prelucra conținutul acestuia iar în final se poate închide fișierul
 - Fișierul nu a putut fi deschis
 - Pointer-ul la structura **FILE** este NULL
 - Nu se poate continua cu prelucrarea conținutului acestuia
 - Nu este necesară închiderea fișierului – acesta neputând fi deschis



- ```
FILE* fopen(const char *nume, const char *mod);
```

- **mod** reprezintă modul de prelucrare a fișierului deschis

- "r", "w", "a", "r+", "w+", "a+"**

- "rb", "wb", "ab", "r+b", "w+b", "a+b"**

- În modurile "r", "w", "r+", "w+", "rb", "wb", "r+b", "w+b" indicatorul este poziționat inițial la începutul fișierului

- În modurile "**a**", "**a+**", "**ab**", "**a+b**", indicatorul este poziționat inițial la sfârșitul fișierului

- După fiecare citire/scriere indicatorul este mutat mai departe cu numărul de octeți citiți/scriși



| Mod | Descriere                                                                                                                                             |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| r   | Deschiderea fișierului pentru citire. Fișierul trebuie să existe!                                                                                     |
| w   | Crearea unui fișier pentru scriere. Dacă fișierul există, conținutul acestuia este șters în totalitate!                                               |
| a   | Deschiderea sau crearea (dacă nu există) unui fișier pentru adăugarea de conținut numai la sfârșitul acestuia                                         |
| r+  | Deschiderea unui fișier pentru actualizarea conținutului (citire și scriere). Fișierul trebuie să existe!                                             |
| w+  | Deschiderea unui fișier pentru actualizarea conținutului (citire și scriere). Dacă fișierul există, conținutul acestuia este șters în totalitate!     |
| a+  | Deschiderea unui fișier pentru citirea conținutului și adăugarea de conținut numai la sfârșitul acestuia. Dacă fișierul nu există, acesta este creat. |



- Deschiderea unui fișier pentru citire în mod text

- Crearea unui fișier pentru scriere în mod binar

6



```
int fgetc (FILE * pf);
```

- Scrierea unui caracter în fișier

```
int fputc (int character, FILE * pf);
```

- Funcția returnează codul ASCII al caracterului scris în caz de succes și **EOF** în caz de eșec
- `fputc('x', stdout)` este echivalent cu `putchar('x')`



- ```
char * fgets ( char * str, int num, FILE * pf );
```

- Scrierea unui șir de caractere (*string*) în fișier

```
int fputs ( const char * str, FILE * pf );
```

- 8



- Citirea cu format din fișier

```
int fscanf ( FILE * pf, const char * format, ... );
```

- Funcția se comportă similar cu funcția **scanf** care citește de la intrarea standard (tastatură)
- Funcția returnează numărul de argumente citite corect în caz de succes și **EOF** în caz de eșec

- Scrierea cu format în fișier

```
int fprintf ( FILE * pf, const char * format, ... );
```

- Funcția se comportă similar cu funcția `printf` care scrie la ieșirea standard (pe ecran)
- Funcția returnează numărul de caractere scrise în caz de succes și o valoare negativă în caz de eșec



```
int fflush ( FILE * pf );
```

- Dacă fișierul este deschis în scriere, conținutul zonei tampon se scrie efectiv în fișierul respectiv
 - Asigură, după apel, că fișierul stocat pe disc conține efectiv ceea ce a fost scris în el cu ajutorul funcțiilor prezentate anterior
- Dacă fișierul e deschis în citire, caracterele necitite încă din zona tampon (*buffer*) se pierd
 - Pentru golirea *buffer*-ului intrării standard (tastatură) se poate face apelul `fflush(stdin)`
- Funcția returnează zero în caz de succes și **EOF** în caz de eroare
- Observație: la închiderea oricărui fișier, zona tampon este golită automat



- Determinarea poziției curente a indicatorului într-un fișier

```
long ftell(FILE *pf);
```

- Funcția returnează octetul corespunzător poziției curente a indicatorului în fișier (relativă față de începutul fișierului)

- Poziționarea indicatorului într-un fișier pe un anumit octet

```
int fseek (FILE *pf, long deplasament, int origine);
```

- **deplasament** definește numărul de octeți cu care se va face deplasarea indicatorului față de origine

- **origine** poate fi una din constantele

- **SEEK_SET** – începutul fișierului
- **SEEK_CUR** – poziția curentă a indicatorului în fișier
- **SEEK_END** – sfârșitul fișierului

- **SEEK CUR** – poziția curentă a indicatorului în fișier

- **SEEK END** – sfârșitul fișierului

- Funcția returnează zero în caz de succes și altă valoare în caz de eșec



- ```
int feof(FILE *pf);
```

- Închiderea unui fișier

```
int fclose(FILE *pf);
```

- Funcția returnează zero dacă închiderea fișierului s-a realizat cu succes
- Funcția returnează **EOF** dacă închiderea fișierului a eșuat



- De obicei se folosește la atașarea unui fișier la unul dintre fișierele standard `stdin`, `stdout`, `stderr`
- Funcția returnează `pf` în caz de succes și `NULL` în caz de eșec
- Exemplu

- Închide orice alt fișier atașat la **stdout** și deschide **out.txt** care este atașat la **stdout**
- Tot ce va fi scris la ieșirea standard va fi redirectat și scris în fișierul **out.txt**



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
typedef struct {
 char nume[20];
 char UM[10];
 float cantitate;
 float pret;
} produs;
int main()
{
 FILE *pf;
 pf = fopen("produse.txt", "w");
 if (pf == NULL) {
 printf("Nu se poate crea fisierul!"); exit(1);
 }
 produs p[] = {{ "paine taraneasca", "buc", 35, 4.58796 },
 { "lapte dietetic", "litru", 85.58941, 3.4756 },
 { "oua de casa", "buc", 10865, 0.568974 }
 };
}
```



15







```
Paine taraneasca|buc|35.000000|4.587960
lapte sintetic|litru|85.589409|3.475600
oua de casa|buc|10865.000000|0.568974
=====
```

```
Paine taraneasca|buc|35.000000|4.587960
*85.589409*3.475600*
Este sfarsitul fisierului? 0
Caracter citit -1
Este sfarsitul fisierului? 16
Fisierul contine 126 octeti
```



- Diagram illustrating the structure of a record in a file:
- |         |         |         |     |         |         |         |     |
|---------|---------|---------|-----|---------|---------|---------|-----|
| ARTICOL | ARTICOL | ARTICOL | ... | ARTICOL | ARTICOL | ARTICOL | EOF |
|---------|---------|---------|-----|---------|---------|---------|-----|
- The record is composed of multiple fields, each labeled "ARTICOL", followed by an "EOF" (End of File) marker. The fields are grouped by a bracket labeled "înregistrare" (record).

- 18



- ```
unsigned fread ( void * ptr, unsigned dim,
                unsigned nr, FILE * pf );
```

- ```
unsigned fwrite (void * ptr, unsigned dim,
 unsigned nr, FILE * pf);
```

- **ptr** este pointer la zona de memorie unde se citește/scrie
- **dim** este dimensiunea unei înregistrări exprimată în octeți
- **nr** este numărul de înregistrări care se vor citi/scrie

- Se returnează numărul de înregistrări citite/scrise corect

- La citirea/scrierea elementelor unor tablouri

- Primul argument este pointer la elementul de unde începe citirea/scrierea
- Al doilea argument este dimensiunea unui element în octeți
- Al treilea argument este numărul de elemente care se vor citi/scrie



20



21



Conținutul fișierului **produse.dat** (vizualizat cu un editor de texte) la terminarea programului

22



```
paine taraneasca|buc|35.000000|4.587960
lapte dietetic|litru|85.589409|3.475600
oua de casa|buc|10865.000000|0.568974
```

```
paine taraneasca|buc|35.000000|4.587960
lapte sintetic|litru|75.589409|3.475600
oua de casa|buc|10865.000000|0.568974
```

**Fisierul contine 120 octeti**



- ```
int remove ( const char * filename );
```

- Exempu

```
if( remove( "fisier.txt" ) != 0 )
    perror( "Nu s-a putut sterge!" ); //scrie in stderr
else
    puts( "Sters cu succes!" );
```




Trimiterea argumentelor la execuția programului

- Pentru a putea trimite argumente la execuția programului, funcția **main** trebuie să respecte următorul prototip

```
int main( int argc, char *argv[] )
```
- Unde
 - **int argc** este numărul de argumente trimise
 - **char *argv[]** este un șir de *string*-uri constante, care conține numele argumentelor în ordine (**argv[0]** este primul argument – și care este întotdeauna numele programului)
 - Argumentele trebuie să fie separate prin unul sau mai multe spații
- Exemplu

```
...> utilitar.exe aduna 20 45 705
```

 - În urma acestui apel valorile pentru **argc** și **argv** sunt
 - **argc** : 5
 - **argv[0]** : "utilitar.exe"
 - **argv[1]** : "aduna"
 - **argv[2]** : "20"
 - **argv[3]** : "45"
 - **argv[4]** : "705"



26



Trimiterea argumentelor la execuția programului - exemplu

```
int main(int argc, char* argv[])
{
    if (argc<3) {
        printf("Eroare: numar insuficient de argumente!");
        exit(1);
    }
    if ((strcmp(argv[1], "aduna") !=0) &&
        (strcmp(argv[1], "inmulteste") !=0)) {
        printf("Eroare: operatia \"%s\" este
                necunoscuta!", argv[1]);
        exit(2);
    }
    if (strcmp(argv[1], "aduna")==0)
        printf("Suma argumentelor este %d",
                calcul("aduna", argc, argv));
    if (strcmp(argv[1], "inmulteste")==0)
        printf("Produsul argumentelor este %d",
                calcul("inmulteste", argc, argv));

    return 0;
}
```



Trimiterea argumentelor la execuția programului - exemplu

Rezultate obținute în urma execuției programului în diverse situații:

...>utilitar.exe aduna

Eroare: numar insuficient de argumente!

...>utilitar.exe anuleaza 3 4

Eroare: operatia "anuleaza" este necunoscuta!

...>utilitar.exe aduna 34 sw32 56

Eroare: argumentul "sw32" nu este numar intreg!

...>utilitar.exe aduna 34 32.35 56

Eroare: argumentul "32.35" nu este numar intreg!

...>utilitar.exe aduna 45 2 100

Suma argumentelor este 147

...>utilitar.exe inmulteste 45 2 100 30

Produsul argumentelor este 270000