

PROGRAMAREA CALCULATOARELOR FIȘIERE ȘI OPERAȚII CU FIȘIERE ÎN LIMBAJUL C

Prelegere

Kulev Mihail, dr., conf. univ.

Stimați studenți și stimată audiență!

Mă numesc Kulev Mihail sunt doctor, conferințiar universitar la Universitatea Tehnică a Moldovei. Din cadrul cursului PROGRAMAREA CALCULATOARELOR Vă propun spre atenția Dumneavoastră prelegerea cu tema:

” FIȘIERE ȘI OPERAȚII CU FIȘIERE ÎN LIMBAJUL C”

PROGRAMAREA CALCULATOARELOR FIȘIERE ȘI OPERAȚII CU FIȘIERE ÎN LIMBAJUL C

Prelegere

Kulev Mihail, dr., conf. univ.



FIȘIERE ȘI OPERAȚII CU FIȘIERE ÎN LIMBAJUL C

Conținutul prelegerii

1. Noțiuni de fișier.

Vom defini noțiunea de fișier precum și noțiunile de flux, canal (stream), vom afla cu ce tip de fișiere operează limbajul C și care sunt canale de intrare / ieșire standard.

2. Operații cu fișiere. Nivele și etape de prelucrare a fișierelor.

Vom specifica operațiile și nivelele existente de prelucrare a fișierelor, vom afla ce reprezintă pointer la fișier și structura de tip fișier FILE și vom stabili etapele de prelucrare a fișierelor.

3. Funcțiile standard pentru operații cu fișiere.

Vom prezenta, analiza și studia funcțiile standard de prelucrare a fișierelor de diferite tipuri și exemple de utilizare a funcțiilor.

Exemple de cod.

Pe parcursul studierii întrebărilor menționate vom prezenta exemple de cod în limbajul C

1. Noțiune de fișier

Un **fișier** este o colecție de date memorate (păstrate) pe un *suport extern* de memorie și identificate printr-un *nume fișier*.

Conținutul fișierelor reprezintă texte (ex. programe sursă), numere, alte informații binare: programe executabile, imagini sau sunete codificate numeric s.a. Limbajul C permite operarea cu fișiere:

- de **tip text** - un astfel de fișier conține o succesiune de linii separate prin caracterul new line ('\n'), fiecare linie are 0 sau mai multe caractere;
- de **tip binar** - un astfel de fișier conține o succesiune de octeți, fără nici o structură.

Noțiune de fișier

Noțiunea de fișier este mai generală și include orice flux (stream) de date din exterior spre memoria internă sau din memoria internă spre exterior. Prelucrarea unui fișier presupune asocierea acestuia cu un canal de I/E (I/O) (numit flux sau stream), care este gestionat de un pointer spre fișier. Există trei canale standard predefinite, care se deschid automat la lansarea unui program:

- **stdin** pentru fișier de intrare de tip text, este intrarea standard - cu date introduse de la tastatură;
- **stdout** pentru fișier de ieșire de tip text, este ieșirea standard - cu rezultatele afișate pe ecranul monitorului;
- **stderr** – fișier de eroare de tip text, este ieșirea standard - afișează mesajele de eroare pe ecranul monitorului.

2. Operații cu fișiere

Operațiile specifice prelucrării fișierelor sunt:

- deschiderea unui fișier
- închiderea unui fișier
- crearea unui fișier
- citirea de articole din fișier (consultarea fișierului)
- actualizarea (sau modificarea) fișierului
- adăugarea de articole la sfârșitul fișierului
- poziționarea pe o anumită poziție din fișier
- ștergerea unui fișier
- schimbarea numelui unui fișier

Nivele de prelucrare a fișierelor

Se cunosc două nivele de prelucrare a fișierelor și anume:

1. Nivelul inferior de prelucrare – se face apel direct la funcțiile sistemului de operare.

Pentru a putea utiliza prelucrarea fișierelor la nivel inferior, trebuie să includem în program următoarele fișiere standard: `<io.h>` și `<fcntl.h>`.

2. Nivelul superior de prelucrare – se folosesc funcții specializate în lucrul cu fișierele.

Pentru a putea utiliza prelucrarea fișierelor la nivel superior, trebuie să includem în program fișierul standard `<stdio.h>` și să folosim funcțiile standard din bibliotecă.

Noi vom studia și utiliza nivelul superior de prelucrare a fișierilor folosind funcțiile specializate din biblioteca standard a limbajului C.

Etape de prelucrare a unui fișier

Pentru a prelucra un fișier la nivel superior, trebuie parcurse următoarele etape:

- **se definește** o variabilă pointer la fișier de tip **FILE*** pentru accesarea și gestionarea fișierului;

FILE este un tip structură definit în `<stdio.h>`, care conține informații referitoare la fișier și la zona buffer de transfer de date între memoria internă și fișier (adresa, lungimea zonei buffer, modul de utilizare a fișierului, indicator (cursor) de poziție în fișier, indicator de sfârșit - end-of-file - EOF);

- **se deschide fișierul** pentru un anumit **mod de acces**, folosind funcția de bibliotecă **fopen()**, care și realizează asocierea între variabila pointer la fișier și numele extern al fișierului;
- **se prelucrează fișierul** în citire/scriere cu **funcțiile specifice**;
- **se închide fișierul** folosind funcția de bibliotecă **fclose()**.

3. Funcțiile standard pentru operații cu fișiere

Fișierele sunt accesate și gestionate prin pointeri la structura **FILE**, asociate fișierelor pe durata prelucrării. Fișierele standard au pointerii predefiniți: **stdin**, **stdout**, **stderr**.

Declararea unui pointer la fișier este următoarea:

Deschiderea unui fișier

- asociază unui *nume extern de fișier* un *pointer la fișier*,
- stabilește un *mod de acces* la fișier,
- stabilește o conexiune logică între fișier și variabila pointer **pf**,
- alocă o zonă de memorie de tip **FILE** pentru realizarea mai eficientă a operațiilor de intrare / ieșire

Funcțiile standard pentru operații cu fișiere

Pentru deschiderea unui fișier se folosește funcția:

```
FILE* fopen(const char *filename, const char *mod) ;
```

deschide fișierul cu numele extern **filename** pentru acces de tip **mod**.

Returnează **pointer la fișier** sau **NULL** dacă fișierul nu poate fi deschis;

valoarea returnată este memorată în variabila pointer **pf** de tip **FILE***, care a fost declarată pentru accesarea și gestionarea fișierului.

Funcțiile standard pentru operații cu fișiere

Modul de acces (șir de caractere – între 1 și 3 caractere) **poate fi :**

- **“r” – read only**, este permisă doar citirea (sau consultarea) dintr-un fișier existent, citirea dintr-un fișier inexistent va genera eroare;
- **“w” - write**, crează un nou fișier pentru scriere sau dacă există deja, distruge vechiul conținut;
- **“a” - append**, deschide pentru scriere într-un fișier existent (scrierea se va face în continuarea informației deja existente în fișier, deci, cursorul de poziție se plasează la sfârșitul fișierului);
- **“*+”** - permite scrierea și citirea - **actualizare** (Ex: “r+”, “w+”, “a+”).

Între citire și scriere trebuie repositionat cursorul de poziție;

- **“*b”** - specifică fișier de tip binar (Ex.: ”wb”, ”wb+”, ”w+b”);
- **“*t”** - specifică fișier de tip text (implicit).

Funcțiile standard pentru operații cu fișiere

Numele extern a fișierului poate include următoarele:

- Numele unității de disc sau partiției disc (ex: A:, C:, D:, E:)
- "Calea" spre fișier: succesiune de nume de fișiere catalog (director), separate printr-un caracter ('\ ' în MS-DOS și MS Windows, sau '/' în Unix și Linux)
- Numele propriu-zis al fișierului
- Extensia care indică tipul fișierului și care poate avea între 0 și 3 caractere în MS-DOS. Sistemele MS-DOS și MS-Windows nu fac deosebire între litere mari și litere mici, în cadrul numelor de fișiere
- **Atenție!** pentru separarea numelor de cataloage dintr-o cale se vor folosi:
- `\\` , pentru caracterul `\`
- sau caracterul `/`
- Ex.: `char *filename = "C:\\\\WORK\\T.TXT";` `char *filenum = "c:/work/t.txt";`

Funcțiile standard pentru operații cu fișiere

Închiderea unui fișier:

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

- întoarce **0** la închiderea normală și **EOF** la producerea unui incident
- fișierele standard nu se închid de către programator
- ! pentru un fișier de ieșire se scriu datele rămase nescrise din buffer în fișier, deci, operația de închidere este obligatorie
- în cazul unui fișier de intrare, datele necitite din bufferul de intrare sunt abandonate
- se eliberează bufferele alocate
- se întrerupe conexiunea pointer – fișier.

Exemplu

```
#include <stdio.h>

int main ( ) {
    FILE * f; // pentru referire la fișier
    // deschide un fișier text pentru citire
    f = fopen ( "t.txt", "rt" );
    (f==NULL)? puts("Fișier negasit"):: puts( " Fișier gasit");
    if (f)        // dacă fișier existent
    fclose(f); // închide fișier
    f=NULL;
    return 0;
}
```

Funcțiile standard pentru operații cu fișiere

Operații de intrare / ieșire:

Tip fișier	Conversie (formatare)	Unitate transferată	Funcții folosite
text	fără	caracter	fgetc() fputc()
		linie	fgets() fputs()
	cu	linii	fscanf() fprintf()
binar	fără	articol	fread() fwrite()

Operații de intrare / ieșire la nivel de caracter

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

întoarce următorul caracter citit din fișierul cu pointerul **pf** sau EOF dacă s-a citit sfârșit de fișier.

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

scrie caracterul **c** în fișierul cu pointerul **pf**, întoarce **c** sau EOF la eroare.

Exemplu : Scrieți un program care copiază un fișier. Numele celor două fișiere (sursă și destinație) sunt citite de la tastatură.

Copierea se face caracter cu caracter.

Exemplu:

```
void copiere1(FILE *, FILE*);  
int main() {  
    char numes[12], numed[12];  
    gets(numes);  
    gets(numed);  
    FILE* s = fopen(numes, "r");  
    FILE* d = fopen(numed, "w");  
    copiere1(d, s);  
    fclose(s);  
    fclose(d);  
}
```

```
void copiere1(FILE*d, FILE *s)  
{  
    int c;  
    while ((c=fgetc(s)) != EOF)  
        fputc(c, d);  
}
```

Operații de intrare / ieșire pentru șiruri de caractere

char *fgets(char *s, int n, FILE *pf);

- citește caractere din fișierul cu pointerul **pf**, până la întâlnirea primului caracter **'\n'** (cel mult **n-1** caractere) în tabloul **s**; pune la sfârșit **'\n'** și **'\0'**
- întoarce **s** sau **NULL** la întâlnire sfârșit de fișier sau la eroare

int fputs(char *s, FILE *pf);

- copiază șirul în fișierul de ieșire **pf**
- nu copiază terminatorul de șir **'\0'**
- întoarce un rezultat nenegativ (numărul de caractere scrise în fișier) sau **EOF** la producerea unei erori.

Exemplu: Să se scrie o funcție care copiază un fișier folosind funcții orientate pe șiruri de caractere.

```
#define MAX 100
void copiere2(FILE *d, FILE *s) {
    char linie[MAX];
    while(fgets(linie, MAX, s))
        fputs(linie, d);
}
```

Operații de intrare / ieșire cu format

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

- transferă în fișierul specificat valorile expresiilor convertite, potrivit formatului în caractere
- întoarce numărul de caractere scrise sau o valoare negativă dacă s-a produs o eroare.

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

- se citesc date din fișierul **pf**, sub controlul formatului, inițializându-se variabilele din listă
- funcția întoarce numărul de câmpuri citite sau **EOF**, în caz de producere a unui incident la citire, sau întâlnire a marcajului de sfârșit de fișier.

Funcțiile sunt utilizate pentru citire/scriere în mod text și sunt asemănătoare cu printf/scanf (diferența fiind că trebuie dat pointerul la fișier ca prim parametru).

Operații de intrare / ieșire în modul de acces binar

- sunt operații de transfer (citiri / scrieri) fără conversii
- se fac la nivel de articol
- poziția în fișier este actualizată după fiecare citire / scriere

`unsigned fread(void *zona, unsigned la, unsigned na, FILE*pf) ;`

- citește cel mult **na** articole, de lungime **la** fiecare, din fișierul **pf** în **zona** ;
- întoarce numărul de înregistrări citite sau 0 în caz de eroare sau sfârșit de fișier.

Operații de intrare / ieșire în modul de acces binar

```
unsigned fwrite(void *zona,unsigned la,unsigned na,FILE*pf) ;
```

- scrie **na** articole de lungime **la** din **zona** în fișierul **pf**
- întoarce numărul de articole scrise.

Pentru a copia un fișier binar (sau text) folosind funcțiile **fread()** și **fwrite()** vom considera lungimea articolului 1 octet.

Exemplu: Copierea unui fisier în mod binar

```
void copiere(FILE * d, FILE * s)
{
    int noc; // numar de octeti cititi
    char zona[MAX];
    while( (noc=fread(zona,1,MAX,s)) > 0)
        fwrite(zona, 1, noc, d);
}
```

Poziționarea în fișier - acces direct la datele dintr-un fișier

```
int fseek(FILE *pf, long depl, int orig);
```

- modifică poziția curentă în fișierul cu pointerul **pf** cu **depl** octeți relativ la cel de-al treilea parametru **orig**, după cum urmează:
- față de începutul fișierului, dacă **orig = 0** (sau **SEEK_SET**)
- față de poziția curentă, dacă **orig = 1** (sau **SEEK_CUR**)
- față de sfârșitul fișierului, dacă **orig = 2** (sau **SEEK_END**)
- întoarce rezultatul 0 pentru o poziționare corectă și diferit de 0, în caz de eroare.

Poziționarea în fișier - acces direct la datele dintr-un fișier

`void rewind(FILE *pf) ;`

- realizează o poziționare la începutul fișierului fiind echivalent cu:

`-fseek(pf, 0, SEEK_SET) ;`

`-fseek(fp, -1, SEEK_CUR) ;` - poziționarea la octetul precedent

-poziționarea la sfârșitul fișierului: **`fseek(fp, 0, SEEK_END) ;`**

`long ftell(FILE *pf) ;`

- întoarce poziția curentă în fișier, exprimată prin numărul de octeți față de începutul fișierului pentru fișierele binare.

Exemplu: Dimensiunea unui fișier

```
long fileSize(FILE *pf)
{ long pozv, noct;
  pozv = ftell(pf); //salvare poz curenta
  //pozitionare la sfarsit
  fseek(pf, 0L, SEEK_END);
  noct = ftell(pf); // nr de octeti
  //revenirea la pozitia veche
  fseek(pf, pozv, SEEK_SET);
  return noct;
}
```

Tratarea erorilor. Ștergerea și redenumirea fișierului

int feof(FILE *pf) ;

- întoarce o valoare diferită de 0, dacă s-a detectat marcajul de sfârșit de fișier (EOF);

int ferror(FILE *pf) ;

- întoarce o valoare diferită de 0, dacă s-a detectat o eroare în cursul operației de intrare / ieșire;

int remove(const char *filename) ;

- șterge fișierul cu denumirea stocată la adresa **filename** – întoarce 0 la ștergerea reușită;

int rename(const char *old, const char *new) ;

- schimbă denumirea fișierului – întoarce 0 la schimbarea reușită.

Exemplu: Scrierea unui fisier de la sfârșit

```
// . . .  
char    file_name[MAXSTRING] ;  
int      c ;  
FILE     *ifp ;  
  
fprintf(stdout, "\nInput a file name:  ") ;  
scanf("%s", file_name) ;  
ifp = fopen(file_name, "rb") ;  
fseek(ifp, 0, 2) ;    // pozitionare la sfarsit  
fseek(ifp, -1, 1) ;  // pozitionare la ultimul octet  
while (ftell(ifp) > 0) {  
    c = fgetc(ifp) ;  
    putchar(c) ;  
    fseek(ifp, -2, 1) ; //octetul anterior  
}
```

Probleme propuse spre rezolvare folosind funcțiile pentru operații cu fișiere

1. Să se elaboreze un program care scrie sub formă de litere mici caracterele dintr-un fișier în alt fișier.
2. Să se scrie un program care introduce informația despre un grup de studenți de la tastatură, scrie informația despre studenți într-un fișier și apoi citește informația despre studenți din fișier și afișează pe ecran folosind funcții și memoria dinamică.
3. Să se scrie un program care determină numărul de linii și numărul de caractere din fișier.
4. Să se scrie un program în C pentru evidența studenților de la facultate. Despre un student se cunosc următoarele informații: numele, anul și nota medie. Programul trebuie să permită, printr-un meniu interactiv, următoarele operații (prin funcții cu transfer de parametri – parametrii obligatorii ai funcțiilor sunt adresa tabloului de studenți și numărul curent de studenți): introducerea informației despre studenți de la tastatură și din fișier, scrierea informației despre studenți pe ecran și în fișier, căutarea unui student după nume, sortarea studenților după nume.

Tutoriale online pentru tema prelegerii:

1. <https://ocw.cs.pub.ro/courses/programare/laboratoare/lab12>
2. <http://andrei.clubcisco.ro/cursuri/1pc/curs/1/Curs%2010-11%20Doc.pdf>
3. https://staff.fmi.uvt.ro/~victoria.iordan/Programare_MI/Curs12.pdf
4. <http://www.phys.ubbcluj.ro/~vasile.chis/cursuri/info/c08ppt.pdf>
5. http://www.cs.ucv.ro/staff/gmarian/Programare/cap9_Functii_IE.pdf
6. http://www.aut.upt.ro/~rraul/PC/2009-2010/PC_lab10.pdf
7. <https://igotopia.ro/cum-citesti-si-cum-scrii-fisiere-folosind-limbajul-c/>
8. <https://info64.ro/Fisiere/>
9. <http://andrei.clubcisco.ro/cursuri/1pc/co/curs10.pdf>
10. <http://www.cplusplus.com/reference/cstdio/>



FACULTATEA
CALCULATOARE, INFORMATICĂ
ȘI MICROELECTRONICĂ

FCIM

VĂ MULȚUMESC PENTRU ATENȚIE!

MULTĂ SĂNĂTATE ȘI SUCCESE!