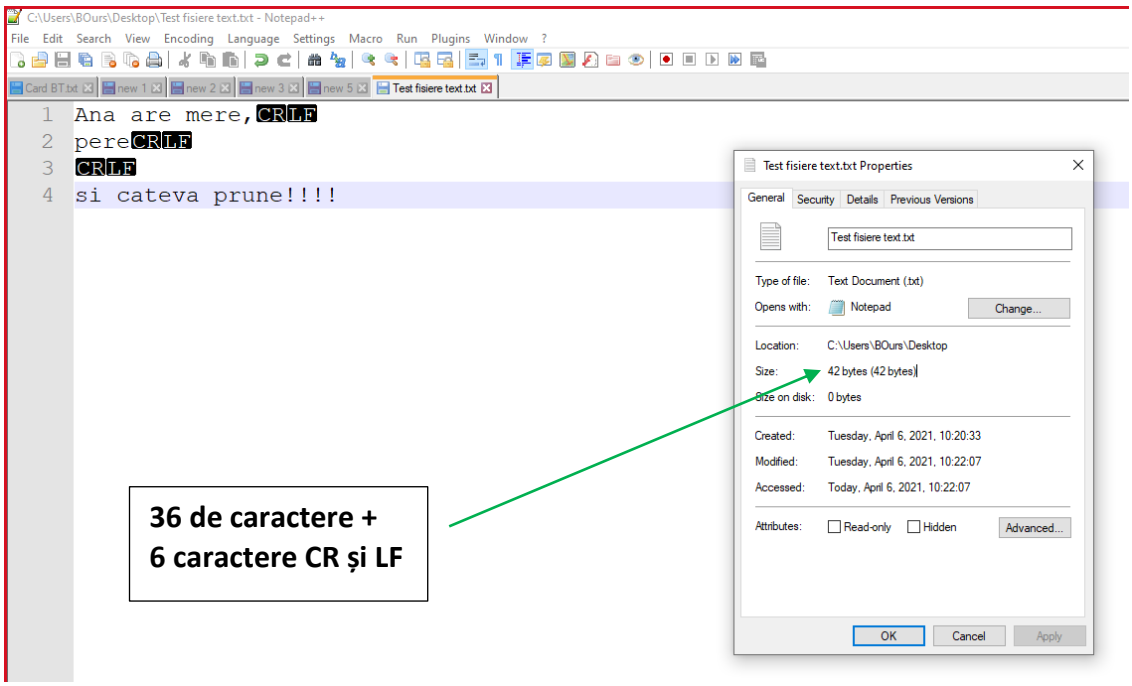
CURS 09 – PP
FIȘIERE TEXT

Fișier = un șir de octeți memorat pe un suport de memorie externă (HDD, CD, DVD, stick USB etc.).
Fișier = asigură persistența datelor

Fișiere:

1. Fișiere binare – un octet nu are o semnificație specială
2. Fișiere text – un octet reprezintă codul ASCII al unui caracter, iar textul este împărțit pe linii folosind caracterele CR+LF ('\r' + '\n') sub sistemul de operare Windows sau caracterul LF sub sistemele de operare Linux/UNIX/MacOS (versiunile mai noi) sau CR sub sistemul de operare MacOS (versiunile mai vechi).

CR = Carriage Return = cod ASCII 13
LF = Line Feed = cod ASCII 10



În momentul prelucrării unui fișier text combinația CR+LF este transformată automat într-un singur caracter LF!!!

Flux = o modalitate de transmitere a informației, în format binar, de la o sursă către o destinație.

Flux de intrare/citire = sursa este un fișier, iar destinația este memoria internă
Flux de ieșire/scriere = sursa este memoria internă, iar destinația este un fișier

Nivelurile unui fișier:

- *nivelul fizic* – calea fișierului din memoria externă
- *nivelul logic* – o variabilă de tip `FILE*` care se va utiliza în program pentru a manipula fișierul respectiv

Legătura dintre cele două niveluri se asigură prin intermediul unui flux!!!

Operații specifice fișierelor text

Tipul de date FILE este o structură care permite memorarea mai multor informații despre un fișier într-un mod transparent pentru programator.

Astfel, variabila de tip FILE* pe care o vom utiliza în program va conține informații referitoare la numărul de octeți care mai pot fi citiți din fișier și poziția curentă în fișierul respectiv (*pointerul de fișier*), sub forma numărului de ordine al caracterului curent (caracterele din fișier sunt numerotate începând de la 0).

Fișierele text sunt fișiere cu acces secvențial, adică se pot parcurge doar caracter cu caracter, începând de la primul. Practic, în momentul deschiderii unui fișier text pointerul de fișier este poziționat pe primul caracter și apoi, după fiecare operație de citire sau scriere, valoarea pointerului de fișier este actualizată.

1. Deschiderea unui fișier text:

- asocierea unui fișier din memoria externă cu o variabilă de tip FILE* in memoria internă (stabilirea unui flux de intrare sau de ieșire):

```
FILE* fopen(char* cale_f, char* mod)
```

- **`cale_f`** = calea fișierului din memoria externă
- **`mod`** = modalitatea de deschidere a fișierului text:
    - `"r"` -> citire (read) -> fișierul trebuie să existe
    - `"w"` -> scriere (write) -> dacă există fișierul respectiv, atunci el va fi șters și înlocuit cu unul vid

- "wx" -> scriere exclusivă (exclusive write) -> dacă există fișierul respectiv, atunci va apărea o eroare și fișierul existent nu va fi șters
- "a" -> adăugare la sfârșitul fișierului (append) -> dacă nu există fișierul respectiv, se va crea unul vid și apoi se va scrie în el
- "ax" -> adăugare exclusivă la sfârșitul fișierului (exclusive append) -> dacă există fișierul respectiv, atunci va apărea o eroare și nu se va scrie la sfârșitul fișierului existent

- Funcția fopen returnează pointerul NULL în cazul unei erori!

Exemplu:

```
FILE *f = fopen("C:\\Test\\exemplu.txt", "r");
FILE *f = fopen("exemplu.txt", "w");
```

2. Prelucrarea conținutului unui fișier text (operații de citire/scriere):

a) la nivel de caracter

- citire: **int fgetc(FILE *f)** - furnizează codul ASCII al caracterului curent din fișier sau constanta EOF (End-Of-File) = -1 dacă apare o eroare la citire sau s-a ajuns la sfârșitul fișierului
- **Exemplu:** afișarea unui fișier text pe ecran

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *fin;
    int c;

    fin = fopen("exemplu.txt", "r");

    if(fin == NULL)
    {
        printf("\nEroare la deschiderea fisierului de
                                                intrare!\n");
        return 0;
    }

    //EOF = -1
    while((c = fgetc(fin)) != EOF)
        printf("%d -> %c\n", c, c);
```

```
    fclose(fin);

    return 0;
}
```

- **scriere:** **`int fputc(char c, FILE *f)`** - furnizează codul ASCII al caracterului scris în fișier sau constanta EOF = -1 dacă apare o eroare la scriere
- **Exemplu:** scrierea unui șir de caractere într-un fișier text

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    FILE *fout;
    char sir[1001];
    int i;

    printf("Sirul de caractere: ");
    fgets(sir, 1001, stdin);

    fout = fopen("C:\\Test\\sir.txt", "w");

    if(fout == NULL)
    {
        printf("\nEroare la deschiderea fisierului
                                                de iesire!\n");
        return 0;
    }

    for(i = 0; i < strlen(sir); i++)
        fputc(sir[i], fout);

    fclose(fout);

    return 0;
}
```

Observație: Funcțiile `fgetc` și `fputc` modifică pointerul de fișier cu 1 (un caracter / un octet) la fiecare apel.

b) la nivel de linie

- **citire:** **`char* fgets(char* sir, int max, FILE *f)`** - furnizează linia curentă din fișier (inclusiv cu '\n' la sfârșitul său!!!) sub forma unui șir de caractere sau constanta NULL dacă apare o eroare la citire sau s-a ajuns la sfârșitul fișierului. Funcția va citi **`max`** caractere de pe linia curentă, după care va adăuga un caracter `'\0'` la sfârșitul șirului.
- **Exemplu:** afișarea unui fișier text pe ecran, linie cu linie

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *fin;
    char linie[1001];
    int lcrt;

    fin = fopen("exemplu.txt", "r");

    lcrt = 0;
    while(fgets(linie, 1001, fin))
    {
        if(linie[strlen(linie) - 1] == '\n')
            linie[strlen(linie) - 1] = '\0';
        printf("Linia %d: %s\n", lcrt++, linie);
    }


    fclose(fin);

    return 0;
}
```

- **scriere:** **`int fputs(char* sir, FILE *f)`** - furnizează o valoare nenulă în caz de succes sau constanta EOF = -1 dacă apare o eroare la scriere
- **Exemplu:** scrierea unor șiruri de caractere într-un fișier text

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    FILE *fout;
    char sir[1001];
```

```
    int i, n;

    fout = fopen("exemplu.txt", "w");

    printf("Numarul de siruri: ");
    scanf("%d", &n);
    scanf("%*c");
    for(i = 0; i < n; i++)
    {
        printf("Sirul %d: ", i+1);
        fgets(sir, 1001, stdin);
        if(sir[strlen(sir) - 1] == '\n')
            sir[strlen(sir) - 1] = '\0';

        fputs(sir, fout);
        fputs("\n", fout);
    }

    fclose(fout);

    return 0;
}
```

Observație: Constantele `stdin` și `stdout`, ambele de tip `FILE*`, reprezintă tastatura, respectiv monitorul, și pot fi utilizate în orice funcție care lucrează cu fișiere text pentru a citi de la tastatură sau pentru a scrie pe monitor!!!

Exemplu: copierea unui fișier text în alt fișier text (numele celor două fișiere se citesc de la tastatură)

```
#include <stdio.h>
int main ()
{
    char nume_fin[101], nume_fout[101];
    FILE *fin, *fout;
    char linie[1001];

    printf("Calea fisierului sursa: ");
    fgets(nume_fin, 101, stdin);
    if(nume_fin[strlen(nume_fin) - 1] == '\n')
        nume_fin[strlen(nume_fin) - 1] = '\0';

    fin = fopen(nume_fin, "r");

    if(fin == NULL)
    {
        printf("\nFisier sursa inexistent!\n");
        return 0;
    }
```

```
    printf("Calea fisierului destinatie: ");
    fgets(nume_fout, 101, stdin);
    if(nume_fout[strlen(nume_fout) - 1] == '\n')
        nume_fout[strlen(nume_fout) - 1] = '\0';

    fout = fopen(nume_fout, "a");

    if(fout == NULL)
    {
        printf("\nEroare la crearea fisierului destinatie!\n");
        return 0;
    }

    while(fgets(linie, 101, fin) != NULL)
        fputs(linie, fout);

    fclose(fin);
    fclose(fout);

    return 0;
}
```

Observație: Funcțiile `fgets` și `fputs` modifică pointerul de fișier cu numărul caracterelor din șirul citit sau scris la fiecare apel.

c) la nivel de date formatate

- citire: **`int fscanf(FILE* fin,...)`** – furnizează numărul valorilor citite corect din fișier conform specificatorilor de format
- scriere: **`int fprintf(FILE* fout,...)`** – furnizează numărul valorilor scrise corect în fișier conform specificatorilor de format

- **Exemplu**: calculul sumei unor numere dintr-un fișier text

  Varianta 1: pe prima linie din fișierul text este scris numărul valorilor din fișierul text respectiv

```
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fin, *fout;
    int i, n, x, suma;
    char numef[101];

    printf("Numele fisierului de intrare: ");
    fgets(numef, 101, stdin);
```

```
    if(numef[strlen(numef) - 1] == '\n')
        numef[strlen(numef) - 1] = '\0';

    fin = fopen(numef, "r");

    if(fin == NULL)
    {
        printf("Eroare la deschiderea fisierului %s!\n",
                                                        numef);
        return 0;
    }

    //citim numarul de valori din fisier
    //de pe prima linie a sa
    fscanf(fin, "%d", &n);

    //citim cele n numere si calculam suma lor
    suma = 0;
    for(i = 0; i < n; i++)
    {
        fscanf(fin, "%d", &x);
        suma = suma + x;
    }

    fclose(fin);

    fout = fopen("suma.txt", "w");

    fprintf(fout, "Suma numerelor din fisierul %s: %d\n",
                                                numef, suma);

    fclose(fout);

    return 0;
}
```

Varianta 2: nu cunoaștem numărul valorilor din fișierul text respectiv

```
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fin, *fout;
    int x, suma;
    char numef[101];

    printf("Numele fisierului de intrare: ");
    fgets(numef, 101, stdin);
```

```
    if(numef[strlen(numef) - 1] == '\n')
        numef[strlen(numef) - 1] = '\0';

    fin = fopen(numef, "r");

    if(fin == NULL)
    {
        printf("Eroare la deschiderea fisierului %s!\n",
                                                    numef);
        return 0;
    }

    //citim toate numerele din fisier, unul cate unul
    suma = 0;
    while(fscanf(fin, "%d", &x) == 1)
        suma = suma + x;

    fclose(fin);

    fout = fopen("suma.txt", "w");

    fprintf(fout, "Suma numerelor din fisierul %s: %d\n",
                                                numef, suma);

    fclose(fout);

    return 0;
}
```

Varianta 3: scriem suma numerelor la sfârșitul fișierului inițial

```
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fin, *fout;
    int x, suma;
    char numef[101];

    printf("Numele fisierului de intrare: ");
    fgets(numef, 101, stdin);
    if(numef[strlen(numef) - 1] == '\n')
        numef[strlen(numef) - 1] = '\0';

    fin = fopen(numef, "r");

    if(fin == NULL)
    {
```

```
        printf("Eroare la deschiderea fisierului %s!\n",
                                                    numef);
        return 0;
    }

    //citim toate numerele din fisier, unul cate unul
    suma = 0;
    while(fscanf(fin, "%d", &x) == 1)
        suma = suma + x;
    fclose(fin);

    fout = fopen(numef, "a");
    fprintf(fout, "\nSuma numerelor din fisierul %s: %d\n",
                                            numef, suma);
    fclose(fout);

    return 0;
}
```

- **Exemplu**: calculul sumelor numerelor aflate pe fiecare linie dintr-un fișier text

  Folosim funcția `strtok` pentru a extrage numerele de pe fiecare linie:

```
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fin, *fout;
    int aux, suma;
    char numef[101], linie[1001], *numar;

    printf("Numele fisierului de intrare: ");
    fgets(numef, 101, stdin);
    if(numef[strlen(numef) - 1] == '\n')
        numef[strlen(numef) - 1] = '\0';

    fin = fopen(numef, "r");
    fout = fopen("sume_linii.txt", "w");

    //citim continutul fisierului de intrare linie cu linie
    while(fgets(linie, 1001, fin) != NULL)
    {
        suma = 0;
        //extrag fiecare numar de pe linia curenta
        numar = strtok(linie, " \n");
        while(numar != NULL)
```

```
            {
                sscanf(numar, "%d", &aux);
                suma = suma + aux;
                numar = strtok(NULL, " \n");
            }
            fprintf(fout, "%d\n", suma);
        }

        fclose(fin);
        fclose(fout);

        return 0;
    }
```

3. Închiderea unui fișie text:

int fclose(FILE* f)

Funcția returnează valoarea 0 dacă fișierul a fost închis cu succes sau EOF = -1 în caz de eroare!