Laborator 1

Recapitulare

Gestiunea fişierelor în C
Funcţii de intrare/ieşire de nivel superior

În cazul funcțiilor de intrare / ieşire de nivel superior, schimbul de date cu dispozitivele fizice de intrare / ieşire (tastatura, ecranul, imprimanta, hard-disk-ul) se face prin intermediul unor dispozitive logice numite "*stream-uri*" (*fluxuri*). Fiecărui dispozitiv fizic i se asociază un stream.

Pentru a lucra cu fişiere trebuie introdusă următoarea linie în cadrul fiecărui program:

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

Inițierea unui transfer de informație va fi precedată de crearea unui stream. Se declară o variabilă de tip structură `FILE`, urmată de alocarea unei zone de memorie tampon prin intermediul căreia se vor efectua transferurile de date.

Există două tipuri de stream-uri:

1. *stream de tip text*: informația este interpretată ca un flux de caractere. Se transferă secvențe de caractere organizate în blocuri separate prin `0Ah` (caracterul "new line"). Acest separator este transformat în `0Dh`, `0Ah` în momentul scrierii textului pe disc.
2. *stream de tip binar*: informația este interpretată ca un flux de octeți fără nicio structură.

La crearea fiecărui program, sunt creați 5 identificatori de tip constante pointer la `FILE`:

- `stdin` – stream de tip text, asociat cu intrarea standard, tastatura;
- `stdout` – stream de tip text, asociat cu ieşirea standard, ecranul;
- `stderr` – stream de tip text, asociat cu ieşirea standard pentru erori, ecranul;
- `stdprn` – stream de tip binar, asociat cu imprimanta;
- `stdaux` – intrare/ieşire auxiliară.

1. Închidere/deschidere

Deschiderea unui fişier se face astfel:

```
FILE *fopen(const char *cale, const char *acces)
```

`cale` – numele fişierului şi eventual calea către acesta pe disc

`acces` – şir de caractere ce precizează scopul deschiderii

`r – citire` : Deschide un fişier doar pentru citire.

`w – scriere` : Deschide un fişier doar pentru scriere. Dacă fişierul cu acest nume nu există atunci el va fi creat. Dacă, în schimb, există un fişier cu numele specificat, atunci conţinutul fişierului este şters iar acesta se comportă ca un fişier proaspăt creat.

`a – adaugare` : Deschide un fişier doar pentru adăugare (scriere la sfârşit). Dacă fişierul nu există atunci este creat un fişier vid.

`r+` – Deschide un fişier pentru operaţiile de citire şi scriere. Fişierul trebuie să fi fost creat anterior momentului încercării de deschidere.

`w+` – Crează un fişier vid asupra căruia pot fi aplicate operaţii atât de citire cât şi de scriere. În situaţia în care există deja un fişier cu acelaşi nume, atunci conţinutul acestuia este şters.

`a+` – Deschide un fişier pentru operaţii de citire şi adăugare la sfârşit. Fişierul este creat dacă nu există deja. Operaţiile de scriere sunt efectuate la sfârşitul fişierului. Pointerul asociat fişierului ce indică poziţia unde se va realiza următoarea operaţie de scriere sau de citire poate fi mutat oriunde în cadrul fişierului. Orice operaţie de scriere îl va muta însă la sfârşitul fişierului.

`t` – fişier de tip text

`b` – fişier de tip binar

iar închiderea

```
int fclose(FILE *fp)
```

2. Intrări/ieşiri cu conversie de format

`int fprintf(FILE *fp,const char *format,...)` – funcţia este identică cu *printf* cu deosebirea că ea scrie într-un fişier.

`int fscanf(FILE *fp,const char *format,...)` – realizeaza citirea cu format dintr-un fişier.

3. Prelucrări la nivel de caracter

`int fgetc(FILE *fin)` – întoarce următorul caracter din `fin` ca un *unsigned char* convertit la *int*, sau EOF dacă s-a întâlnit sfârşitul de fişier sau în caz de eroare.

`int fputc(int c, FILE *fout)` – se scrie caracterul `c` în fişierul `fout`. Întoarce valoarea scrisă sau `EOF` în caz de eşec.

`char *fgets(char *s, int n, FILE *fin)` – se citesc cel mult n-1 caractere din fişierul `fin` în variabila `s`, citirea oprindu-se la întâlnirea caracterului '`\n`'. La sfârşitul şirului se pune automat '`\0`'.

`char *fputs(char *s, FILE *fout)` – se scrie conţinutul şirul `s` în fişierul reprezentat de `fout` returnându-se ultimul caracter scris sau `EOF` în caz de eroare.

4. Prelucrare pe înregistrări

Aceste funcţii citesc/scriu din/în fişiere fără nici o conversie. Se folosesc cu modul de acces binar. După citire/scrire poziţia în fişier este actualizată. Înregistrările pot conţine orice fel de date. În fişier, datele vor apare în reprezentarea internă de pe calculator.

```
size_t fread(void *ptr, size_t size, size_t nrec, FILE *fin)
```

Se citesc cel mult *nrec* înregistrări de lungime *size* fiecare într-o zonă de memorie reprezentată de *ptr*.

`ptr` – adresa zonei de memorie unde se vor citi datele

`size` – lungimea în octeţi a fiecărui bloc transferat

`nrec` – numărul de blocuri care se transferă

`fin` – pointerul la fişierul din care se realizează citirea.

```
size_t fwrite(const void *ptr, size_t size, size_t nrec, FILE *fout)
```

realizează scrierea a *nrec* înregistrări.

`ptr` – adresa zonei de memorie de unde se vor

`size` – lungimea în octeţi a fiecărui bloc transferat

`nrec` – numărul de blocuri care se transferă

`fout` – pointerul la fişierul în care se realizează scrierea.

5. Poziţionare într-un fişier

```
int fseek(FILE *fp, long depl, int origine)
```

`origine` poate avea una din următoarele valori:

SEEK_SET - începutul fişierului

SEEK_END - sfârşitul fişierului

SEEK_CUR - poziţia curentă

Funcţia *fseek* poziţionează pointerul de fişier la distanţa *depl* octeţi faţă de o poziţie de referinţă aleasă.

`int ftell(FILE *fp)` - întoarce poziţia curentă din fişier sau `-1L` în caz de eroare.

`int rewind(FILE *fp)` - poziţionare la începutul fişierului.

`int feof(FILE *fp)` - întoarce o valoare diferită de 0 dacă s-a ajuns la sfârşitul fişierului.

`int ferror(FILE *fp)` - întoarce o valoare diferită de 0 dacă s-a obţinut o eroare la vreo operaţie.

Exemplu 1: Să se scrie un program de copiere a unui fişier în alt fişier, folosindu-se numai funcţii de prelucrare pe caractere.

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

```
int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  int c;

  if (argc != 3) {
    fprintf(stderr, "Utilizare : copy sursa dest \n");
    exit(1);
  }

  if ((fin = fopen(argv[1], "rt")) == NULL) {
    fprintf(stderr, "Eroare deschidere fisier %s \n", argv[1]);
    exit(1);
  }
  if ((fout = fopen(argv[2], "wt")) == NULL) {
    fprintf(stderr, "Eroare deschidere fisier %s \n", argv[2]);
    exit(1);
  }

  c = getc(fin);
  while (c != EOF) {
    fputc(c, fout);
    c = fgetc(fin);
  }

  fclose(fin);
  fclose(fout);

  return 0;
}
```

Exemplu 2: Aceeaşi cerinţă ca la exemplul anterior, dar se vor folosi numai funcţii pe şiruri de caractere.

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

#define BUFLEN 1000

char buffer[BUFLEN];

int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  char *sir;

  if (argc != 3) {
    fprintf(stderr, "Utilizare : copy sursa dest \n");
    exit(1);
  }

  if ((fin = fopen(argv[1], "rt")) == NULL) {
    fprintf(stderr, "Eroare deschidere fisier %s \n", argv[1]);
    exit(1);
  }
  if ((fout = fopen(argv[2], "wt")) == NULL) {
    fprintf(stderr, "Eroare deschidere fisier %s \n", argv[2]);
    exit(1);
  }
```

```
    sir = fgets(buffer, BUFLEN, fin);
    while (sir != NULL) {
      fputs(buffer, fout);
      sir = fgets(buffer, BUFLEN, fin);
    }

    fclose(fin);
    fclose(fout);

    return 0;
  }
```

Exemplu 3: Aceeaşi cerinţă, dar se vor folosi numai funcţii la nivel de înregistrare.

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

  #define BUFLEN 1000

  char buffer[BUFLEN];

  int main(int argc, char *argv[]) {
    FILE *fin, *fout;
    size_t n;

    if (argc != 3) {
      fprintf(stderr, "Utilizare : copy sursa dest \n");
      exit(1);
    }

    if ((fin = fopen(argv[1], "rt")) == NULL) {
      fprintf(stderr, "Eroare deschidere fisier %s \n", argv[1]);
      exit(1);
    }
    if ((fout = fopen(argv[2], "wt")) == NULL) {
      fprintf(stderr, "Eroare deschidere fisier %s \n", argv[2]);
      exit(1);
    }

    n = fread(buffer, 1, BUFLEN, fin);
    while (n > 0) {
      fwrite(buffer, 1, n, fout);
      n = fread(buffer, 1, BUFLEN, fin);
    }

    fclose(fin);
    fclose(fout);

    return 0;
  }
```

Exemplu 4: Scrieți un program care să afişeze la imprimantă conținutul unui fişier text.

```
#include <stdio.h>

FILE *fin;
FILE *print;
char c;
```

```
int main() {
  char fname[12];

  printf("Numele fisierului: ");
  scanf("%s", fname);

  fin = fopen(fname, "r");
  print = fopen("PRN", "w");

  do {
     c = getc(fin);
     if (c != EOF) {
       putchar(c);
       putc(c, print);
     }
  } while (c != EOF);

  fclose(fin);
  fclose(print);

  return 0;
}
```