Operaţii cu fişiere. Aplicaţii folosind fişiere.

Resposabili:Responsabili:

- Dorinel Filip (CA 2016-2020) [mailto:ion_dorinel.filip@cti.pub.ro]
- Darius Neațu (CA 2019-2020) [mailto:neatudarius@gmail.com]
- Mihaela Vasile (2015) [mailto:mihaela.vasile@gmail.com]

Probleme [https://we.tl/t-U1WwFpnaFU]

Ultima modificare: 09.12.2018

Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- lucreze cu fişiere text (deschidere, închidere, citire, scriere)
- înteleaga un fişier binar şi să lucreze cu el;
- să se poziţioneze in interiorul unui fişier;
- poată determina poziţia în cadrul unui fişier;
- înteleagă diferenţa între organizarea internă a fişierelor pe sistemele de operare Linux şi Windows.

Noţiuni teoretice

Introducere

Un fişier este o structură dinamică, situată în memoria secundară (pe disk-uri). Limbajul C permite operarea cu fişiere:

- de **tip text** - un astfel de fişier conţine o succesiune de **linii**, separate prin new line ('\n')
- de **tip binar** - un astfel de fişier conţine o succesiune de octeti, **fără nici o structură**.

Prelucrarea unui fişier presupune asocierea acestuia cu un canal de I/E (numit flux sau stream). Există trei canale predefinite, care se deschid automat la lansarea unui program:

- **stdin** - fişier de intrare, text, este intrarea standard - tastatura
- **stdout** - fişier de iesire, text, este ieşirea standard - ecranul monitorului.
- **stderr** – fişier de iesire, text, este ieşirea standard unde sunt scris mesajele de eroare - ecran.

Pentru a prelucra un fişier, trebuie parcurse următoarele etape:

- **se defineşte** o variabilă de tip **FILE*** pentru accesarea fişierului; **FILE** este un tip structură definit în <stdio.h>, care conţine informaţii referitoare la fişier şi la tamponul de transfer de date între memoria centrală şi fişier (adresa, lungimea tamponului, modul de utilizare a fişierului, indicator de sfârsit, de poziţie în fişier). Puteți citi mai multe aici [http://stackoverflow.com/questions/5672746/what-exactly-is-the-file-keyword-in-c].
- **se deschide fişieru**l pentru un anumit **mod de acces**, folosind funcţia de bibliotecă **fopen**, care realizează şi asocierea între variabila fişier şi numele extern al fişierului
- **se prelucrează fişierul** în citire/scriere cu **funcţiile specifice**
- **se închide fişierul** folosind funcţia de bibliotecă **fclose**

Funcții

Mai jos se prezintă restul funcţiilor de prelucrare a fişierelor. Pentru documentația oficială puteți citi aici [http://www.cplusplus.com/reference/cstdio/].

fopen

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

deschide fişierul cu numele **filename** pentru acces de tip **mod**.

Returnează **pointer la fişier** sau **NULL** dacă fişierul nu poate fi deschis; valoarea returnată este memorată în variabila fişier, care a fost declarată pentru accesarea lui.

Modul de deschidere poate fi:

- "**r**" - **readonly** , este permisă doar citirea dintr-un fişier existent
- "**w**" - **write**, crează un nou fişier, sau dacă există deja, distruge vechiul continut
- "**a**" - **append**, deschide pentru scriere un fişier existent (scrierea se va face în continuarea

informaţiei deja existente în fişier, deci pointerul de acces se plasează la sfârşitul fişierului)

- "**+**" - permite scrierea şi citirea - **actualizare** (ex: "r+", "w+", "a+"). Între read şi write trebuie repoziţionat cursorul de acces printr-un apel la **fseek**.

- "**b**" - specifică fişier de tip **binar**
- "**t**" - specifică fişier de tip **text** (implicit), la care se face automat conversia CR-LF("\n\f") în sau din CR ('\n').

fclose

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

închide fişierul asociat cu variabila **pFile** şi eliberează zona tampon; returnează 0 la succes, EOF (end of file) la eroare

fseek

```
int fseek(FILE *pFile, long offset, int whence);
```

repoziţionează pointerul asociat fişierului **pFile**; offset - numărul de octeţi între poziţia dată de whence şi noua poziţie.

whence - are una din cele trei valori posibile:

- SEEK_SET = 0 - Căutarea se face de la începutul fişierului
- SEEK_CUR = 1 - Căutare din poziţia curentă
- SEEK_END = 2 - Căutare de la sfârşitul fişierului

ftell

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

întoarce poziţia curentă în cadrul fișierului asociat cu pFile.

fgetpos

```
int fgetpos(FILE *pFile, fpos_t *ptr);
```

această funcţie **memorează poziţia curentă** în variabila ptr în cadrul fişierului asociat cu pFile (ptr va putea fi folosit ulterior cu funcţia fsetpos).

fsetpos

```
int fsetpos(FILE *pFile, const fpos_t *ptr);
```

această funcţie **setează poziţia curentă** în fişierul asociat cu pFile la valoarea ptr, obţinută anterior prin funcţia fgetpos.

feof

```
int feof(FILE *fis);
```

returnează 0 dacă nu s-a detectat **sfârşit de fişier** la ultima operaţie de citire, respectiv o valoare nenulă (adevărată) pentru sfârşit de fişier.

freopen

```
FILE* freopen(const char *filename, const char *mode, FILE *fp);
```

se închide fişierul fp, se deschide fişierul cu numele filename în modul mode şi acesta se asociază la fp; se întoarce fp sau NULL în caz de eroare.

fflush

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

Această funcţie se utilizează pentru fişierele deschise pentru scriere şi are ca efect scrierea în fişier a datelor din bufferul asociat acestuia, care înca nu au fost puse în fişier.

Citirea şi scrierea în/din fişiere

Citirea/scrierea în fişiere se poate face în două moduri (în funcţie de tipul fişierului): în mod text sau în mod binar. Principalele diferenţe dintre cele două moduri sunt:

- în modul text, la sfarsitul fişierului se pune un caracter suplimentar, care indică sfârşitul de fişier. În DOS şi Windows se utilizează caracterul cu codul ASCII 26 (Ctrl-Z), iar în Unix se utilizează caracterul cu codul ASCII 4. Dacă citim un fişier în mod text, citirea se va opri la intâlnirea acestui caracter, chiar dacă mai există şi alte caractere după el. În modul binar nu există caracter de sfârşit de fişier (mai precis, caracterul cu codul 26, respectiv 4, este tratat la fel ca şi celelalte caractere).
- în DOS şi Windows, în modul text, sfârşitul de linie este reprezentat prin două caractere, CR (Carriage Return, cod ASCII 13) şi LF (Line Feed, cod ASCII 10). Atunci când în modul text scriem un caracter '\n' (LF) în fişier, acesta va fi convertit într-o secventă de 2 caractere CR şi LF. Când citim în mod text dintr-un fişier, secvenţa CR, LF este convertită într-un '\n' (LF). În Unix, sfârşitul de linie este reprezentat doar prin caracterul LF. În mod binar, atât în DOS-Windows cât şi în Unix, sfârşitul de linie este reprezentat doar prin caracterul LF.

Modul binar se utilizează de obicei pentru a scrie în fişier datele exact aşa cum sunt reprezentate în memorie (cu functiile **fread**, **fwrite**) - de exemplu pentru un număr intreg se va scrie reprezentarea internă a acestuia, pe 2 sau pe 4 octeti.

Modul text este utilizat mai ales pentru scrierea cu format (cu funcţiile fprintf, fscanf) - în cazul acesta pentru un număr întreg se vor scrie caracterele ASCII utilizate pentru a reprezenta cifrele acestuia (adică un şir de caractere cum ar fi "1" sau "542").

Citire/scriere cu format

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

Funcţiile sunt utilizate pentru citire/scriere în mod text şi sunt asemănătoare cu printf/scanf (diferenţa fiind că trebuie dat pointerul la fişier ca prim parametru).

Exemplu 1

Să presupunem că avem următorul fișier care pe prima linie conține un număr natural nenul n, iar pe a doua linie se află n numere întregi reprezentând elementele unui vector. Se cere citirea vectorului, dublarea fiecarui element, apoi salvarea rezultatelor în fișierul "gigel.out".

```
gigel.in
5
1 3 -1 6 7
```

Rulați și înțelegeți următorul cod C.

```
#include <stdio.h>
#define NMAX 100

int main() {
    // numele fisierului de intrare
    char input_filename[] = "gigel.in";

    // deschidere fisier de intrare pentru
    // citire (r) in modul text (t)
    FILE *in = fopen(input_filename, "rt");

    // verific daca fisierul a fost deschis cu succes
    // altfel opresc executia (in cazul acestei probleme)
    if (in == NULL) {
        fprintf(stderr, "ERROR: Can't open file %s", input_filename);
        return -1;
    }

    int n, v[NMAX], i; // numarul de elemente && vectorul

    // citesc n din fisier
    fscanf(in, "%d", &n);
    //citesc tabloul
    for (i = 0; i < n; ++i) {
        fscanf(in, "%d", &v[i]);
    }

    // deoarece stiu sigur ca nu mai am nimic de citit
    // pot inchide fisierul de intrare
    fclose(in);

    // dublez elementele din vector
    for (i = 0; i < n; ++i) {
        v[i] <<= 1;
    }

    // deschid fisierul pentru a scrie rezultatele
    char output_filename[] = "gigel.out";
    // deschid pentru scriere (w) in modul text (t)
    FILE *out = fopen(output_filename, "wt");

    // verific daca fisierul a fost deschis cu succes
    // altfel opresc executia (in cazul acestei probleme)
    if (out == NULL) {
```

```
        fprintf(stderr, "ERROR: Can't open file %s", output_filename);
        return -1;
    }

    // scriu n si vectorul in fisier
    fprintf(out, "%d\n", n);
    for (i = 0; i < n; ++i) {
        fprintf(out, "%d ", v[i]);
    }
    fprintf(out, "\n");

    // inchid fisierul de iesire
    fclose(out);

    return 0;
}
```

Pentru exemplul de fișier de intrare de mai sus, rezultatul este următorul.

```
gigel.out
5
2 6 -2 12 14
```

Exemplu 2

Fie un fișier text cu un nume dat. Scrieți o funcție care calculează numărul de bytes (dimensiune fișierului).

```
#include <stdio.h>

int sizeof_file(char *filename) {
    // deschidere fisier de intrare pentru
    // citire (r) in modul text (t)
    FILE *file = fopen(filename, "rt");

    // verific daca fisierul a fost deschis cu succes
    // altfel opresc executia
    if (file == NULL) {
        return -1; // nu am putut calcula dimensiunea
    }

    // ma pozitionez la sfarsit
    fseek(file, 0, SEEK_END);
    // acum cursorul este dupa ultimul caracter
    // deci ftell imi va spune pozitia, care este echivalenta cu numarul de bytes
    int bytes_count = ftell(file);

    // inchid fisierul
    fclose(file);

    return bytes_count;
}

int main() {
    // numele fisierului de intrare
    char filename[] = "gigel.in";

    int sz = sizeof_file(filename);
    if (sz < 0) {
        // afisez la stderr
        fprintf(stderr, "ERROR: Can't open file %s", filename);
        return -1;
    }

    // afisez la stdout rezultatul
    printf("fisierul %s are %d bytes\n", filename, sz);

    // pot folosi tot fprintf pentru a afisa la stdout
    fprintf(stdout, "fisierul %s are %d bytes\n", filename, sz);

    return 0;
}
```

Citire/scriere fără conversie

```
size_t fread(void *ptr, size_t size, size_t nrec, FILE *fp);
size_t fwrite(const void *ptr, size_t size, size_t nrec, FILE *fp);
```

Cu aceste funcţii lucrăm cand deschidem fişierul în mod binar; citirea/scrierea se face fără nici un fel de conversie sau interpretare. Se lucrează cu "înregistrări", adică zone compacte de memorie: funcţia fread citeşte nrec înregistrări începând de la poziţia curentă din fişierul fp, o înregistrare având dimensiunea size. Acestea sunt depuse în tabloul ptr. "Înregistrările" pot fi asociate cu structurile din C - adică în mod uzual, tabloul ptr este un tablou de structuri (dar în loc de structuri putem avea şi tipuri simple de date).

Exemplu 3

Să reluăm problema de la Exemplul 1. Să presupunem că rezolvăm aceeași problemă, doar ca fișierul de intrare este binar: primii 4 bytes din fișier reprezintă numarul n (numărul de elemente); următorii 4*n bytes reprezintă vectorul.

Luăm ca exemplul fișierul text care are conținutul.

```
gigel.in
3
1 2 3
```

Fișierul binar echivalent este "gigel_in.bin".

```
gigel_in.bin
0300 0000 0100 0000 0200 0000 0300 0000
```

- Fișierul poate fi descărcat de aici [http://ocw.cs.pub.ro/courses/_media/programare/teme_2016/lab11_gigel_in.zip].
- Pentru a putea vizualiza continutul mai usor, acesta poate fi deschis cu Sublime [https://www.sublimetext.com/]. Observați reprezentarea în baza 16 a numerelor. Fiecare grup de câte 2 cifre reprezintă un octet. 4 astfel de grupuri formează un int: "0300 0000" semnifică numărul 3.
- Atenție la | Endianness [https://en.wikipedia.org/wiki/Endianness]!

Următoarea sursă C citește vectorul din fișierul "gigel_in.bin", dublează elementele și apoi scrie rezultatul în "gigel_out.bin".

```
#include <stdio.h>
#define NMAX 100

int main() {
    // numele fisierului de intrare
    char input_filename[] = "gigel_in.bin";
    FILE *in;

    // incerc sa dechid pentru citire (r)
    // fisierul in mod binar (b)
    // observati alt mod de a scrie deschiderea && verificarea
    if ((in = fopen(input_filename, "rb")) == NULL) {
        fprintf(stderr, "Can't open %s", input_filename);
        return -1;
    }

    // numarul de elemente, vectorul, variabila de iterare
    int n, v[NMAX], i;

    // doresc sa citesc tinand cont de:
    // voi stoca in n (deci specific adresa lui n ca la scanf => &n)
    // valoarea citita are dimensiunea unui int => sizeof(int)
    // citesc o singura variabila => 1
    // voi citi din fisierul asociat cu variabila in
    fread(&n, sizeof(int), 1, in);
```

```
    // fread permite citirea unui vector printr-un singur apel
    // doresc sa citesc tinand cont de:
    // voi stoca in v (care este adresa primului element)
    // o valoare citita are dimensiunea unui int => sizeof(int)
    // citesc n valori
    // voi citi din fisierul asociat cu variabila in
    fread(v, sizeof(int), n, in);

    // pot inchide fisierul
    fclose(in);

     // dublez elementele din vector
    for (i = 0; i < n; ++i) {
        v[i] <<= 1;
    }

    // salvez rezultatul in fisierul out
     // numele fisierului de intrare
    char output_filename[] = "gigel_out.bin";
    FILE *out;

    // incerc sa dechid pentru scriere (w)
    // fisierul in mod binar (b)
    // observati alt mod de a scrie deschiderea && verificarea
    if ((out = fopen(output_filename, "wb")) == NULL) {
        fprintf(stderr, "Can't open %s", output_filename);
        return -1;
    }

    // doresc sa scriu tinand cont de:
    // voi scrie valoare de la adresa lui n (&n)
    // valoarea scrisa are dimensiunea unui int => sizeof(int)
    // scriu o singura variabila => 1
    // voi scrie in fisierul asociat cu variabila out
    fwrite(&n, sizeof(int), 1, out);


    // fwrite permite scrierea unui vector printr-un singur apel
    // doresc sa scriu tinand cont de:
    // voi scrie elemente incepand de la adresa indicata de v
    // o valoare scrisa are dimensiunea unui int => sizeof(int)
    // scriu n valori
    // voi scrie in fisierul asociat cu variabila out
    fwrite(v, sizeof(int), n, out);

    // inchid fisierul
    fclose(out);

    return 0;
}
```

Rezultatul îl putem vizualiza tot cu Sublime.

```
gigel_out.bin
0300 0000 0200 0000 0400 0000 0600 0000
```

- În acest exemplu am pus extensia “bin” pentru a sugera faptul că fișierele sunt binare. Acest lucru este irelevant pentru funcțiile fread și fwrite.

Citire/scriere la nivel de caracter

```
int fgetc(FILE *fp); // întoarce următorul caracter din fișier, EOF la sfârșit de fișier
```

```
char* fgets(char *s, int n, FILE *fp); // întoarce următoarele n caractere de la pointer sau pâna la sfârșitul de linie
```

```
int fputc(int c, FILE *fp); //pune caracterul c in fișier
```

```
int ungetc(int c, FILE *fp); // pune c în bufferul asociat lui fp (c va fi următorul caracter citit din fp)
```

Exercitii laborator CB/CD

Codul sursa se gaseste aici [http://swarm.cs.pub.ro/~gmuraru/PC/lab11.c], iar fisierul text necesar rularii programului este aici [http://swarm.cs.pub.ro/~gmuraru/PC/lab11.in].

Primul exercitiu presupune modificarea/adaugarea de instructiuni unui cod existent pentru a realiza anumite lucruri. In momentul actual programul citeste dintr-un fisier text mai multe date si le afiseaza la output.

Cerinte:

- Observati care este diferenta de size intre structuri, daca se foloseste pragma si daca nu se foloseste.
- Sa se populeze o structura de tipul 'Group' folosind fisierul binar de aici [http://swarm.cs.pub.ro/~gmuraru/PC/lab11.bin], iar apoi aceste date sa se scrie intr-un fisier text. Datele se citesc in ordine din fisierul binar (adica prima data numarul de persoane apoi persoanele efective).

Următoarele două probleme vă vor fi date de asistent în cadrul laboratorului.

Următoarele două probleme vă vor fi date de asistent în cadrul laboratorului.

Checker laborator 12 [https://drive.google.com/drive/folders/1qB6EZLGVubKbuTXMtMue06egH_8fo25M]

Exerciţii de Laborator

- [2p] Generati un **fisier binar** care sa contina un numar n de numere intregi. n se va citi de la tastatura si cele n numere se vor genera random (hint: **rand**). In fisierul de output, numit `gigel.bin`, scrieti **doar** cele n numere intregi generate. Pentru a va asigura ca in fisier ati scris ce ati dorit, puteti sa afisati numerele pe ecran, apoi sa deschideti fisierul in Sublime (vedeti exemplul din laborator) si sa comparati numerele afisate cu ce ati scris in fisier.

- [2p] Se da un **fisier binar** cu un numar nedeterminat de numere intregi (puteti folosi fisierul generat de voi de la exercitiul anterior). Să se calculeze si sa se afiseze pe ecran suma numerelor **impare** din fişier. Numerele **pare** vor fi citite si apoi afisate in **fisierul text** `gigel.txt`.

- [3p] Să se scrie un program pentru crearea unui **fişier binar**, având articole structuri (de tip student) cu următoarele câmpuri:

```
nume student
   şir de maxim 30 de caractere (alocat static)
data nastere
   o structură având câmpurile de tip int: zi,lună,an
medie
   o valoare reala, de tip float.
```

Pentru a demonstra ca programul vostru functioneaza corect, rulati-l si scrieti cateva astfel de structuri in fisier (ex. un vector cu 10 studenti), apoi scrieti o functie care deschide fisierul creat, citeste structura cu structura din fisier si afiseaza pe ecran ce a citit.

- [3p] Realizati un alt program in care rezolvati exercitiul anterior pentru cazul in care **nume student** este un **sir de caractere alocat dinamic** .

Bonus

1. [3p] Se dau n nume de fișiere citite de la tastatură, fiecare fișier binar conține un vector de numere intregi sortat crescător (nu se cunosc dimensiunile vectorilor). Se cere să salveze în fișierul `sortare.out` (de tip text) vectorul sortat crescător obținut prin interclasarea celor n vectori.

Referinţe

- The GNU Programming Tools - Input and Output [http://crasseux.com/books/ctutorial/Input-and-output.html#Input%20and%20output]

Probleme laborator 14:00 - 16:00 [https://drive.google.com/open?id=1mWuEO8gdQTTfi_BahavCrwsVdiyBZWX_]

programare/laboratoare/lab12.txt · Last modified: 2020/10/05 00:38 by darius.neatu