Limbaje de Programare
Curs 8 – Fișiere

Dr. Casandra Holotescu

Universitatea Politehnica Timișoara

Ce discutăm azi...

1. Lucrul cu fișiere
2. Fișiere de tip text
3. Funcții pentru cazuri de eroare
4. Fișiere binare
5. Poziționare în fișier

Fișiere

Fișier = o colecție de date, păstrate **în mod persistent** pe un dispozitiv de stocare (ex. disc).

Conținutul unui fișier este o **secvență de octeți**.

Aceasta poate fi interpretată în diverse moduri:

linii de text

binar, etc.

Din punct de vedere logic, un fișier poate fi privit ca un flux (**stream**) de octeți.

Fișiere

La nivel de *utilizator*, ne referim la un fișier prin **nume** (ex.: tema.doc, program.c, melodie.mp3, text.txt).

La nivelul interfeței de programare, bibliotecile limbajului C definesc un tip **FILE** cu **elementele necesare accesului la fișier**:

poziția curentă în fișier
tamponul de date
indicatori de eroare și EOF

Atenție: tipul FILE nu poate fi folosit ca atare!

Structura internă a tipului FILE este **invizibilă programatorului**!

Putem folosi doar pointeri **FILE ***
prin intermediul funcțiilor de bibliotecă!

Fișiere standard

Fișiere (FILE *) **standard predefinite** (deschise automat la rulare):

stdin : fișierul standard de **intrare** (implicit: tastatura)
stdout : fișierul standard de **ieșire** (implicit: ecranul)
stderr : fișierul standard de **eroare** (implicit: ecranul)

Toate cele 3 fluxuri de octeți standard (de intrare, de ieșire, de eroare) **pot fi redirecționate**, de ex. din/către alte fișiere.

Obs: E bine ca mesajele de eroare să fie scrise la **stderr**, pentru a putea fi separate (prin redirectare) de mesajele normale de ieșire.

Fișiere standard: redirecționare

Se poate face la rularea programului, din **linia de comandă**:

redirect. intrării standard (stdin):

./program < **in.txt** (citește din in.txt)

redirect. ieșirii standard (stdout):

./program > **out.txt** (scrie la out.txt)

ambele:

./program < **in.txt** > **out.txt**

Redirecționarea se poate face și din program (cu funcția **freopen**).

Lucrul cu fișiere

Pentru a lucra cu fișiere:

1. se **deschide** fișierul: i se asociază un **stream** (o variabilă de tipul de date **FILE ***)
2. se lucrează cu **stream**-ul ca și cum s-ar citi/scrie de la/la intrarea/ieșirea standard (cu aceleași funcții sau cu funcții asemănătoare din **stdio.h**)
3. la sfârșitul prelucrării se **închide** fișierul

Atenție: Orice operație cu fișiere **poate rezulta în eroare**
$\Rightarrow$ e obligatorie **testarea valorii returnate**
de funcțiile care lucrează cu fișiere!

Deschiderea unui fișier

Se face cu funcția:

FILE *fopen (const char *path, const char *mode);

care returnează un **stream** (FILE *) asociat fișierului sau **NULL** dacă fișierul nu s-a putut deschide / în caz de eroare.

Argumentele funcției **fopen**:

- const char* path: **numele/calea fișierului** (absolută sau fața de directorul curent), ca șir de caractere
- const char *mode: **modul de deschidere**, tot ca șir de caractere

Deschiderea unui fișier

const char *mode: **modul de deschidere** poate fi

- **r**: deschidere la citire, fișierul trebuie să existe pe disc!
- **w**: deschidere la scriere, dacă fișierul nu există este creat, dacă există deja conținutul său este trunchiat
- **a**: deschidere la scriere, dacă fișierul există se adaugă conținut în continuarea celui existent

Șirul de caractere **mode** mai poate conține, în continuare:

- **+**: permite și celălalt mod (**r/w**) în plus față de cel specificat deja de primul caracter
- **b**: fișierul se deschide în mod binar (default: în mod text)

Închiderea unui fișier

Se face cu funcția

int fclose(FILE *stream);

Aceasta:

- scrie orice a rămas în tampoanele de date
- închide fișierul
- returnează **0** în caz de **succes**, **EOF** în caz de **eroare**.

⇒ pentru a ne asigura că fișierul s-a închis cu bine
se testează valoarea returnată.

Lucrul cu fișiere: structura tipică

```
char *name = "f.txt";
// numele se poate lua si din argv[]
// sau poate fi citit

FILE *fp = fopen(name, "r");
if(fp == NULL){
    // eroare la deschidere
}else{
    //succes, putem lucra cu fisierul
}

if (fclose(fp)){
    // eroare la inchidere
}
```

Fișiere text

Fișierele text sunt fișiere într-un format ce conține caractere ASCII, ușor de citit de către om. Ex.: prog.c, notitze.txt, pag-web.html

La citirea/scrierea datelor în mod text se pot petrece diverse conversii (de exemplu '\n' din UNIX devine '\r' '\n' în DOS).

Datele citite corespund celor scrise doar dacă:

- caracterele sunt tiparibile, '\t' sau '\n'
- '\n' nu e precedat de spații
- ultimul caracter e '\n'

⇒ altfel, deschideți fișierele **în mod binar** (asigură corespondența **exactă** între datele scrise și cele citite).

Fișiere text - funcții din **stdio.h**

*int fputc(int c, FILE *stream);* //scrie caracter în fișier
*int fgetc(FILE *stream);* //citește caracter din fișier
// getc, putc: ca și fgetc, fputc, dar sunt macrouri (#define)

*int ungetc(int c, FILE *stream);* // pune caracterul c înapoi

*int fscanf (FILE *stream, const char *format, ...);*
*int fprintf(FILE *stream, const char *format, ...);*
//citire/scriere formatata, ca scanf, printf, dar din fisierul stream

*int fputs(const char *s, FILE *stream);* // scrie un șir
*int puts(const char *s);* // scrie șirul și apoi '\n' la ieșire

*char *fgets(char *s, int size, FILE *stream);*
// citește până la (inclusiv) linie nouă, sau max. size - 1 caractere

Exemple

```
#include <stdio.h>
void cat(FILE *fi) // afiseaza fisierul la stdout
{ int c;
  while ((c = fgetc(fi)) != EOF)
      putchar(c);
 }
void main(int argc, char *argv[]) {
  if (argc == 1) cat(stdin); // de la intrare
  else if (argc == 2) {
    FILE *fp = fopen(argv[1],"r");
    if (fp == NULL)
      fprintf(stderr,"can't open %s",argv[1]);
    else {
      cat(fp); fclose(fp);
    }
  }
}
```

Observații: fișiere în mod dual

Citirea și scrierea dintr-un / într-un fișier se fac folosind **același indicator de poziție**.

Pentru un fișier **deschis în mod dual** (cu "r+" sau "w+"):

- **Nu se va citi direct după scriere** fără a goli tampoanele (cu funcția fflush) sau a repoziționa indicatorul de poziție.
- **Nu se scrie direct după citire** fără repoziționarea indicatorului sau fără sa ajungem la EOF.

Funcții pentru cazuri de eroare

*void clearerr(FILE *stream);*
//resetează indicatorii de sfârșit de fișier
//și de eroare pentru fișierul dat

*int feof(FILE *stream);*
// dacă ret. != 0: a ajuns la sfârșit de fișier

*int ferror(FILE *stream);*
// ret. != 0 la eroare pt. acel fișier

*void perror(const char *s);*
//tipărește mesajul s dat de utilizator, un ':', apoi descrierea erorii

void exit(int status);
//termină execuția programului cu valoarea status

Fișiere binare

Păstrează datele exact așa cum au fost scrise,
ca **secvența de octeți neinterpretați**.

Citirea și scrierea se face **direct**, în format binar, fără a ține cont de semnificația datelor scrise/citite (sunt văzute doar ca un număr de octeți).

Prelucrarea fișierelor binare

Se face folosind funcțiile:

*size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);*

*size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *stream);*

Funcțiile de citire/scriere directă:

- citesc/scriu **nmemb** obiecte de câte **size** octeți
- returnează **numărul** obiectelor complete **citite/scrise corect** (dacă e mai mic decât **nmemb** cauza se află din **feof** și **ferror**)

Exemple – citire/scriere binara

```
size_t readint(int *pn, FILE *stream) {
//citeste un int in format binar la adresa pn
//returneaza nr. de valori int citite (0 sau 1)

   return fread(pn, sizeof(int), 1, stream);
}

size_t writedbl(double x, FILE *stream){
//scrie un double x in format binar
//return. nr. de valori double scrise (0 sau 1)

   return fwrite(&x, sizeof(double), 1, stream);
}
```

Copierea a doua fisiere

```
#include <stdio.h>
#define MAX 512
int filecopy(FILE *fi, FILE *fo) {
    char buf[MAX];
    int size; //nr de octeti cititi

    while (!feof(fi)) {
        size = fread(buf, 1, MAX, fi);
        fwrite(buf, 1, size, fo);
        //scrie doar atatia octeti cat a citit

        if (ferror(fi) || ferror(fo))
            return -1;  //eroare
    }
    return 0; //copiere ok
}
```

Funcții de poziționare în fișier

*long ftell(FILE *stream);*
// returnează pozitia de la începutul fișierului

*int fseek(FILE *stream, long offset, int whence);*
// poziționare în fișierul stream

Al treilea parametru la **fseek** (*whence*):
punctul de referință pt. poziționarea cu offset:

- SEEK_SET – începutul fișierului
- SEEK_CUR – punctul curent
- SEEK_END – sfârșitul fișierului

Funcții de repoziționare

*void rewind(FILE *stream);*
// repoziționează indicatorul la început
(echivalent cu (void) fseek(stream, 0L, SEEK_SET), plus clearerr)

Repoziționarea trebuie efectuată:

- când dorim să ignorăm o anumită porțiune din fișier
- când fișierul a fost scris și dorim să revenim să citim din el

*int fflush(FILE *stream);*
//scrie în fișier toate datele din tampoanele de date
//rămase nescrise pt. fluxul de ieșire stream

Alte funcții de lucru cu fișiere

*int remove(const char *filename);*
// șterge un fișier

*int rename(const char *old, const char *new);*
// redenumește un fișier
Ambele funct returnează 0 la succes și != 0 la eroare.

*FILE *freopen(const char * filename, const char * mode, FILE * restrict_stream);*
// deschide fișierul filename și îl asociază cu fluxul stream
// (redirectează fluxul logic stream în fișierul fizic filename)
// returnează NULL în caz de eroare, stream la succes
// închide un eventual fișier asociat anterior cu stream
// se poate folosi pentru redirectarea stdin, stdout, stderr