# Работа с вход-изход. Текстови и двоични файлове

# Входно изходни потоци. Файлове. Работа с файлове.

Въпреки, че С няма дефинирани ключови думи за извършване на файлов вход/изход, стандартната библиотека на С съдържа богата колекция от входно/изходни функции, даващи доста гъвкави и ефикасни решения за работа с потоци от данни.

# Какво е поток?

Входно/изходната система на С предоставя постоянен интерфейс на програмиста, който не зависи от използваното устройство. Това отделяне на хардуера от софтуера се нарича поток, а действително устройството което извършва вход/изход се нарича файл. Потокът се явява логически интерфейс към файла. Така дефиниран в С, терминът файл може да се отнася за дисков, екранен, лентов, клавиатурен файл или такъв в паметта, но без значение от източника, формата и възможностите на файла потоците са еднакви и автоматично се преодоляват различията, което значи, че за програмиста хардуерните устройства са еднотипни.

Потокът се свързва с файл чрез операция за отваряне, а освобождаването става чрез операция за затваряне.

Има два типа потоци: текстов и двоичен. Текстовият поток съдържа ASCII знакове, докато двоичният поток може да се използва със всякакъв тип данни. Не се извършват никакви знакови преобразование и предаването на информация е едно към едно между изпратеното от потока, и това което се съдържа във файла.

## Текущо местоположение на файла

ако файла е голям 500 байта, и вече са прочетени 150 байта, следващият оператор за четене ще започне от 151 байт, тоест от мястото до последната извършена с файла операция.

## Работа с фаилове

Асоцииране на файл ->

За да отворите файл и да го асоциирате с поток, използвайте fopen(). FILE \*fopen(char \*име\_на\_файл, char \*режим);

Функцията fopen() се намира в библиотеката STDIO.H . Името на файла и точният път до него седят в име\_на\_файл. Това име трябва да е валидно файлово име, както е определено от операционната система.

## Режим

r – Отваря текстов файл за четене w – Създава текстов файл за запис

a – добавя към текстов файл, като ако не съществува го създава rb/wb/ab прави горните операции но за двоичен файл (бинарен) r+ -отваря текстов файл за четене/запис

хон. ас. инж. Валентин Бахтев, хон. ас. Момчил Петков, София, 2013

w+ – Създава текстов файл за четене/запис

а+ -добавя към или създава текстов файл за четене/запис

r+b – отваря двоичен файл за четене/запис. Можете да използвате и rb+ w+b – създава двоичен файл за четене/запис. Можете да използвате и wb+

а+b – Добавя към или създава двоичен файл за четене/запис. Можете да използвате и аb+

Ако операцията fopen() е валидна се връща валиден файлов указател. Никога не бива да променяте този указател, или обекта, сочен от него.

Ако fopen() не успее да отвори файл се връща нулев указател. ПРИМЕР:

FILE \*fp;

If((fp = fopen(“myfile” , ”r”)) ==NULL){ prinf(“Error opening file.\n”);

exit(1); // или заменяте с ваш код за грешки

}

Особеност на режимите за работа е, че ако имате файл и го отворите с w то файла ще бъде

изтрит и ще започне записването отначало в този файл. Ако използвате r+ няма да създаде файл, ако той не съществува, отварянето на файл с w+ отново би изтрило старото съдържание на файла.

# Затваряне на файл – fclose()

Int fclose(FILE \*fp);

Използвайки този оператор премахвате асоциацията между потока и файла. Информацията в системите се съхранява на сектори, и до запълване на нужното количество данни за запис на сектор информацията се складира в буфер. Чрез използване на fclose() автоматично записване от буфера на сектора. Функцията fclose() връща нула ако операцията е успешна, а ако се появи грешка връща EOF.

# Функции за четене/запис на байтове (знаци)

Int fgetc(FILE \*fp); -> ако не успее да прочете следващия бит връща EOF Int fputc(int ch, FILE \*fp); -> връща EOF ако се появи грешка.

Пример:

#include<stdio.h> #include<stdlib.h> int main(void)

{

char str[80] = "This is a file system test. \n"; FILE \*fp;

char \*p;

int i;

/\* отваря myfile за изход \*/

if((fp = fopen("myfile", "w"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

/\* записва str на диска \*/ p = str;

while(\*p)

{

if(fputc(\*p, fp)==EOF)

{

printf("Error writing file. \n"); exit(1);

} p++;

}

fclose(fp);

/\* отваря myfile за вход \*/

if((fp = fopen("myfile", "r"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

/\* чете отново файла \*/ for(;;)

{

i = getc(fp);

if(i == EOF) break; putchar(i);

}

fclose(fp); return 0;

}

# Функции от високо ниво

fprintf() и fscanf() действат като scanf() и printf(), с изключение на това, че работят с файлове. Прототипите им са:

int fprintf(FILE \*fp, char \* форматиращ низ); int fscanf(FILE \*fp, char \* форматиращ низ);

имат същите, вече познати свойства като printf() и scanf().

Функцията feof() дава индикатор дали сме стигнали до края на файла, който използваме. Прототип:

Int feof(FILE \*fp);

ПРИМЕР:

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

int main(int argc, char \*argv[])

{

FILE \*fp; double ld; int d;

char str[80];

/\* проверка за аргумент от командния ред \*/ if(argc!=2)

{

printf("Specify file name. \n"); exit(1);

}

/\* отваряне на файл за изход \*/ if((fp = fopen(argv[1], "w"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

fprintf(fp, "%f %d %s", 12345.342, 1908, "hello"); fclose(fp);

/\* отваряне на файл за вход \*/ if((fp = fopen(argv[1], "r"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

fscanf(fp, "%lf%d%s", &ld, &d, str); printf("%f %d %s", ld, d, str); fclose(fp);

return 0;

}

# Четене и записване на двоични данни

Колкото и да са полезни fprintf() и fscanf(), те не са най-ефективният начин за четене и запис на цифрови данни, поради факта че се преминава от двоичен формат в ASCII текст. Функциите fread() и fwrite() могат да четат всякакви типове данни, като използват техните двоични представяния. Прототипите им са:

size\_t fread(void \*buffer, size\_t size,size\_t num,FILE \*fp); size\_t fwrite(void \*buffer, size\_t size,size\_t num,FILE \*fp);

fread() чете от файл \*fp num броя обекти, всеки от тях с размер size байта, в буфер, сочен от buffer. Тя връща броя на прочетените обекти.

fwrite() е противоположна на fread() и записва вместо да чете, аналогично на fread();

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

int main(void)

{

FILE \*fp; int i;

/\* отваряне на файл за изход \*/

if((fp = fopen("myfile", "wb"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

i = 100;

if(fwrite(&i, 2, 1, fp) != 1)

{

printf("Write error occured. \n"); exit(1);

}

fclose(fp);

/\* отваряне на файл за вход \*/

if((fp = fopen("myfile", "rb"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

if(fread(&i, 2, 1, fp) != 1)

{

printf("Read error occured. \n"); exit(1);

}

printf("i is %d", i); fclose(fp);

return 0;

}

# Произволен достъп

До момента записвахме или четяхме файл последователно от началото до края му (ако достигнем такъв). Може да имаме достъп до всяка позиция във файла чрез fseek()

Int fseek(FILE \*fp, long отместване, int начало);

Стойността на отместване определя броя на байтовете след начало, които ще зададат новата текуща позиция. Начало може да бъде:

SEEK\_SET - търси от началото на файла SEEK\_CUR – търси от текущата позиция SEEK\_END – търси от края на файла

Например ако искате да зададете текущото положение на 10 байта от началото на даден файл, тогава начало ще бъде SEEK\_SET, а отместването – 10.

ftell() – определя текущата позиция на файла. Int ftell(FILE \*fp);

Ако се появи греша връща -1.

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

double d[10] = {10.23, 19.87, 1002.23, 12.9, 0.897, 11.45, 75.34,

0.0, 1.01, 875.875};

int main(void)

{

long loc; double value; FILE \*fp;

if((fp = fopen("myfile", "wb"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

/\* записване на целия масив в една стъпка \*/ if(fwrite(d, sizeof d, 1, fp) != 1)

{

printf("Write error. \n");

exit(1);

}

fclose(fp);

if((fp = fopen("myfile", "rb"))==NULL)

{

printf("Cannot open file. \n"); exit(1);

}

printf("Which element? "); scanf("%ld", &loc);

if(fseek(fp, loc\*sizeof(double), SEEK\_SET))

{

printf("Seek error. \n"); exit(1);

}

fread(&value, sizeof(double), 1, fp); printf("Element %ld is %f", loc, value);

fclose(fp); return 0;

}

# Преименуване на файл

Може да преименувате файл, като използвате rename() с прототип: Int rename(char \*старо име, char \*ново име);

Функцията връща нула, ако е завършила и ненулев резултат, ако има грешка.

# Изтриване на файл

Можете да изтриете файл чрез remove(), с прототип:

Int remove(char \* име\_на\_файл);

Позициониране в нчалото на файл чрез използването на функция rewind() с прототип: Int rewind(FILE \*fp);

# Бинарни файлове

В **С** са дефинирани два режима на работа: работа с текстови файлове и работа с двоични файлове.

Съществена разлика между двата вида файлове няма. Единствената разлика идва от интерпретирането на информацията(данните) и различните налични функции за употреба. Основателен би бил въпросът кога какви файлове да използваме. Когато се налага работа с текст, в който всяка една буква се разглежда като символ, по – добрият подход е да се използват текстови файлове. Когато обаче е необходимо да се работи с различен тип данни, които изискват допълнителна обработка, тогава по – добрия избор би бил работа с двоични файлове.

Отварянето на файл е свързано с работа в даден режим:

**rb** - отваря двоичен файл за четене.

**wb** - отваря двоичен файл за запис. Ако такъв файл не съществува, този режим го създава. Ако съществува, то старото съдържание се изтрива.

**ab** - добавя съдържание към двоичен файл. **rb**+ - отваря двоичен файл за четене и запис. **wb+**- отваря двоичен файл за четене и запис.

**ab+ -**добавя към или създава двоичен файл за четене и запис. Нека дадем няколко примера за по-голяма яснота.

Следващата програма отваря два файла, един текстов и един двоичен. Записва в тях числото, записано в променливата ***i*** и след което програмата прочита двата файла и записва тяхното съдържание в променливите ***а*** и ***b***.

Ако след компилирането на дадената програма отворим с текстов редактор двата файла ще установим, че в текстовия файл наистина е записано числото 1500(в дадения случай), защото функцията *fprintf e* форматирала числото като низ от символи. При двоичния файл ще установим, че то е записано с неговия вътрешен формат. Този пример показва, че предимството на работата с текстови файлове е по-добрата им възприемчивост. От друга страна те са по-бавни, заради двойното им преобразуване, заемат повече място и са по-трудни за обработка при различни типове данни.

**#include<stdio.h> #include<stdlib.h> main()**

**{**

**FILE \*fp,\*fp1; int i,a,b; i=1500;**

**if((fp=fopen("binary","wb"))==NULL)**

**{**

**printf("the binary file couldnt open\n"); exit(1);**

**}**

**if((fp1=fopen("txt","w"))==NULL)**

**{**

**printf("the txt file couldnt open\n"); exit(2);**

**}**

**fprintf(fp1,"%d",i); if(fwrite(&i,sizeof(int),1,fp)!=1)**

**{**

**printf("write error occured\n"); exit(3);**

**}**

**fclose(fp); fclose(fp1);**

**if ((fp=fopen("binary","rb"))==NULL)**

**{**

**printf("the binary file couldnt open\n"); exit(4);**

**}**

**if ((fp1=fopen("txt","r"))==NULL)**

**{**

**printf("the txt file couldnt open\n"); exit(5);**

**}**

**if(fread(&a,sizeof(int),1,fp)!=1)**

**{**

**printf("write error occured\n"); exit(6);**

**}**

**fscanf(fp1,"%d",&b);**

**printf("a is %d and b is %d",a,b); fclose(fp);**

**fclose(fp1); return 0;**

**}**

В дадената програма са показани трите основни стъпки при работа с файлове. Отваряне на файл и асоциирането му с поток чрез файлов указател. Ако файла не може да бъде отворен, функцията връща нулев указател. Работата с двоичния файл (четене и запис) се извършва, чрез функциите *fwrite* и *fread*. Прототипите и на двете функции са идентични.

size\_t **fread**(void \*buffer, size\_t size, size\_t num, FILE\*fp); size\_t **fwrite**(void \*buffer, size\_t size, size\_t num, FILE\*fp);

За затварянето на файловете се използва функцията *fclose,* чийто прототип е:

int fclose(FILE\*fp)

***Забележка***: Не затварянето на файл може да има фатални последици, такива като загуба на информация.

В следващата програма ще демонстрираме запис и четене от двоичен файл на различни типове данни. Основната трудност при работата с двоични файлове идва от това, че да прочетем даден файл е необходимо да знаем как този файл е записан. В предната задача записахме едно целочислено число, което е или 2 или 4 байта. Ако не знаем, че сме записали целочислено число, бихме прочели тези 2 или 4 байта последователно по един байт като отделни символи, което ще ни върне напълно безсмислен текст.

Целта на дадената програма е да запише във файл изреченията :

*„My name is Ivan. I am 5 years old.”* , като два отделни низа и целочислена стойност между тях.

**#include<stdio.h> #include<stdlib.h> main()**

**{**

**int a=5,b,i; FILE \*fp;**

**char mas[]="My name is Ivan.I am "; char mas2[]=" years old!";**

**char mas1[sizeof mas];** /\* *масив с големината на масив mas \*/* **char mas3[sizeof mas2];** /\* *масив с големината на масив mas2 \*/* **if((fp=fopen("myfile","wb"))==NULL)** */\* отваря файл за запис\*/*

**{**

**printf("cant open the file"); exit(1);**

**}**

**if(fwrite(mas,1,strlen(mas),fp)!=strlen(mas))** */\*записва масива mas във*

*файла\*/*

**{**

**printf("write error occurred"); exit(2);**

**}**

**if(fwrite(&a,sizeof(int),1,fp)!=1)** */\* записва променливата а във фаила\*/*

**{**

**printf("write error occurred"); exit(3);**

**}**

**if(fwrite(mas2,1,strlen(mas2),fp)!=strlen(mas2))** */\*записва масива mas във*

*файла\*/*

**{**

**printf("write error occurred"); exit(4);**

**}**

**fclose(fp);**

**if((fp=fopen("myfile","rb"))==NULL)** */\* отваря файл за четене\*/*

**{**

**printf("cant open the file"); exit(5);**

**}**

**if(fread(mas1,1,(sizeof mas1)-1,fp)!=((sizeof mas1)-1))** */\* чете от файла и*

*записва в масива mas1 \*/*

**{**

**printf("read error occurred"); exit(6);**

**}**

**if(fread(&b,sizeof(int),1,fp)!=1)** /\*чете от файла годините и записва в b \*/

**{**

**printf("read error occurred"); exit(7);**

**}**

**if(fread(mas3,1,(sizeof mas3)-1,fp)!=(sizeof mas3)-1)** */\* чете от файла и*

*записва в масива mas3 \*/*

**{**

**printf("read error occurred"); exit(8);**

**}**

**fclose(fp);**

**for(i=0; i<(sizeof mas1)-1; i++)** /\* отпечатва информацията последователно

на екрана\*/

**printf("%c",mas1[i]);**

**printf("%d",b);**

**for(i=0; i<(sizeof mas3)-1; i++) printf("%c",mas3[i]);**

**return 0;**

**}**

Какъв е основният недостатък на тази програма? Когато четем от файла, ние четем дадените низове като използваме това, че знаем тяхната дължина предварително. Основателен би бил въпросът възможно ли е да се напише програма в която програмистът, който чете от файла не знае предварително дължината на всеки низ, а само подредбата на данните във файла.

**Пример**

Даден файл има следния формат:

**int | string | double**

В този случай ние не знаем дължината на дадения низ. Съществуват два стандартни начина за решение на дадения проблем. Първият начин е когата записваме файла да добавим едно поле от целочислен тип, което показва дължината на дадения низ.

**int | length of string | string | double**

В този случай четем и записваме целочислена стойност, след това четем байтове на брой дължината на низа и накрая четем числена стойност с двойна точност.

При втория начин избираме един специален символ, който ще ни служи за край на низа, например ; . Дължината на низа се определя от избрания разделител. Този метод обаче е свързан с изрична забрана на дадения символ за използването му в низа. Недостатък на този метод е необходимостта от проверката на всички символи .

Нека покажем един пример как би изглеждала една такава задача.

Да се реализира програма за четене и запис във файл чрез меню за избор. Да се реализират полета както следва: идентификационен номер на дадена стока, име на стока с технически характеристики , цена на дадената стока. Четенето от файл да се реализира, като по зададен идентификационен номер на екрана да се отпечатват всички полета на дадената стока.

**#include<stdio.h> void writening(); void reading(); main()**

**{**

**char choise; do**

**{**

**printf("Enter 1 for writening in file and 2 for reading!\n"); choise=getchar();**

**fflush(stdin);**

**}**

**while(choise!='1'&&choise!='2'); switch(choise)**

**{**

**case '1': writening(); break;**

**case '2': reading(); break;**

**}**

**return 0;**

**}**

**void writening()**

**{**

**int i,number,j; double price; char article[50]; FILE \*fp;**

**printf("enter the string less than 50\n"); gets(article);**

**j=((sizeof(article)));**

**printf("enter the number"); scanf("%d",&number); printf("enter the price"); scanf("%lf",&price);**

**if((fp=fopen("withIn","ab"))==NULL) printf("the file cant open");**

**if(fwrite(&number,sizeof(int),1,fp)!=1) printf("writening error");**

**if(fwrite(&j,sizeof(int),1,fp)!=1) printf("writening error");**

**if(fwrite(article,sizeof (article),1,fp)!=1) printf("writening error");**

**if(fwrite(&price,sizeof(double),1,fp)!=1) printf("writening error");**

**fclose(fp);**

**}**

**void reading()**

**{**

**int i,number,number1; char article[50]; double price;**

**FILE \*fp;**

**printf("enter the number"); scanf("%d",&number1); if((fp=fopen("withIn","rb"))==NULL)**

**printf("the file cant open"); for( ; ;)**

**{**

**if(fread(&number,sizeof(int),1,fp)!=1)**

**{**

**printf("reading error"); break;**

**}**

**if(fread(&i,sizeof(int),1,fp)!=1)**

**{**

**printf("readung error"); exit(1);**

**}**

**if(fread(article,1,i,fp)!=i)**

**{**

**printf("reading error"); exit(7);**

**}**

**if(fread(&price,sizeof(double),1,fp)!=1)**

**{**

**printf("reading error"); exit(5);**

**}**

**if(number==number1)**

**{**

**printf(" %d ",number); printf(article);**

**printf(" %g",price); break;**

**}**

**}**

**fclose(fp);**

**}**

Задачи:

1.В двoичен файл имате цяло число N и след него N цели числа.

а)Да се напише програма, която създава файла.

б)Да се напише програма, която изкарва на екрана броя на четните и броя на нечетните цифри от така дадените N числа.

в)Да се напише програма, която прочита така подадения масив от N елемента и изкарва в текстови файл сортирания по възходящ ред масив.

2. В двоичен файл имате цели числа (не е указано колко на брой).

а)Да се напише програма, която създава файла.

б)Да се напише програма, която изкарва на екрана броя на четните и броя на нечетните цифри от така дадените N числа.

в)Да се напише програма, която прочита така подадения масив от N елемента и изкарва в текстови файл сортирания по възходящ ред масив.

3. Трябва да опишете база данни която работи с автокъща. Данните ви трябва да имат следния вид - марка на автомобил, модел на автомобила - 5 знака, обем на двигателя, цена, има ли регистрация или не (символ Y/N). Създайте програма, която записва така дадената структура на базата данни във двоичен и паралелно в текстови файл. Напишете възможност за добавяне на нов запис. Създайте програма, която прочита двата създадени файла и изкарва един след друг данните от бинарния и текстовия файл, за да направите сравнение за правилната работа на програмата ви.