Práce se soubory

Doteď jsme pracovali tak, že vstup zadával uživatel z klávesnice (v konzoli). Tento přístup se však nehodí pro programy, kde bychom chtěli zadávat velké množství dat. Z tohoto důvodu si ukážeme, jak můžeme v jazyce C pracovat se soubory.

1 Soubor

Soubor je z implementačního hlediska posloupnost bytů uložených na disku v několika blocích. Bloky mají stejnou velikost, ale nemusí nutně ležet v paměti přímo za sebou. Jak se s nimi pracuje je věcí operačního systému a nás to teď nemusí zajímat. Z uživatelského hlediska je soubor posloupnost po sobě jdoucích bytů od začátku do konce souboru (často se používá výraz **proud dat**, nebo stream). Operační systém se stará o to, abychom dostali tyto bloky ve správném pořadí.

Z důvodu rychlosti se soubory pracuje po jednotlivých blocích. Do paměti (bufferu) se načte celý blok a pracuje se s tímto blokem. Pokud například chceme přečíst 2 znaky. Načteme jeden blok do bufferu a z bufferu přečteme nejprve jeden znak a pak druhý (v tuto chvíli už nepřistupujeme k disku).

Obdobně je tomu i u výstupních operací. Nejprve ukládáme data do bufferu a když je plný, zapíše se automaticky celý jeho obsah na disk do souboru jako jeden blok dat.

Základní datový typ pro práci se souborem v jazyce C je `FILE *` (ukazatel na objekt typu `FILE`).

```
FILE *f;
```

`FILE` musí být velkými písmeny. Proměnná `f` se dá použít jak pro čtení, tak pro zápis do souboru. Je vhodné používat identifikátor `*fr` pro soubor, který čteme a `*fw` pro soubor, do kterého chceme zapisovat. Tentýž identifikátor by neměl být v jednom programu používán jak pro čtení, tak pro zápis do souboru.

2 Otevření souboru

Chceme-li se souborem pracovat, je potřeba ho otevřít pomocí funkce `fopen`, která bere jako argumenty název souboru a jakým stylem budeme se souborem pracovat.

```
// otevreni souboru pro cteni
fr = fopen("soubor.txt", "r");

// otevreni souboru pro zapis
fw = fopen("soubor.txt", "w");
```

Přičemž soubor je zadáván pomocí jeho relativní nebo absolutní cesty (dávejte si pozor při zadávání absolutní cesty na použití znaku \, v adrese je potřeba ho zadat zdvojeně. V jazyce C je totiž \ escape sekvence).

Existuje více režimů, jak soubor otevírat, ne jen "w"a "r". Můžeme dále rozlišovat, zda budeme otevírat binární ("rb"pro čtení, "wb"pro zápis), nebo textový soubor ("rt", "wt"). Pokud typ souboru nespecifikujeme, pak se s ním pracuje, jako s textovým.

3 Základní operace s otevřeným souborem

Uvedeme si operace, které jsou součástí standardní knihovny `<stdio.h>`. Proměnná `f` je typu `FILE *`.

```
// cteni znaku ze souboru
c = getc(f);

// zapis znaku c do souboru
putc(c,f);

// formatovane cteni ze souboru
fscanf(f, "format", argumety);

// formatovany zapis do souboru
fprintf(f, "format", argumenty);
```

V podstatě se práce se soubory tolik neliší od práce s konzolí.

```
// cteni znaku z klavesnice
c = getchar();

// zapis znaku na obrazovku
putchar(c);

// formatovane cteni z klavesnice
```

```
scanf("format", argumety);

// formatovany zapis na obrazovku
printf("format", argumenty);
```

4 Ukončení práce se souborem

Když ukončíme práci se souborem (už z něj nepotřebujeme více číst, nebo do něj už nechceme zapisovat, tak to musíme systému oznámit. Říkáme, že soubor chceme zavřít a systému to dámě vědět příkazem `fclose`.

```
fclose(f);
```

Sice se v mnoha systémech soubory po ukončení programu automaticky zavřou, ale není dobré na to spoléhat. Navíc počet současně otevřených souborů je omezený. Díky tomu, že se zapisuje do souboru přes buffer (jak bylo řečeno dříve), tak by se mohlo stát, že po havárii programu by nemuselo být do souboru zapsáno vše (zůstalo to v bufferu).

5 Kontrola správného otevření, nebo zavření souboru

Je dobré testovat, zda se podařilo soubor otevřít/zavřít. Nejčastější příčinou, kdy se nepodařilo soubor otevřít je v případě otevírání souboru pro čtení špatná adresa/název souboru. Další možnou příčinou je to, že je možné mít otevřeno jen omezené množství souborů najednou. Při uzavírání souborů může nastat problém s tím, že tak velký soubor není možné na disk uložit.

Obě funkce `fopen()` i `fclose()` vrací hodnotu, která se používá právě k této kontrole. Při nesprávném otevření `fopen()` vrací `NULL`. Při nesprávném uzavření `fclose()` vrací `EOF`.

```
#include <stdio.h>

int main()
{
    FILE *fr;

    if((fr = fopen("soubor.txt", "r"))==NULL)
        printf("soubor.txt se nepodarilo otevrit \n");

     if(fclose(fr)==EOF)
        printf("soubor.txt se nepodarilo uzavrit \n");

    return 0;
}
```

6 Příklady práce se soubory

6.1 Příklad 1

V prvním příkladu si ukážeme zápis do souboru. Program otevře soubor "priklad1.txt"(pokud neexistuje, tak ho vytvoří) a do něj zapíše na každý řádek číslo od 1 do 10.

```
#include <stdio.h>


int main()
{
    FILE *fw;
    int i;

    fw = fopen("priklad1.txt", "w");

    for(i = 1; i<=10; i++)
        fprintf(fw, "%i\n",i);

    fclose(fw);
    return 0;
}
```

Prohlédněte si soubor "priklad1.txt".

6.2 Příklad 2

Následující program přečte ze souboru "priklad2.txt"(ten si předem vytvořte v libovolném textovém editoru. Je jedno, zda budou čísla oddělena mezerou, nebo každé na jednom řádku) 3 desetinná čísla a vypíše do konzole jejich součet.

```c
#include <stdio.h>


int main()
{
    FILE *fr;
    float x,y,z;

    fr = fopen("priklad2.txt", "r");

    fscanf(fr, "%f %f %f", &x, &y, &z);
    printf("%f\n", x+y+z);

    fclose(fr);
    return 0;
}
```

Funkce `fscanf` vrací počet úspěšně přečtených položek (při čtení konce souboru vrací `EOF`), je tedy možné kontrolovat, zda soubor obsahuje 3 desetinná čísla.

```c
if (fscanf(fr, "%f %f %f", &x, &y, &z)==3)
    printf("%f\n", x+y+z);
else
    printf("Soubor neobsahuje 3 realna cisla");
```

6.3 Příklad 3

Program přečte 2 znaky ze souboru "priklad3.txt"a zapíše je do souboru "priklad3kopie.txt".

Jak bylo zmíněno v předchozím příkladu, pokud čteme konec souboru, dostaneme konstantu `EOF`, která je rovna -1. Je dobré tedy znaky načítat jako int (char s hodnotou -1 není roven -1 a později by nám to zkomplikovalo práci).

```c
#include <stdio.h>


int main()
{
    FILE *fr, *fw;
    int c;

    fr = fopen("priklad3.txt", "r");
    fw = fopen("priklad3kopie.txt", "w");

    // s pomoci promenne c
    c = getc(fr);
    putc(c, fw);

    // primo
    putc(getc(fr), fw);

    fclose(fr);
    fclose(fw);
    return 0;
}
```

6.4 Příklad 4

Nyní si upravíme předchozí program tak, aby zkopíroval celý soubor (ne jen 2 znaky). Jak bylo řečeno, poslední znak v souboru je roven konstantě `EOF`. Budeme tedy testovat, zda přečtený znak není roven této konstantě.

```c
#include <stdio.h>


int main()
{
    FILE *fr, *fw;
    int c;

    fr = fopen("priklad3.txt", "r");
    fw = fopen("priklad3kopie.txt", "w");

    while((c = getc(fr)) != EOF)
        putc(c, fw);

    fclose(fr);
    fclose(fw);
    return 0;
}
```

Jiným způsobem, jak otestovat, zda jsme nenarazili na konec souboru, je využití makra `feof()`. Toho se využívá hlavně v případě, že pracujeme s binárními soubory. Toto makro vrací nenulovou hodnotu (TRUE), pokud poslední čtený znak byl za koncem souboru, nulovou hodnotu (FALSE), pokud ještě nejsme na konci souboru.

```
#include <stdio.h>


int main()
{
    FILE *fr, *fw;
    int c;

    fr = fopen("priklad3.txt", "r");
    fw = fopen("priklad3kopie.txt", "w");

    c = getc(fr);
    while(feof(fr)==0)
    {
        putc(c, fw);
        c = getc(fr);
    }

    fclose(fr);
    fclose(fw);
    return 0;
}
```

6.5 Příklad 5

Při práci s textovými soubory můžeme pracovat s jednotlivými řádky souboru. Pro přečtení jednoho řádku slouží funkce `fgets()`, která uloží přečtený řádek do řetězce.

```
char *fgets(char *str, int max, FILE *fr);
```

Funkce čte řetězec ze souboru `fr` až do konce řádky, maximálně však `max` znaků a to včetně znaků `'\n'` a `'\0'`. Funkce vrací ukazatel na `str` nebo při dosažení konce souboru NULL.

V případě, že je řádek delší, než stanovené `max` bude do řetězce načteno `max-1` znaků a na konec bude přidán znak `'\0'`. Opětovné volání `fgets()` bude pokračovat načítáním stejného řádku, kde přestal.

Předpokládejme, že máme v textovém souboru "priklad5.txt"řádek obsahující čísla od 1 do 9, bude výsledkem následujícího kódu

```
1234
5678
9
```

```
#include <stdio.h>
#define MAX 5

int main()
{
    FILE *fr;
    char str[MAX];

    fr = fopen("priklad5.txt", "r");

    while(fgets(str,MAX, fr) != NULL)  // overeni konce souboru
    {
        printf("%s\n", str);
    }

    fclose(fr);
    return 0;
}
```

Kdybychom chtěli však vypsat celý řádek, museli bychom testovat, zda byl přečtený celý řádek, nebo ne. Pokud jsme načetli celý řádek, pak je na konci řetězce `str` znak dalšího řádku (pokud tedy nejsme na konci souboru. Za posledním řádkem znak dalšího řádku není).

Pro zápis celé řádky (celý řetězec) do souboru použijeme funkci `fputs()`.

```
char *fputs(char *str, FILE *fw);
```

Funkce po zapsání řetězce do souboru nepřidává znak dalšího řádku, ani nezapisuje ukončovací znak. Vrací nezápornou hodnotu, pokud byl zápis úspěšný, `EOF` pokud ne.

Dalším způsobem, jak přečíst řádek souboru, je číst jednotlivé znaky (`getc()`, ale musíme si říct, jak detekovat konec řádku. Jak ale už víme, tak v jazyce C znak `'\n'` označuje nový řádek. Tedy pokud bychom chtěli číst znaky do konce řádku, tak budeme číst znaky dokud nenarazíme na tento znak (případně konec souboru).

```
#include <stdio.h>

int main()
{
    FILE *fr;
    char c;

    fr = fopen("priklad5a.txt", "r");

    // neni zde kontrola konce souboru!
    while((c=getc(fr))!='\n')
    {
        printf("%c", c);
    }

    fclose(fr);
    return 0;
}
```

7 Cvičení

1. Upravte Příklad 4 tak, jak by měl vypadat s ošetřením všech souborových operací (správné otevření/uzavření).
2. Napište program, tak aby zkusil číst neexistující soubor. Zajistěte, aby program vhodně reagoval na tuto situaci.
3. Napište program, který spočítá celkový počet znaků v souboru.
4. Napište program, který do souboru uloží prvních 10 násobků čísel od 1 do 10 (každá série na jeden řádek). V souboru tedy bude 10 řádků, první bude obsahovat čísla od 1 do 10, druhý násobky 2 a pod.
5. Napište program, který bude číst ze souboru desetinná čísla (libovolný počet) a vrátí jejich průměr.
6. **POVINNÁ ÚLOHA - odevzdat do 24. 4.** Napište program, který vypíše na obrazovku soubor (například pr10-dopis.txt) tak, že všechna malá písmena nahradí velkými a naopak a navíc vypíše délku nejdelšího řádku.