KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Jazyk C

Práce se soubory

Mgr. Markéta Trnečková, Ph.D.

Jazyk C

„Give a man a program, frustrate him for a day. Teach a man to program, frustrate him for a lifetime.“

(Muhammad Waseem)

Soubor

- posloupnost bytů uložených v několika blocích – **proud dat** (stream)
- soubory:
  - textové
  - binární

Příklad

Číslo 65535 zabere v textovém souboru prostor 5 bytů, zatímco v binárním stačí byty 2.

Práce se soubory

- **buffer**
- **Základní datový typ**: ukazatel na objekt typu `FILE`

```
FILE *f;
```

- **Otevření souboru**: `fopen()`

```
f = fopen(soubor, rezim);
```

- **Uzavření souboru**: `fclose()`

```
fclose(f);
```

Režim otevření

Požadavek	Režim otevření					
	`"r"`	`"w"`	`"a"`	`"r+"`	`"w+"`	`"a+"`
soubor musí existovat	×			×		
existující soubor bude vymazán		×			×	
existující soubor bude rozšiřován			×			×
neexistující soubor bude založen		×	×		×	×
data lze odkudkoliv číst	×		×	×	×	×
data lze kamkoliv zapisovat		×		×	×	
data lze zapisovat pouze na konec			×			×

- `t` – textový
- `b` – binární

Práce s daty

- **formátovaně**:
  - `fprintf()`
  - `fscanf()`
- **neformátovaně**:
  - po znacích – `getc()`, `putc()`, `fgetc()`, `fputc()`
  - po řádcích – `fgets()`, `fputs()`
  - po blocích – `fread()`, `fwrite()`

Konec souboru – `EOF`

Testování

„Vrací-li funkce chybový kód jako znamení potíží, budeš sledovat její návratovou hodnotu, ano i tehdy, když kontroly ztrojnásobí délku tvého kódu a způsobí puchýře na tvých prstech. Neboť jestliže se domníváš, že "se to nemůže stát", budeš zajisté potrestán za svou domýšlivost.“

(Henry Spencer)

- **Při chybě**:
  - `fopen()` vrací `NULL`
  - `fclose()` vrací konstantu `EOF`

Testování

Příklad

```c
#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;
}
```

Čtení a zápis dat

Formátovaný vstup a výstup

Příklad

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

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

`f` proud dat typu `FILE*`

Čtení a zápis dat

Čtení a zápis jednoho znaku

Příklad

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

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

`f` proud dat typu `FILE*`
`c` proměnná, z/do které budeme načítat znak

Čtení a zápis dat

Čtení a zápis jednoho řádku

Příklad

```
/* cteni radku ze souboru */
char *fgets(char *str, int max, FILE *fr);

/* zapis radku c do souboru */
char *fputs(char *str, FILE *fw);
```

`fr` proud dat typu `FILE*`
`str` řetězec, z/do kterého budeme načítat řádek
`max` maximální počet načítaných znaků

Čtení a zápis dat

Čtení a zápis jednoho bloku

Příklad

```
/* cteni bloku */
int fread(void *kam, size_t rozmer, size_t pocet, FILE *fr);

/* zapis */
int fwrite(void *odkud, size_t rozmer, size_t pocet, FILE *fw);
```

`fr`, `fw` proud dat typu `FILE*`
`kam` ukazatel do paměti, kam budeme načítat
`odkud` ukazatel do paměti, odkud budeme načítat
`rozmer` velikost položky, kterou načítáme (v bytech)
`pocet` počet načítaných položek

`feof()` testuje konec souboru
`ferror()` testuje zda nedošlo k chybě

Čtení a zápis dat

Pozice čtení a zápisu

- **kurzor** – pozice v souboru
- **změna pozice** – `fseek()`

Příklad

```
int fseek(FILE *f, long posun, int odkud);
```

`fr`, `fw` proud dat typu `FILE*`
`posun` o kolik bytů se má kurzor posunout
`odkud` odkud se posun počítá

- **odkud**:
  - `SEEK_SET` – začátek souboru,
  - `SEEK_CUR` – aktuální pozice,
  - `SEEK_END` – konec souboru.

Čtení a zápis dat

Další funkce

- `ftell()` – vrací aktuální pozici kurzoru v souboru `f`
- `rewind()` – posune kurzor na začátek souboru `f`
- `ungetc()` – vrácení znaku zpět do bufferu

Standardní vstup a výstup

- knihovna `stdio` obsahuje ukazatele typu `FILE*`:
  - `stdin` – většinou vstup z klávesnice
  - `stdout` – většinou výpis do konzole
  - `stderr` – výpis chybových zpráv

Příklad

```
/* cteni jednoho znaku z klavesnice */
getc(stdin);
getchar();

/* vypis jednoho znaku na obrazovku */
putc(c, stdout);
putchar(c);
```

Standardní vstup a výstup

Přesměrování

Operační systém:

```
C:\Umisteni_souboru\program.exe > output.txt
C:\Umisteni_souboru\program.exe < input.txt
C:\Umisteni_souboru\program1.exe | program2.exe
```

Bufferování

- **Řádkové bufferování** – `stdin`
- **Blokové bufferování** – práce se soubory
- **Žádné bufferování** – `stdout` a `stderr`.

Bufferování
Funkce

- `setbuf()`

```
void setbuf(FILE *f, char *buffer);
```

- `setvbuf()`

```
int setvbuf(FILE *f, char *buffer, int rezim, int velikost);
```

režimy bufferování:
- `_IOFBF` – blokové (F – full)
- `_IOLBF` – řádkové (L – line)
- `_IONBF` – žádné (N – none)

Další funkce

- `freopen()`

```
FILE *freopen(const char *jmeno, char *rezim, FILE *stream);
```

- `rename()`

```
int rename(const char *stary, const char *novy);
```

- `remove()`

```
int remove(const char *jmeno);
```

- `tmpfile()`

```
FILE *tmpfile();
```

Cvičení

1. Napište program, který bude číst ze souboru desetinná čísla (libovolný počet) a vrátí jejich průměr.
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 uloží čísla od 0 do 9 do binárního i textového souboru. Zjistěte velikosti těchto souborů. Otevře si tyto soubory v textovém editoru.
5. Napište program, který uloží číslo 1234567 do binárního i textového souboru. Zjistěte velikosti těchto souborů. Proč je v minulém případě velikost binárního souboru větší a v tomto ne?
6. Napište program, který do binárního souboru uloží pole desetinných čísel (záleží na vás, jak velké).

Cvičení

7. Napište program, který z desetinných čísel uložených v binárním souboru (například vytvořeného v předchozí úloze) spočítá průměr. Měl by být napsán obecně tak, aby fungoval i když nebudeme vědět, kolik je v souboru čísel.
8. Napište program, který čte znaky ze vstupního souboru a opisuje je buď na obrazovku, nebo do souboru `"novy.txt"`. To, kam se má provést výstup, zvolí uživatel.
9. Napište program, který čte čísla ze vstupního souboru a ukládá je do jiného souboru. Pokud je čtené číslo větší než 100, číslo se neuloží do souboru, ale na obrazovku se vypíše chybová hláška `Špatně zadané číslo CISLO` (`CISLO` je konkrétní číslo). Použijte vhodně proud `stderr`.
10. Předpokládejme, že existuje soubor zprava.txt, obsahující následující text (bez uvozovek, včetně mezer):
    `"jakuzyp k sec stje v cajooecl"`
    Máme následující program `tajemstvi`:

Cvičení

Příklad (tajemstvi.c)

```c
#include <stdio.h>
int main(){
    char c;
    int i=0;
    while((c=getchar()) != EOF){
        if((i % 4) > 1)
            putc(c,stdout);
        else
            putc(c,stderr);
        i++;
    }
    return 0;
}
```

Co bude obsahem souborů `zprava1.txt` a `zprava2.txt`, pokud program spustíme následujícím způsobem?

`tajemstvi.exe < zprava.txt > zprava1.txt 2> zprava2.txt`

Svou domněnku ověřte.