9. gyakorlat

Fájlok, fájlkezelés.

Elmélet

A fájlok a lemezen (HDD, SSD, flash) tárolt állományok. Tartalmukhoz az operációs rendszeren keresztül férhetünk hozzá. A fájlok elhelyezése, írása, olvasása a lemezen a fájlrendszer feladata, ezekkel nem nekünk kell foglalkoznunk.

A fájl tartalma az amit beleírunk. Amikor elmentünk egy új txt fájlt "Hello World" tartalommal, az pontosan ezeket az adatokat fogja tartalmazni: 11 karaktert. A fájlhoz társított információk, mint a fájlnév, jogosultságok, létrehozás dátuma a fájlrendszerben kerülnek tárolásra, és nem a fájlban jelennek meg.

A fájlokat két típusba lehet sorolni:

1. A **szövegfájlok** szöveges információt tárolnak. Nem csak a txt fájlok tartoznak ide, hanem minden, amit a szövegszerkesztővel megnyitva értelmes dolgokat olvashatsz. Szövegfájlok a C forrásfájlok, a CSV fájlok, a HTML weboldalak, a JavaScript kód, az XML dokumentumok (docx tartalma), az svg képfájlok.
2. A **bináris fájlok** nem szöveges adatokat tárolnak. A számok és egyéb objektumok bináris formában kerülnek benne tárolásra, vagyis úgy, ahogy a memóriában is vannak. Ezeket szövegszerkesztővel megnyitva csak felismerhetetlen hieroglifákat láthatunk. Ilyen fájlok például az .exe programfájlok, a dll könyvtárak, a modellezőprogramok mentései.

A szövegfájlok előnye, hogy rendszerek között könnyedén hordozhatók, mindenféle rendszeren el lehet őket olvasni. Programok közti kommunikációra ilyeneket szoktunk használni. Például a táblázatos adatokat CSV (Comma Separated Values), XML vagy JSON formában szoktuk átadni egyik programból a másikba.

A bináris fájlok ellenben kisebb helyet foglalnak el, gyorsabb a mentésük és a betöltésük. Gondoljunk bele: az "1234567890" szám tárolása szovegesen 10 karaktert, azaz 10 bájtot foglal el. Ugyanez binárisan belefér egy 4 bájt méretű integer változóba.

A C-ben minden fájl *stream*, azaz magnószalagként lehet elképzelni a működést. A "szalagot" lehet tekerni előre-hátra, olvasni azokat a bájtokat amik éppen az olvasófej alatt vannak. Szövegfájl esetén bájtonként olvasunk, bináris fájl esetén az adattípus méretének megfelelő blokkonként. Mi most csak a szövegfájlokkal fogunk foglalkozni.

Fájl megnyitása

A fájl betöltésére és megnyitására az `fopen` függvény való. Ez kér tőlünk egy elérési utat (*path*) ami tartalmazza a fájl nevét is. Ha a program mappájában akarunk fájlt nyitni, akkor a path egyszerűen a fájl neve.

```
FILE* fopen(const char* path, const char* mode)
FILE* fopen_s(FILE** fileprt, const char* path, const char* mode)
```

A mód a megnyitás módját tartalmazza. Lehetőségek:

- `r` csak olvasás
- `w` csak írás, szükség esetén létrehozza a fájlt

- `r+` írás és olvasás, létező fájlon
- `w+` írás és olvasás, szükség esetén létrehozza a fájlt
- `a` fájl végére írás (append)
- `a+` fájl végére írás és olvasás

Általában a `r`, `w` és `a` módokat használjuk.

A fopen visszatérési értéke egy FILE pointer. Ez tulajdonképpen a fájlra mutat. Az író-olvasó függvények ez alapján tudják, hogy melyik fájlról van szó.

Fájlba írás

A fájlba írásra az `fprintf` függvény használható. Ennek első argumentuma a FILE pointer, egyébként ugyanúgy működik, mint a sima `printf`.

```
fprintf(FILE* file, const char* format, ...)
```

Fájlból olvasás

Erre lehet használni az `fscanf` függvényt, de általában célszerűbb `fgets`-et használni.

```
fscanf(FILE* file, const char* format, ...)
fscanf_s(FILE* file, const char* format, ...)
```

```
fgets(char* string, int size, FILE* stream)
```

Az fscanf csak egy-egy szót olvas be, mert a szóközöknél befejezi az olvasást. Az fgets egy sort olvas be a fájlból, a végén lévő enterrel együtt.

A fájl tartalma egy hosszú stringként képzelhető el, melyben a sorokat `\n` karakterek vagy `\r\n` karakterpárok zárják. A fájl végén nincs semmilyen speciális jelzés.

> Megjegyzés: *A `\n` vagy LF karakter a soremelést jelenti. A `\r` vagy CR a "carriage return" parancs, ami a villany-írógépeken a sor elejére küldte vissza a kocsit. A Windowson létrehozott fájlokban CRLF van a sorok végén. Mindenhol máshol csak LF.*

Az oprendszer tudja, hogy hol van vége a fájlnak. Ezt a `feof` parancssal kérdezhetjük meg tőle. A válasz igaz (1), ha a "kurzorunkkal" a fájl végére értünk, és hamis (0), ha még nem.

```
int feof(FILE* file)
```

A fájl végigolvasása ilyen ciklussal megy:

```
while (feof(file) == 0)
{
    fgets...
}
```

Fájl bezárása

A fájlokat használat után be kell zárni, hogy más program is használhassa őket.

```
fclose(FILE* file)
```

Példaprogram

Írjunk egy egyszerű példaprogramot, ami fájlba ír egy szöveget, aztán elolvassa azt.

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

#define FILENAME "tesztfajl.txt"
#define SOR_HOSSZ 81

void fajl_iras()
{
    FILE* file;
    file = fopen(FILENAME, "w");

    // hosszú szöveg így is megadható:
    const char* szoveg = "Boci boci tarka\n"
                         "Se fule se farka\n"
                         "Oda megyunk lakni\n"
                         "Ahol tejet kapni\n";

    fprintf(file, "%s", szoveg);

    fclose(file);
}

void fajl_olvasas()
{
    FILE* file;
    file = fopen(FILENAME, "r");
    if (file == NULL)
    {
        printf("A fajl nem lezetik.");
        return;
    }

    char buffer[SOR_HOSSZ]; // max sorhossz

    while (feof(file) == 0)
    {
        memset(buffer, 0, SOR_HOSSZ);
        fgets(buffer, SOR_HOSSZ, file);
        printf("%s", buffer);
    }

    fclose(file);
}
```

```
int main()
{
    fajl_iras();
    fajl_olvasas();

    exit(EXIT_SUCCESS);
}
```

Feladatok

1. Írj egy programot, ami megszámolja egy fájl sorait!

2. Írj egy programot, ami a saját forrásfájlját kiírja a képernyőre!

3. Készíts függvénytáblát!

A program 0 és 2*pi közötti x értékekre számolja ki a sin(x) értékeit, és ezeket soronként, vesszővel elválasztva tárolja így:

```
x,y
0.00,0.00
0.01,0.01
0.02,0.02
...
```

Excelben nyisd meg és plottold ki az eredményt!

> Megjegyzés: *CSV helyett szinte mindig jobb TSV-t, tabulátorral tagolt fájlt használni. A* `\t` *tabulátor karakter nem keveredik össze a tizedesponttal és tizedesvesszővel, és szövegben sem szoktunk ilyen karaktert használni.*