# Óbudai Egyetem
# Kandó Kálmán Villamosmérnöki Kar

## C programozási nyelv
## Magasszintű fájlkezelés I.

Dr. Schuster György

2018. január 3.

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

Ebben minnden fontos függvény és fordítás idejű konstans szerepel.

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

Ebben minnden fontos függvény és fordítás idejű konstans szerepel.

Magasszintű fájlkezelésnél a fájl azonosítása egy **`FILE`** típusú mutatóval történik. A szokásos neve fájl mutató, vagy fájl pointer.

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

Ebben minnden fontos függvény és fordítás idejű konstans szerepel.

Magasszintű fájlkezelésnél a fájl azonosítása egy **`FILE`** típusú mutatóval történik. A szokásos neve fájl mutató, vagy fájl pointer.
**Nem tévesztendő a fájl pozíció mutatóval**.

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

Ebben minnden fontos függvény és fordítás idejű konstans szerepel.

Magasszintű fájlkezelésnél a fájl azonosítása egy **`FILE`** típusú mutatóval történik. A szokásos neve fájl mutató, vagy fájl pointer.
**Nem tévesztendő a fájl pozíció mutatóval**.

A **`FILE`** típus egy az **`stdio.h`**-ban típusként definiált struktúra, amely az adott fájl jellemzőit tartalmazza megnyitás után.

# Ismérvek

Az alap szinten a magasszintű fájlkezeléshez csak egyetlen header fájl szükséges az

**`stdio.h`**

Ebben minnden fontos függvény és fordítás idejű konstans szerepel.

Magasszintű fájlkezelésnél a fájl azonosítása egy **`FILE`** típusú mutatóval történik. A szokásos neve fájl mutató, vagy fájl pointer.
**Nem tévesztendő a fájl pozíció mutatóval**.

A **`FILE`** típus egy az **`stdio.h`**-ban típusként definiált struktúra, amely az adott fájl jellemzőit tartalmazza megnyitás után.

*Néha a magasszintű fájlkezelést folyam (stream) jellegű fájlkezelésnek is hívják.*

# fopen

Magasszinten a fájl megnyitása az **fopen** függvénnyel történik.

# fopen

Magasszinten a fájl megnyitása az **fopen** függvénnyel történik.Deklarációja:

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

# fopen

Magasszinten a fájl megnyitása az **fopen** függvénnyel történik.Deklarációja:
**FILE *fopen(const char *path, const char *mode);**
Paraméterei:

- **const char *path** a fájl elérési útja, pl.:
  **fp=fopen("./text.txt", ...**,

# fopen

Magasszinten a fájl megnyitása az **fopen** függvénnyel történik.Deklarációja:
**FILE *fopen(const char *path, const char *mode);**
Paraméterei:

- **const char *path** a fájl elérési útja, pl.:
  **fp=fopen("./text.txt", ...**,
- **const char *mode** a megnyitás módja:
  - **r** a fájl megnyitása olvasásra,
  - **r+** a fájl megnyitása olvasásra és írásra,
  - **w** a fájl megnyitása létrehozásra és írásra, a megynyitáskor a már esetlegesen létező fájlt 0 hosszúságúra levágja,
  - **w+** a fájl megnyitása létrehozásra, írásra, és olvasására a megynyitáskor a már esetlegesen létező fájlt 0 hosszúságúra levágja,
  - **a** a fájl megnyitása létrehozásra, és hozzáfűzésre, a fájl pozíció mutató a fájl végére áll.
  - **a+** a fájl megnyitása létrehozásra, olvasásra, és hozzáfűzésre, a fájl pozíció mutató a fájl végére áll.

# fopen

Megjegyzés: Windows esetben, ha a fájlt bináris módban akarjuk megnyitni, akkor a **mode** paraméterhez hozzá kell írni egy **b**-t, pl.: **"rb"**.

# fopen

Megjegyzés: Windows esetben, ha a fájlt bináris módban akarjuk megnyitni, akkor a **mode** paraméterhez hozzá kell írni egy **b**-t, pl.: **"rb"**.

Visszatérési érték:

- siker esetén a fájl pointer,
- hiba esetén **NULL** pointer (a **NULL** egy 0 értékű **void** pointer).

# fopen

Megjegyzés: Windows esetben, ha a fájlt bináris módban akarjuk megnyitni, akkor a **mode** paraméterhez hozzá kell írni egy **b**-t, pl.: **"rb"**.

Visszatérési érték:

- siker esetén a fájl pointer,
- hiba esetén **NULL** pointer (a **NULL** egy 0 értékű **void** pointer).

Példa:

```
{
FILE *fp;
    :
fp=fopen("./text.txt","r");
```

A **text.txt** fájlt csak olvasásra nyitjuk meg.

# `fclose` függvény

Ha az állományt nem használjuk többé, azt le kell zárni. Erre szolgál a **`fclose`** függvény.

# `fclose` függvény

Ha az állományt nem használjuk többé, azt le kell zárni. Erre szolgál a **`fclose`** függvény. Deklarációja:

```
int fclose(FILE *fp);
```

Paramétere:

- **`FILE *fp`** fájl pointer.

# `fclose` függvény

Ha az állományt nem használjuk többé, azt le kell zárni. Erre szolgál a **`fclose`** függvény. Deklarációja:

```
int fclose(FILE *fp);
```

Paramétere:

- **`FILE *fp`** fájl pointer.

Visszatérési értéke:

- 0, ha a fájl lezárása sikeres,
- -1, vagy **`EOF`**, ha sikertelen.

# `getc` függvény

Egy bájt beolvasása a fájlból.

# `getc` függvény

Egy bájt beolvasása a fájlból. Deklarációja:

```
int getc(FILE *fp);
```

Paramétere:

- **`FILE *fp`** a fájl pointer.

# `getc` függvény

Egy bájt beolvasása a fájlból. Deklarációja:

```
int getc(FILE *fp);
```

Paramétere:

- **`FILE *fp`** a fájl pointer.

Visszatérési érték:

- sikeres beolvasás esetén a az aktuális bájt értéke,
- hiba esetén -1, vagy **`EOF`**.

# `getc` függvény

Egy bájt beolvasása a fájlból. Deklarációja:

```
int getc(FILE *fp);
```

Paramétere:

- **`FILE *fp`** a fájl pointer.

Visszatérési érték:

- sikeres beolvasás esetén a az aktuális bájt értéke,
- hiba esetén -1, vagy **`EOF`**.

Megjegyzés: a függvény típusa **`int`** holott a beolvasott adat csak bájt méretű. Ez azért van így, mert az átvitel transzparens, tehát értékre tekintve $\{0,1,2,\ldots,255\}$. Viszont a hibajelzésre nem marad hely, így a helyes érték esetén az érték 0 és 255 között van, a hiba pedig **`int`** -1.

# `putc` függvény

Egy bájt kiírása a fájlba.

# `putc` függvény

Egy bájt kiírása a fájlba. Deklarációja:

```
int putc(int chr,FILE *fp);
```

Paramétere:

- **`int chr`** a kiírandó bájt értéke,
- **`FILE *fp`** a fájl pointer.

# `putc` függvény

Egy bájt kiírása a fájlba. Deklarációja:

```
int putc(int chr,FILE *fp);
```

Paramétere:

- **`int chr`** a kiírandó bájt értéke,
- **`FILE *fp`** a fájl pointer.

Visszatérési érték:

- sikeres kiírás esetén a kiírt bájt értéke,
- hiba esetén -1, vagy **`EOF`**.

# `putc` függvény

Egy bájt kiírása a fájlba. Deklarációja:

```
int putc(int chr,FILE *fp);
```

Paramétere:

- **`int chr`** a kiírandó bájt értéke,
- **`FILE *fp`** a fájl pointer.

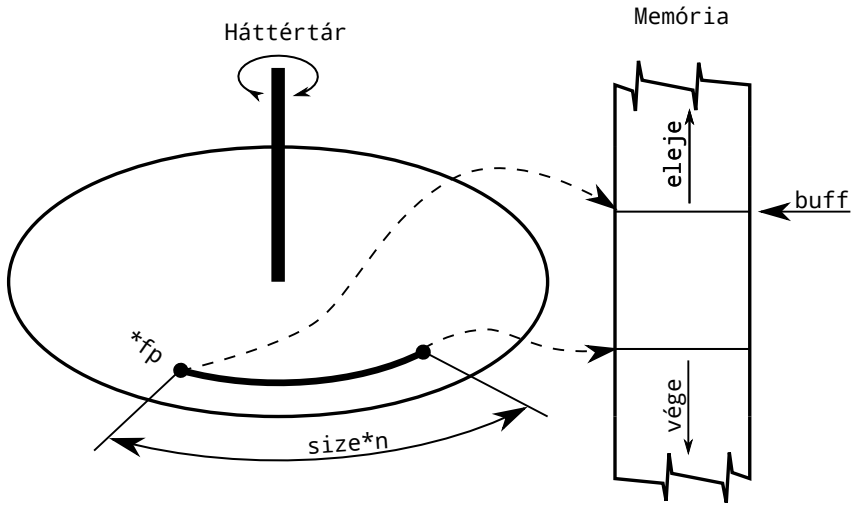
Visszatérési érték:

- sikeres kiírás esetén a kiírt bájt értéke,
- hiba esetén -1, vagy **`EOF`**.

Megjegyzés: a kiírandó bájt típusa **`int`**, ennek történelmi hagyományai vannak. A visszatérési értékkel kapcsolatos megjegyzést lásd **`getc`** függvénynél.

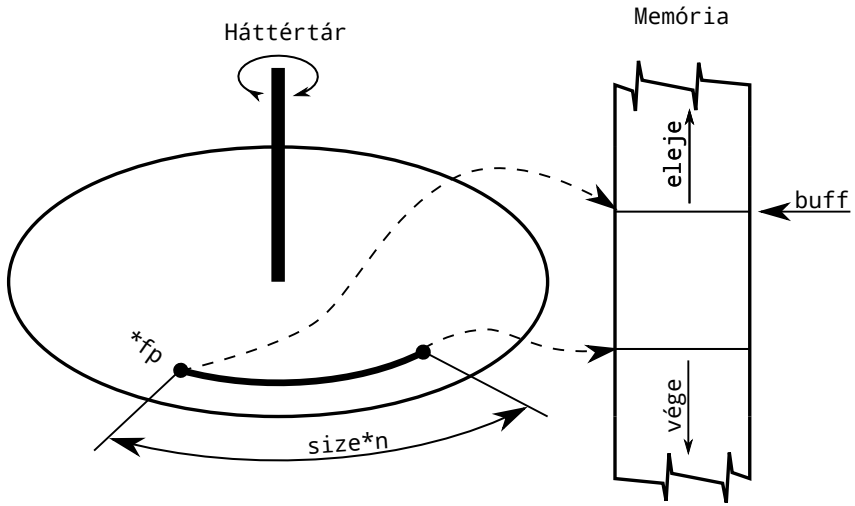
# `fread` függvény

Adatok beolvasása fájlból memóriába.



# `fread` függvény

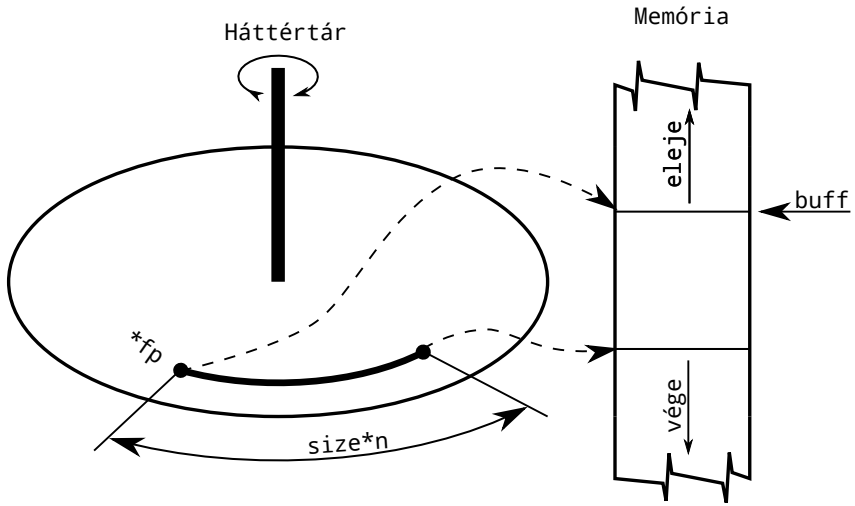
Adatok beolvasása fájlból memóriába. Deklarációja:



# `fread` függvény

Adatok beolvasása fájlból memóriába. Deklarációja:

```
ssize_t fread(void *buff, size_t size,size_t n,
FILE *fp);
```



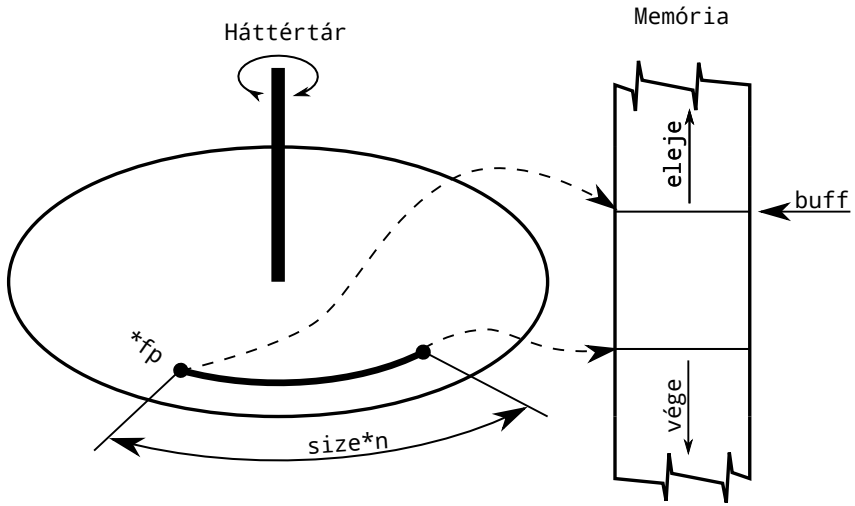
# `fread` függvény

Adatok beolvasása fájlból memóriába. Deklarációja:

```
ssize_t fread(void *buff, size_t size,size_t n,
FILE *fp);
```

Paraméterei:

- **`void *buff`** a memória cím, ahova olvasunk,
- **`size_t size`** az olvasandó adatelemek mérete.
- **`size_t n`** az olvasandó adatelemek száma.
- **`FILE *fp`** a fájl pointer.



# `fread` függvény

Adatok beolvasása fájlból memóriába. Deklarációja:

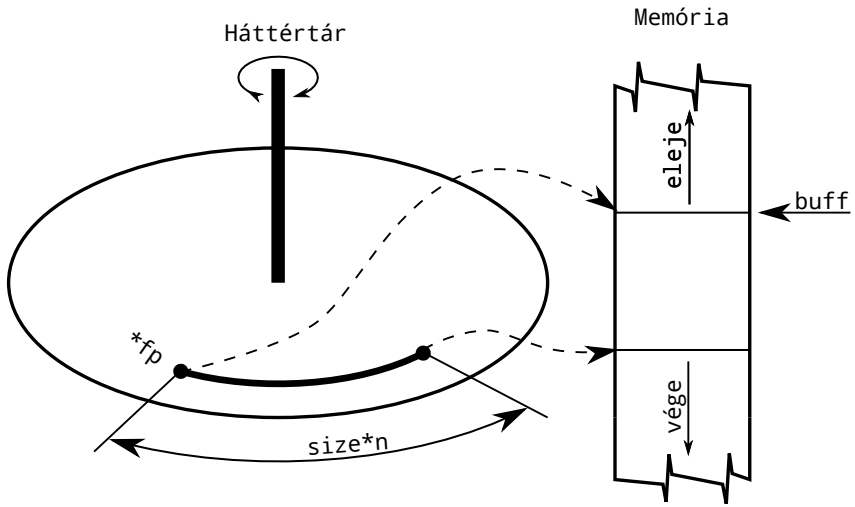
```
ssize_t fread(void *buff, size_t size,size_t n,
FILE *fp);
```

Paraméterei:

- **`void *buff`** a memória cím, ahova olvasunk,
- **`size_t size`** az olvasandó adatelemek mérete.
- **`size_t n`** az olvasandó adatelemek száma.
- **`FILE *fp`** a fájl pointer.

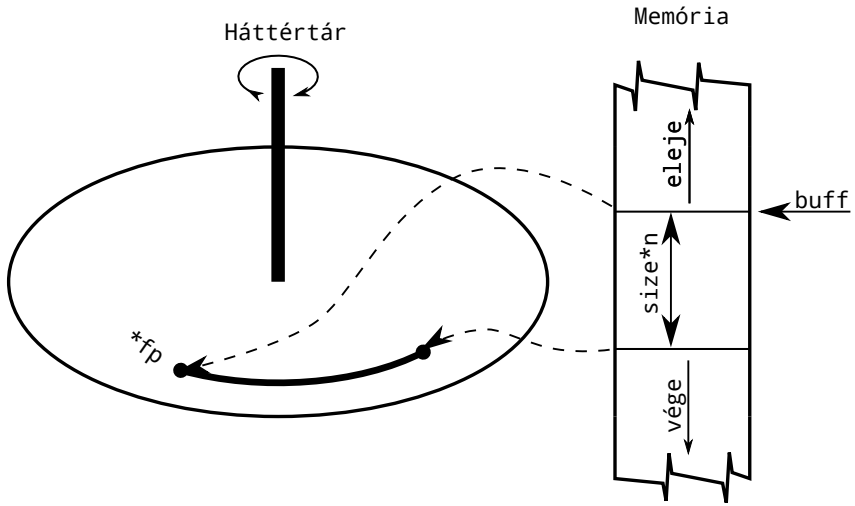
Visszatérési értéke:

- a beolvasott adatelemek száma, ha az olvasás sikeres,
- -1, vagy **`EOF`**, ha sikertelen.



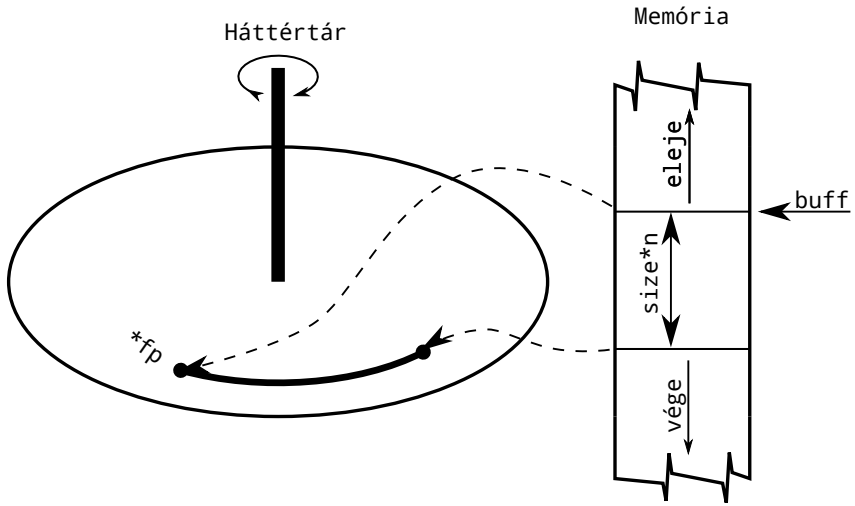
# `fwrite` függvény

Adatok kiírása memóriából fájlba.



# `fwrite` függvény

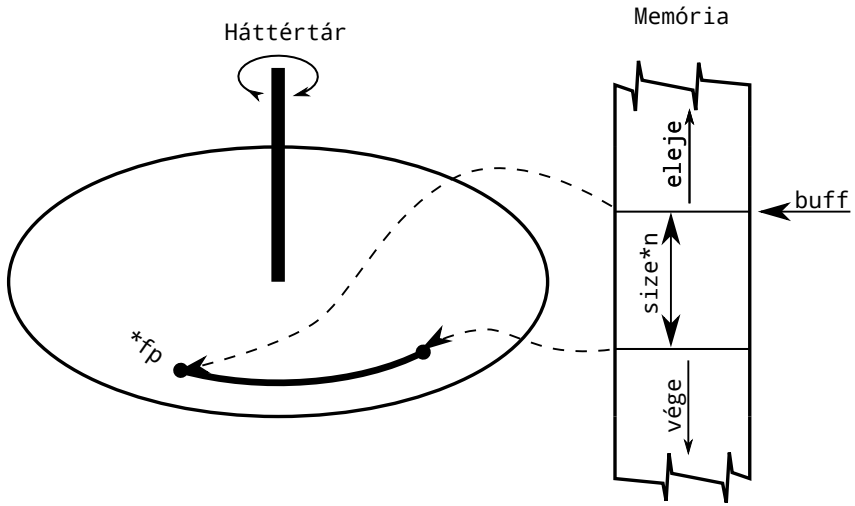
Adatok kiírása memóriából fájlba. Deklarációja:



# `fwrite` függvény

Adatok kiírása memóriából fájlba. Deklarációja:

```
ssize_t fwrite(void *buff, size_t size,size_t n,
FILE *fp);
```



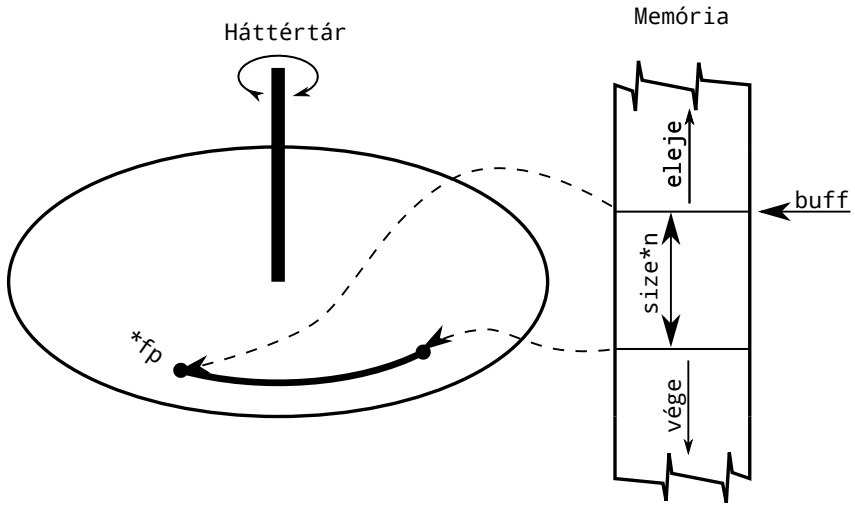
# `fwrite` függvény

Adatok kiírása memóriából fájlba. Deklarációja:

```
ssize_t fwrite(void *buff, size_t size,size_t n,
FILE *fp);
```

Paraméterei:

- **`void *buff`** a memória cím, ahonnan írunk,
- **`size_t size`** az írandó adatelemek mérete.
- **`size_t n`** az írandó adatelemek száma.
- **`FILE *fp`** a fájl pointer.



# `fwrite` függvény

Adatok kiírása memóriából fájlba. Deklarációja:

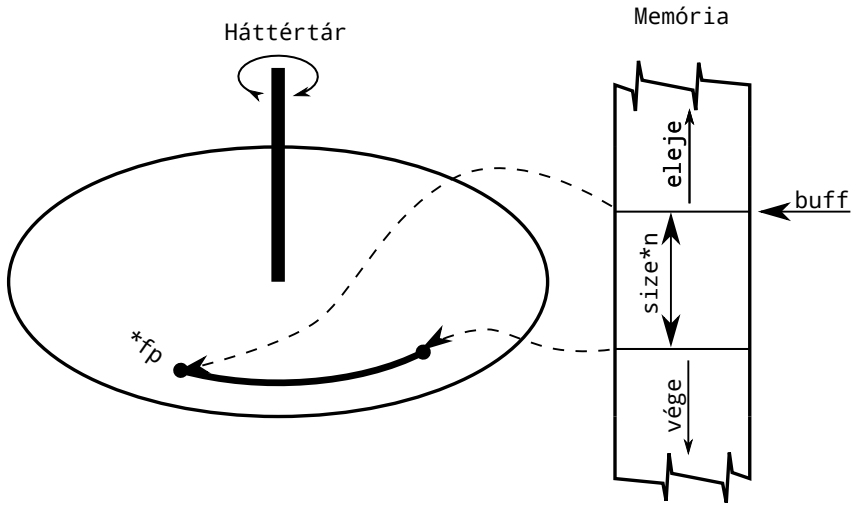
```
ssize_t fwrite(void *buff, size_t size,size_t n,
FILE *fp);
```

Paraméterei:

- **`void *buff`** a memória cím, ahonnan írunk,
- **`size_t size`** az írandó adatelemek mérete.
- **`size_t n`** az írandó adatelemek száma.
- **`FILE *fp`** a fájl pointer.

Visszatérési értéke:

- a kiírt adatelemek száma, ha az írás sikeres,
- -1, vagy **`EOF`**, ha sikertelen.



# `fseek` függvény

A fájl pozíció mutató beállítása:

# `fseek` függvény

A fájl pozíció mutató beállítása:Deklarációja:

```
off_t fseek(FILE *fp, off_t offset, int whence);
```

# `fseek` függvény

A fájl pozíció mutató beállítása:Deklarációja:

```
off_t fseek(FILE *fp, off_t offset, int whence);
```

Paraméterei:

- **`FILE *fp`** a fájl pointer,
- **`off_t offset`** az az érték, amivel a fájl pozíció mutató értéket módosítjuk,
- **`int whence`** honnan értelmezzük a módósítás értékét, ez lehet:
  - **`SEEK_SET`** a fájl elejéről (0),
  - **`SEEK_CUR`** az aktuális értéktől (1),
  - **`SEEK_END`** a fájl végétől (2).

# `fseek` függvény

A fájl pozíció mutató beállítása:Deklarációja:

```
off_t fseek(FILE *fp, off_t offset, int whence);
```

Paraméterei:

- **`FILE *fp`** a fájl pointer,
- **`off_t offset`** az az érték, amivel a fájl pozíció mutató értéket módosítjuk,
- **`int whence`** honnan értelmezzük a módosítás értékét, ez lehet:
  - **`SEEK_SET`** a fájl elejéről (0),
  - **`SEEK_CUR`** az aktuális értéktől (1),
  - **`SEEK_END`** a fájl végétől (2).

Visszatérési értéke:

- az új (beállított) fájl pozíció érték sikeres esetben,
- -1, vagy **`EOF`** hiba esetén.

# `ftell` függvény

A fájl pozíció mutató lekérdezése:

# `ftell` függvény

A fájl pozíció mutató lekérdezése:Deklarációja:

```
long ftell(FILE *fp);
```

# `ftell` függvény

A fájl pozíció mutató lekérdezése:Deklarációja:

```
long ftell(FILE *fp);
```

Paraméterei:

- **`FILE *fp`** a fájl pointer,

# `ftell` függvény

A fájl pozíció mutató lekérdezése:Deklarációja:

```
long ftell(FILE *fp);
```

Paraméterei:

- **`FILE *fp`** a fájl pointer,

Visszatérési értéke:

- a fájl pozíció érték sikeres esetben,
- -1, vagy **`EOF`** hiba esetén.

# Standard fájlok magasszinten

A **UNIX** logikájára épülő, vagy az ezt valamilyen szinten követő operációs rendszerek három úgynevezett **standard fájlt** használnak

# Standard fájlok magasszinten

A **`UNIX`** logikájára épülő, vagy az ezt valamilyen szinten követő operációs rendszerek három úgynevezett **standard fájlt** használnak, ezek:

- **standard bemenet**, amely általában a billentyűzet a fájl mutató neve **`stdin`**,
- **standard kimenet**, amely általában a képernyő a fájl mutató neve **`stdout`**,
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2). a fájl mutató neve **`stderr`**,

# Standard fájlok magasszinten

A **`UNIX`** logikájára épülő, vagy az ezt valamilyen szinten követő operációs rendszerek három úgynevezett **standard fájlt** használnak, ezek:

- **standard bemenet**, amely általában a billentyűzet a fájl mutató neve **`stdin`**,
- **standard kimenet**, amely általában a képernyő a fájl mutató neve **`stdout`**,
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2). a fájl mutató neve **`stderr`**,

A standard ”állományok” alapértelmezetten meg vannak nyitva, így azonnal használhatók.

# Standard fájlok magasszinten

A **UNIX** logikájára épülő, vagy az ezt valamilyen szinten követő operációs rendszerek három úgynevezett **standard fájlt** használnak, ezek:

- **standard bemenet**, amely általában a billentyűzet a fájl mutató neve **stdin**,
- **standard kimenet**, amely általában a képernyő a fájl mutató neve **stdout**,
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2). a fájl mutató neve **stderr**,

A standard ”állományok” alapértelmezetten meg vannak nyitva, így azonnal használhatók.

Egy üzenet kiírása a képernyőre a következőképpen történhet:

```
fwrite("Hello\n",1,6,stdout);
```

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>
```

A megfelelő header fájl beolvasása.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.      {
4.      FILE *fp;
5.      int r;




15.     }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");




15.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {



12.       }



15.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {
9.        r=getc(fp);


12.       }



15.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.
Az aktuális bájt beolvasásása az **`r`** változóba.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {
9.        r=getc(fp);
10.       if(r<0) break;

12.       }



15.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.
Az aktuális bájt beolvasásása az **`r`** változóba.
Ha hibás, vagy vége kilép a ciklusból.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {
9.        r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);
12.       }



15.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.
Az aktuális bájt beolvasásása az **r** változóba.
Ha hibás, vagy vége kilép a ciklusból.
A **c** kiírása a standard outputra.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {
9.        r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);
12.       }

13.    fclose(fp);

15.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.
Az aktuális bájt beolvasásása az **r** változóba.
Ha hibás, vagy vége kilép a ciklusból.
A **c** kiírása a standard outputra.
A fájl lezárása.

# Példa:

Egy fájl kiírása a standard outputra csak magasszintű fájlkezeléssel.

```
1.  #include <stdio.h>

2.  int main(void)
3.     {
4.     FILE *fp;
5.     int r;
6.     fp=fopen("./text.txt","r");

7.     while(1)
8.        {
9.        r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);
12.       }

13.    fclose(fp);
14.    return 0;
15.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása.
A beolvasó ciklus.
Az aktuális bájt beolvasásása az **`r`** változóba.
Ha hibás, vagy vége kilép a ciklusból.
A **`c`** kiírása a standard outputra.
A fájl lezárása.
A program vége (0-ás hibakóddal).

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;
```

```
16.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;

9.     fp=fopen("./text.txt","w");

16.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása írásra és létrehozásra, a fájl írható és olvasható.

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;

9.     fp=fopen("./text.txt","w");

10.    for(c='0';c<='9';c++)
11.       {



13.       }



16.    }
```

A megfelelő header fájl beolvasása.
A **`main`** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása írásra és létrehozásra, a fájl írható és olvasható.
A kiírás ciklusa, a ciklusváltozó a kiírandó adat.

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;

9.     fp=fopen("./text.txt","w");

10.    for(c='0';c<='9';c++)
11.       {

12.       putc(c,fp)

13.       }



16.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása írásra és létrehozásra, a fájl írható és olvasható.
A kiírás ciklusa, a ciklusváltozó a kiírandó adat.
A **c** változó kiírása a fájlba.

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;

9.     fp=fopen("./text.txt","w");

10.    for(c='0';c<='9';c++)
11.       {

12.       putc(c,fp)

13.       }

14.    fclose(fp);

16.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása írásra és létrehozásra, a fájl írható és olvasható.
A kiírás ciklusa, a ciklusváltozó a kiírandó adat.
A **c** változó kiírása a fájlba.
A fájl lezárása.

# Példa:

Adatok (karakterek) kiírása fájlba.

```
1.  #include <stdio.h>

5.  int main(void)
6.     {
7.     FILE *fp
8.     char c;

9.     fp=fopen("./text.txt","w");

10.    for(c='0';c<='9';c++)
11.       {

12.       putc(c,fp)

13.       }

14.    fclose(fp);
15.    return 0;
16.    }
```

A megfelelő header fájl beolvasása.
A **main** és a függvényen belül a szükséges változók deklarálása.
A fájl megnyitása írásra és létrehozásra, a fájl írható és olvasható.
A kiírás ciklusa, a ciklusváltozó a kiírandó adat.
A **c** változó kiírása a fájlba.
A fájl lezárása.
A program vége (0-ás hibakóddal).

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");
```

A header fájlok, a változók deklarációja és a fájl megnyitása.

```
16.     }
```

%

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");

7.     fseek(fp,-1,SEEK_END);
```

A header fájlok, a változók deklarációja és a fájl megnyitása.
A fájl pozíció mutató pozícionálása a fájl utolsó karakterére.

```
16.     }
```

%

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");

7.     fseek(fp,-1,SEEK_END);

8.     while(1)
9.       {




13.       }


16.     }

%
```

A header fájlok, a változók deklarációja és a fájl megnyitása.
A fájl pozíció mutató pozícionálása a fájl utolsó karakterére.
Az olvasás ciklusa.

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");

7.     fseek(fp,-1,SEEK_END);

8.     while(1)
9.       {
13.       r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);



13.       }



16.     }

%
```

A header fájlok, a változók deklarációja és a fájl megnyitása.
A fájl pozíció mutató pozícionálása a fájl utolsó karakterére.
Az olvasás ciklusa.
Az aktuális bájt beolvasása, a beolvasás ellenőrzése és a bájt kiírása, ha az érvényes. Ha nem az, akkor kilép a ciklusból.

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");

7.     fseek(fp,-1,SEEK_END);

8.     while(1)
9.       {
13.       r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);

12.       fseek(fp,-2,SEEK_CUR);

13.       }



16.     }

%
```

A header fájlok, a változók deklarációja és a fájl megnyitása.
A fájl pozíció mutató pozícionálása a fájl utolsó karakterére.
Az olvasás ciklusa.
Az aktuális bájt beolvasása, a beolvasás ellenőrzése és a bájt kiírása, ha az érvényes. Ha nem az, akkor kilép a ciklusból.
A fájl pozíció mutató újrapozícionálása.

# Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <stdio.h>
2.  int main(void)
3.     {
4.     int r,hnd;
5.     FILE *fp,
6.     fp=fopen("./text.txt","r");

7.     fseek(fp,-1,SEEK_END);

8.     while(1)
9.       {
13.       r=getc(fp);
10.       if(r<0) break;
11.       putc(r,stdout);

12.       fseek(fp,-2,SEEK_CUR);

13.       }

14.     fclose(fp);
15.     return 0;
16.     }
```

%

A header fájlok, a változók deklarációja és a fájl megnyitása.
A fájl pozíció mutató pozícionálása a fájl utolsó karakterére.
Az olvasás ciklusa.
Az aktuális bájt beolvasása, a beolvasás ellenőrzése és a bájt kiírása, ha az érvényes. Ha nem az, akkor kilép a ciklusból.
A fájl pozíció mutató újrapozícionálása.
A fájl lezárása és a program befejezése.

# Példa:

után a fájl pozíció mutató a fájl utolsó bájtjára mutat. Ez azért szükséges, mert a beolvasás első bájtja a fájl utolsó bájtja.

A fájl utolsó pozíciója az utolsó bájt utáni pozíció.

# Példa:

után a fájl pozíció mutató a fájl utolsó bájtjára mutat. Ez azért szükséges, mert a beolvasás első bájtja a fájl utolsó bájtja.

A fájl utolsó pozíciója az utolsó bájt utáni pozíció.

A 12. sorban a

```
fseek(fp,-2,SEEK_CUR);
```

a fájl pozíció mutatót az utoljára beolvasott bájt elé pozícionálja. A beolvasás során az aktuális fájl pozícó érték eggyel növekszik, a beolvasás után ezért kell az beolvasás utáni aktuális pozícióból egy -2 -es módosítás.

# Példa:

után a fájl pozíció mutató a fájl utolsó bájtjára mutat. Ez azért szükséges, mert a beolvasás első bájtja a fájl utolsó bájtja.

A fájl utolsó pozíciója az utolsó bájt utáni pozíció.

A 12. sorban a
`fseek(fp,-2,SEEK_CUR);`

a fájl pozíció mutatót az utoljára beolvasott bájt elé pozícionálja. A beolvasás során az aktuális fájl pozícó érték eggyel növekszik, a beolvasás után ezért kell az beolvasás utáni aktuális pozícióból egy -2 -es módosítás.

*Ha az aktuális pozíció 100, akkor a bájt beolvasása után az fájl pozíció mutató 101 lesz. A következő bájt pozíciója 99. Tehát a 101-es értéket -2 -vel kell módosítani.*

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?

Egy `FILE` típusú pointerrel.

Melyik függvénnyel nyitunk meg egy fájlt magasszinten?

Az `fopen` függvénnyel.

Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.
Mivel tér vissza a `getc` függvény hiba esetén?

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.
Mivel tér vissza a `getc` függvény hiba esetén?
-1 értékkel. Ezért `int` típusú a függvény.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.
Mivel tér vissza a `getc` függvény hiba esetén?
-1 értékkel. Ezért `int` típusú a függvény.
Hogyan írunk ki egy bájtot a megfelelően megnyitott fájlba magasszinten

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.
Mivel tér vissza a `getc` függvény hiba esetén?
-1 értékkel. Ezért `int` típusú a függvény.
Hogyan írunk ki egy bájtot a megfelelően megnyitott fájlba magasszinten
A `putc` függvénnyel.

# Kérdések

Mivel azonosítunk egy fájlt magasszintű fájlkezelés esetén?
Egy `FILE` típusú pointerrel.
Melyik függvénnyel nyitunk meg egy fájlt magasszinten?
Az `fopen` függvénnyel.
Hogyan adjuk meg a megnyitás módját a fájl megnyitásánál magasszinten?
Az úgynevezett mód sztringgel.
Mit ad vissza az `fopen` függvény ha a megnyitás sikertelen?
`NULL` pointert.
Hogyan zárunk le egy fájlt magasszinten?
Az `fclose` függvénnyel, melynek paramétere a `FILE` pointer.
Hogyan olvasunk be egy bájtot a megfelelően megnyitott fájlból magasszinten?
A `getc` függvénnyel.
Mivel tér vissza a `getc` függvény hiba esetén?
-1 értékkel. Ezért `int` típusú a függvény.
Hogyan írunk ki egy bájtot a megfelelően megnyitott fájlba magasszinten
A `putc` függvénnyel.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?

Az `fread` függvénnyel.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?

Az `fread` függvénnyel.

Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.
Honnan tudható meg a fájl pozíció mutató értéke magasszinten?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.
Honnan tudható meg a fájl pozíció mutató értéke magasszinten?
Az **`ftell`** függvénnyel.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.
Honnan tudható meg a fájl pozíció mutató értéke magasszinten?
Az **`ftell`** függvénnyel.
Melyek a standard fájlok?

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.
Honnan tudható meg a fájl pozíció mutató értéke magasszinten?
Az **`ftell`** függvénnyel.
Melyek a standard fájlok?
A **`stdin`** a standard bemenet, **`stdout`**, a standard kimenet, **`stderr`** a standard hiba kimenet.

# Kérdések

Hogyan tudunk nagyobb adatmennyiséget beolvasni a megfelelően megnyitott fájlból magasszinten?
Az **`fread`** függvénnyel.
Hogyan tudunk nagyobb adatmennyiséget kiírni a megfelelően megnyitott fájlba magasszinten?
Az **`fwrite`** függvénnyel.
Hogyan állítjuk a fájl pozíció muatatót magasszinten?
Honnan értelmezhető a fájl pozíció mutató megváltoztatása **`fseek`**esetén?
A fájl elejéről **`SEEK_SET`**, az aktuális pozíciótól **`SEEK_CUR`**, a fájl végétől **`SEEK_END`**.
Az **`fseek`** függvénnyel.
Honnan tudható meg a fájl pozíció mutató értéke magasszinten?
Az **`ftell`** függvénnyel.
Melyek a standard fájlok?
A **`stdin`** a standard bemenet, **`stdout`**, a standard kimenet, **`stderr`** a standard hiba kimenet.