Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Alacsonyszintű fájlkezelés

Dr. Schuster György

2018. január 2.

Alapfogalmak

Fájl: logikailag és fizikailag összetartozó (adat)állomány.

Alapfogalmak

Fájl: logikailag és fizikailag összetartozó (adat)állomány.

A C a fájlt egy egydimenziós bájt tömbként értelmezi.

Alapfogalmak

Fájl: logikailag és fizikailag összetartozó (adat)állomány.

A C a fájlt egy egydimenziós bájt tömbként értelmezi.

Ha fájl hossza `n` bájt, akkor a fájl eleje a `0`-ás pozíciónál kezdődik és a fájl vége az `n-1` pozíción van.

Alapfogalmak

Fájl: logikailag és fizikailag összetartozó (adat)állomány.

A C a fájlt egy egydimenziós bájt tömbként értelmezi.

Ha fájl hossza `n` bájt, akkor a fájl eleje a `0`-ás pozíciónál kezdődik és a fájl vége az `n-1` pozíción van.

A C kétféle fájlkezelést használ:

- alacsonyszintű fájlkezelés,
- magasszintű fájlkezelés.

Alapfogalmak

Fájl: logikailag és fizikailag összetartozó (adat)állomány.

A C a fájlt egy egydimenziós bájt tömbként értelmezi.

Ha fájl hossza `n` bájt, akkor a fájl eleje a `0`-ás pozíciónál kezdődik és a fájl vége az `n-1` pozíción van.

A C kétféle fájlkezelést használ:

- alacsonyszintű fájlkezelés,
- magasszintű fájlkezelés.

Valamikor a két módszer különbözött. A modern operációs rendszereknél a két mód gyakorlatilag azonos, szolgáltatásaikban térnek el.

Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Tehát, ha beírunk egy bájtot a fájlba, akkor a kérdéses bájt arra a helyre kerül, amit a művelet előtt a fájl pozíció mutató adott meg.

Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Tehát, ha beírunk egy bájtot a fájlba, akkor a kérdéses bájt arra a helyre kerül, amit a művelet előtt a fájl pozíció mutató adott meg.

A fájl pozíció mutató értéke pedig eggyel növekszik.

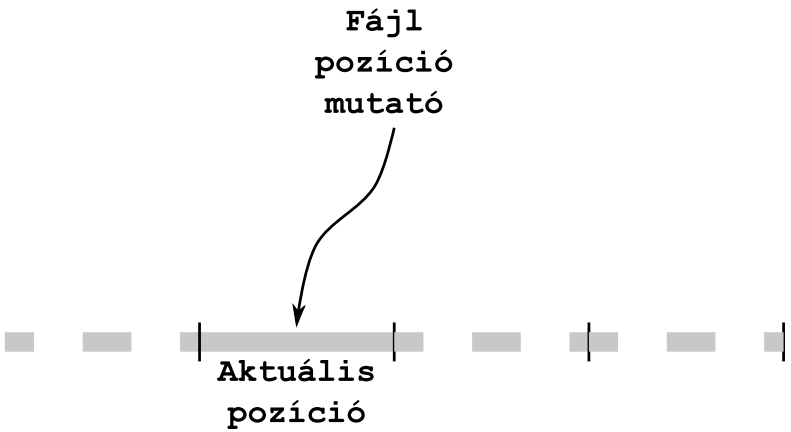
Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Tehát, ha beírunk egy bájtot a fájlba, akkor a kérdéses bájt arra a helyre kerül, amit a művelet előtt a fájl pozíció mutató adott meg.

A fájl pozíció mutató értéke pedig eggyel növekszik.



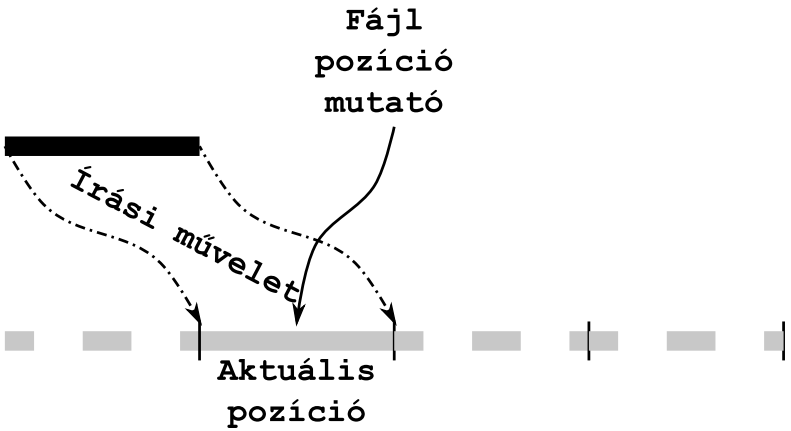
Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Tehát, ha beírunk egy bájtot a fájlba, akkor a kérdéses bájt arra a helyre kerül, amit a művelet előtt a fájl pozíció mutató adott meg.

A fájl pozíció mutató értéke pedig eggyel növekszik.



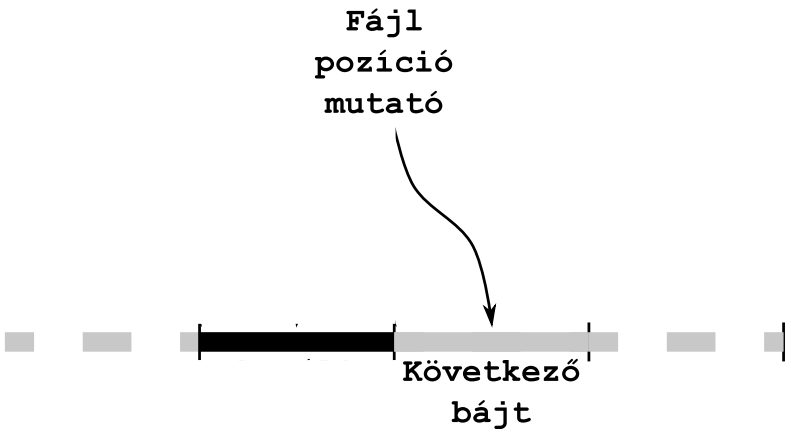
Fájl pozíció mutató

A **fájl pozíció mutató** az a pozíció érték, amely azt mondja meg, hogy a következő művelet a fájl melyik bájtján kerül végrehajtásra.

A fájl pozíció mutató automatikusan csak növekvő irányba mozog. Ha a fájl pozíció mutató értékét mi szeretnénk állítani, akkor függvényt kell hívnunk.

Tehát, ha beírunk egy bájtot a fájlba, akkor a kérdéses bájt arra a helyre kerül, amit a művelet előtt a fájl pozíció mutató adott meg.

A fájl pozíció mutató értéke pedig eggyel növekszik.



Alacsonyszintű fájlkezelés

Az alacsony fájlkezelésnél az alkalmazott header fájlok különbözők:

Linux esetén:

```
sys/types.h
sys/stat.h
fcntl.h
unistd.h
```

Alacsonyszintű fájlkezelés

Az alacsony fájlkezelésnél az alkalmazott header fájlok különbözők:

Linux esetén:

```
sys/types.h
sys/stat.h
fcntl.h
unistd.h
```

Windows esetén:

```
sys/types.h
sys/stat.h
fcntl.h
io.h
```

Alacsonyszintű fájlkezelés

Az alacsony fájlkezelésnél az alkalmazott header fájlok különbözők:

Linux esetén:
```
sys/types.h
sys/stat.h
fcntl.h
unistd.h
```

Windows esetén:
```
sys/types.h
sys/stat.h
fcntl.h
io.h
```

Alacsonszintű fájlkezelésnél a fájl azonosítása egy `int` típusú változóval történik.

Alacsonyszintű fájlkezelés

Az alacsony fájlkezelésnél az alkalmazott header fájlok különbözők:

Linux esetén:
```
sys/types.h
sys/stat.h
fcntl.h
unistd.h
```

Windows esetén:
```
sys/types.h
sys/stat.h
fcntl.h
io.h
```

Alacsonszintű fájlkezelésnél a fájl azonosítása egy `int` típusú változóval történik.

A továbbiakban ez a `int hnd;` változó.

Alacsonyszintű fájlkezelés

Az alacsony fájlkezelésnél az alkalmazott header fájlok különbözők:

Linux esetén:
```
sys/types.h
sys/stat.h
fcntl.h
unistd.h
```

Windows esetén:
```
sys/types.h
sys/stat.h
fcntl.h
io.h
```

Alacsonszintű fájlkezelésnél a fájl azonosítása egy `int` típusú változóval történik.

A továbbiakban ez a `int hnd;` változó.

Tipikus elnevezése **fájl kezelő**, vagy **fájl leíró**.

open

Ha egy fájlt használni szeretnénk, akkor azt előzőleg meg kell nyitni. Erre szolgál az **open** függvény. Deklarációja:

open

Ha egy fájlt használni szeretnénk, akkor azt előzőleg meg kell nyitni. Erre szolgál az **open** függvény. Deklarációja:

```
int open(const char *pathname, int flags);

int open(const char *pathname, int flags, mode_t mode);
```

A függvény visszatérési értéke a fájl kezelő, éréke:

- pozitív szám a fájl megnyitása sikeres,
- -1 másnéven **EOF** a fájl megnyitás hibás.

A továbbiakban ezzel a vissza adott értékkel hivatkozhatunk a fájlra.

open

A függvény paraméterei:

- **`const char *pathname`** a fájl elérési útja, pl:

  ```
  hnd=open("./text.txt", ...
  ```

- **`int flags`** a fájl megnyitási paraméterei,
- **`mode_t mode`** a fájl beállítandó attribútumai. Ennek megadása nem kötelező

int flags

A legfontosabb megnyitási paraméterek:

- **O_RDONLY** megnyitás csak olvasásra,
- **O_WRONLY** megnyitás csak írásra,
- **O_RDWR** megnyitás olvasásra és írásra,
- **O_APPEND** megnyitás hozzáfűzésre (ez alapvetően egy írásra nyitás, csak a fájl pozíció mutató az utolsó utáni bájtra mutat a megnyitás után),
- **O_CREAT** a fájl létrehozása,
- **O_EXCL** ha a fájl már létezik és ezt a flaget beállítottuk az **open** hibával tér vissza,

int flags

- **O_NONBLOCK** a fájl megnyitásánál úgynevezett nem blokkoló módot ír elő (részletesen a **read** függvénynél), a Windows nem tudja,
- **O_TRUNC** írás jellegű megnyitás esetén a fájl hosszát nullára vágja,
- **O_BINARY** a fájl megnyitása bináris módban (Windows esetén ajánlott, bináris állomány kezelésénél),
- **O_TEXT** a fájl megnyitása szöveges módban (Windows esetén ajánlott ez az alapeset),

Ezek a szimbólumok flagek, ha értelemszerűen akarjuk beállítani a megnyitási módot, akkor bináris vagy operátorral '|' tehetjük meg, pl:

```
hnd=open("./text.txt", O_RDONLY | O_NONBLOCK, ...
```

A **text.txt** fájlt csak olvasásra, nem blokkoló módon szeretnénk megnyitni.

mode_t mode

A **mode_t** típus egy átdefiniált **unsigned**.

Ennek a paraméternek a fájl létrehozásánál van értelme. Beállítja a fájl elérési jogait.

- **S_IRWXU** a tulajdonosnak olvasási, írási és végrehajtási joga van,
- **S_IRUSR** a tulajdonosnak olvasási joga van,
- **S_IWUSR** a tulajdonosnak írási joga van,
- **S_IXUSR** a tulajdonosnak végrehajtási joga van,
- **S_IRWXG** a tulajdonos csoportjának olvasási, írási és végrehajtási joga van,
- **S_IRGRP** a tulajdonos csoportjának olvasási joga van,
- **S_IWGRP** a tulajdonos csoportjának írási joga van,
- **S_IXGRP** a tulajdonos csoportjának végrehajtási joga van,

mode_t mode

- **S_IRWXO** a többieknek olvasási, írási és végrehajtási joga van,
- **S_IROTH** a többieknek olvasási joga van,
- **S_IWOTH** a többieknek írási joga van,
- **S_IXOTH** a többieknek végrehajtási joga van,
- **S_IWRITE** Windows-os környezetben írási jog (gcc ismeri),
- **S_IREAD** Windows-os környezetben olvasási jog (gcc ismeri).

A megfelelő flageket itt is a bináris vagy operátorral kapcsoljuk össze, pl:

```
hnd=open("./text.txt", O_CREAT, S_IRWXU | S_IXGRP |
S_IXOTH);
```

Vagyis a létrehozott állomány a tulajdonos számára minden, a csoportja számára és többiek számára csak végrehajtási jogot kap.

Példák az `open` függvényre

Egy állomány megnyitása olvasásra:

```
{
int hnd;
hnd=open("./text.txt",O_RDONLY);
      ⋮
```

Egy állomány létrehozása és megnyitása írásra:

```
{
int hnd;
hnd=open("./text.txt",O_CREAT | O_WRONLY,S_IRWXU);
      ⋮
```

Megjegyzés: a többi elérési jog a rendszer beállításától függ.

`close` függvény

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

```
int close(int hnd);
```

A függvény paramétere a fájl kezelő.

Visszatérési értéke:

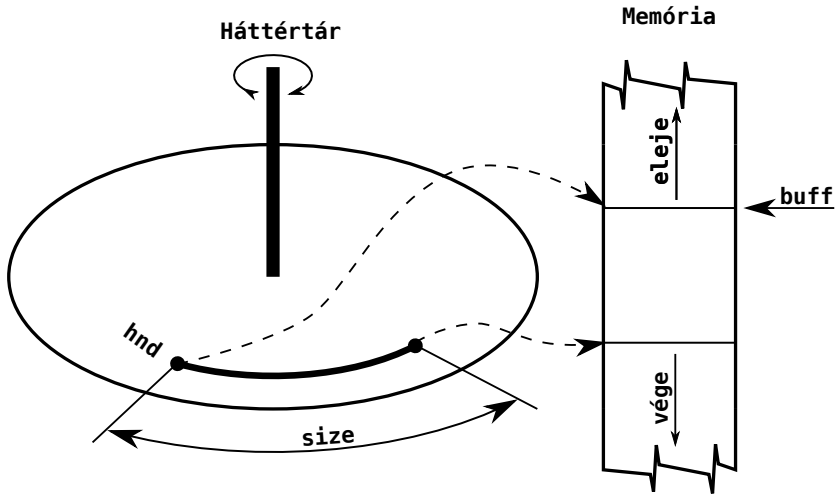
- 0, ha a fájl lezárása sikeres,
- -1, vagy **`EOF`**, ha sikertelen.

`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.

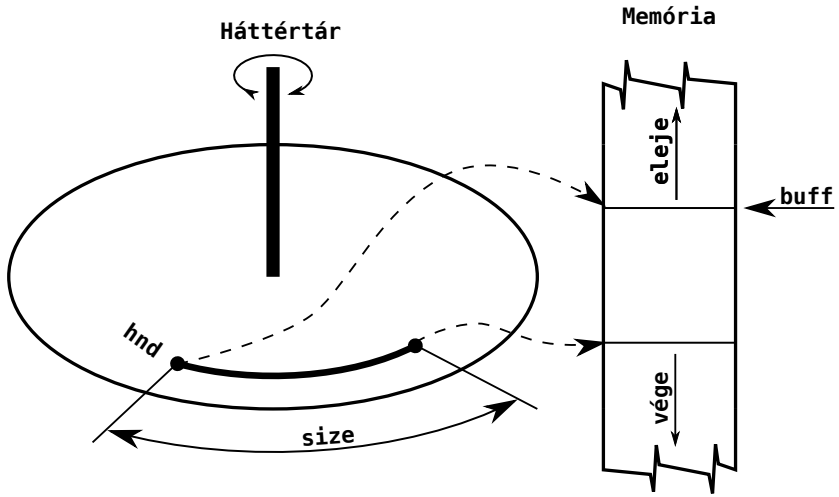
`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.



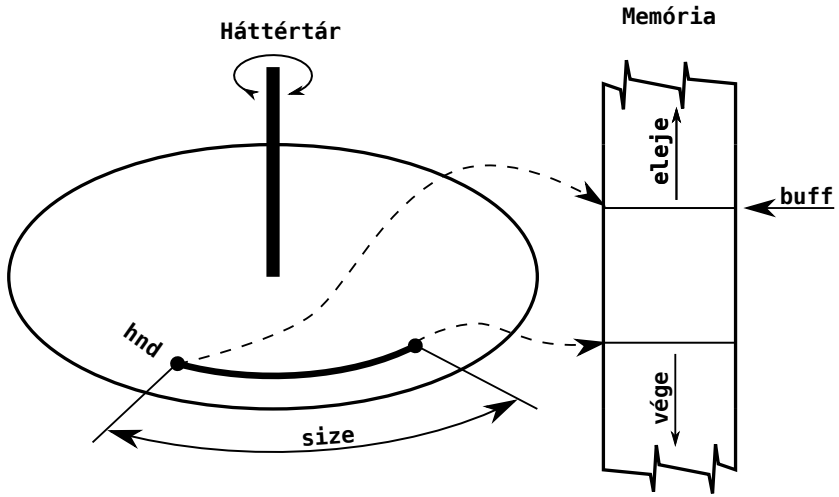
`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.
Deklarációja:



`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.
Deklarációja:



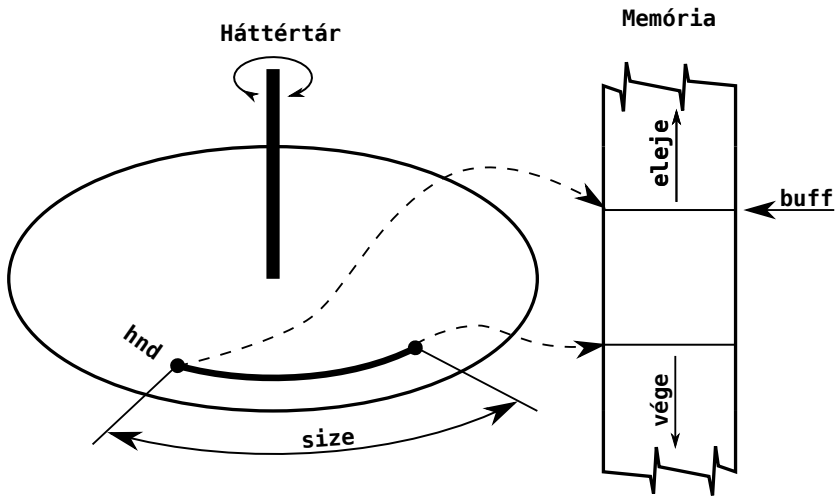
`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.
Deklarációja:

```
ssize_t read(int hnd, void *buff, size_t size);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`void *buff`** a memória cím, ahova olvasunk,
- **`size_t size`** az olvasandó bájtok száma.



`read` függvény

A függvény a megnyitott fájlból egy megadott memória területre másol.
Deklarációja:

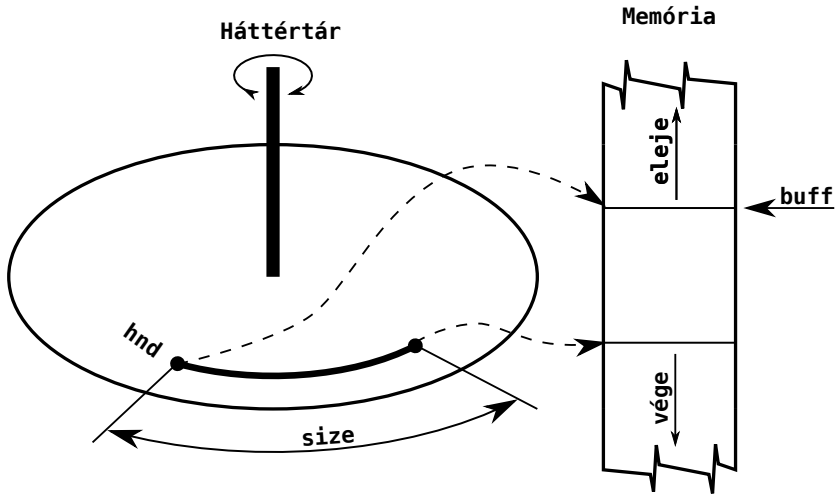
```
ssize_t read(int hnd, void *buff, size_t size);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`void *buff`** a memória cím, ahova olvasunk,
- **`size_t size`** az olvasandó bájtok száma.

Visszatérési értéke:

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

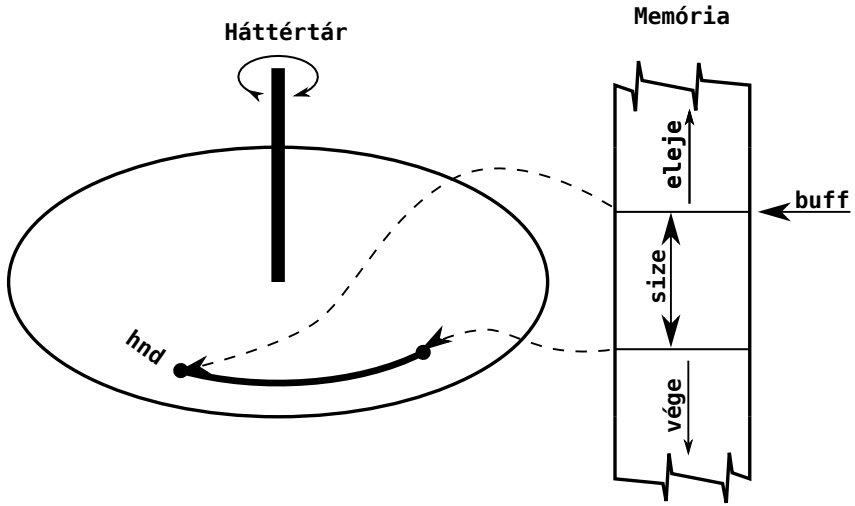


`write` függvény

A függvény a memóriából egy írásra megnyitott fájlba másol.

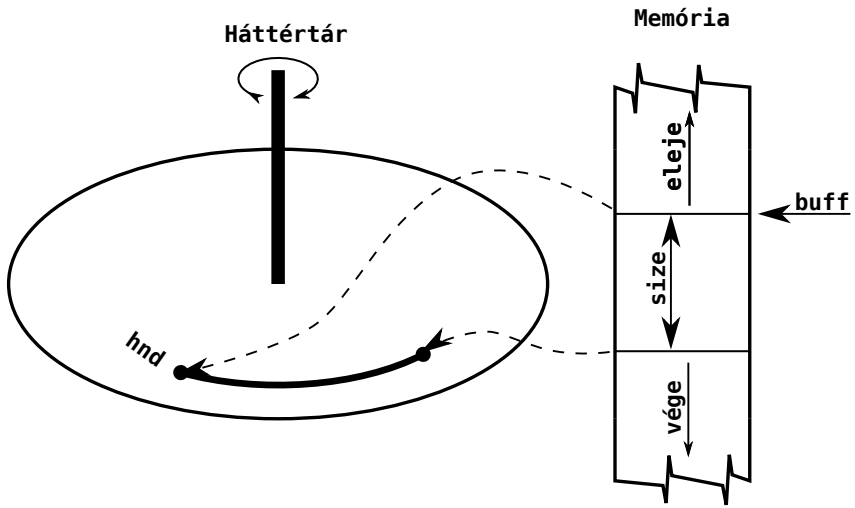
`write` függvény

A függvény a memóriából egy írásra megnyitott fájlba másol.



`write` függvény

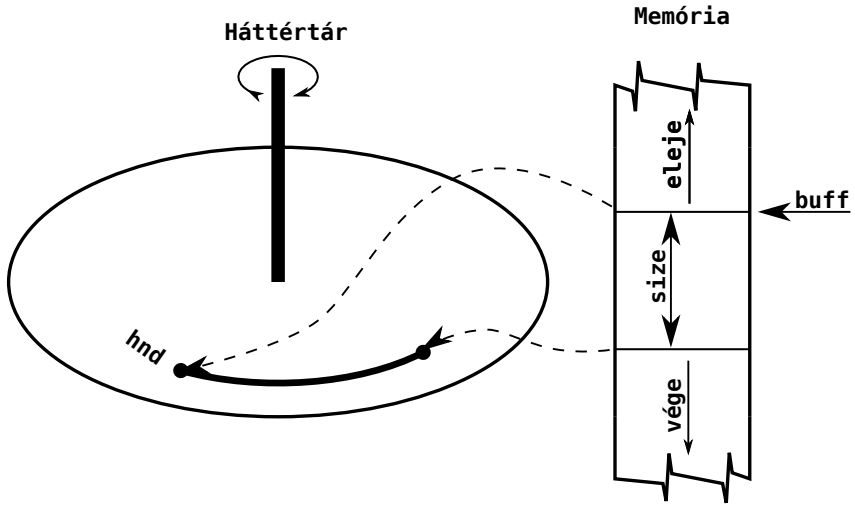
A függvény a memóriából egy írásra megnyitott fájlba másol.
Deklarációja:



`write` függvény

A függvény a memóriából egy írásra megnyitott fájlba másol.
Deklarációja:

```
ssize_t write(int hnd, void *buff, size_t size);
```



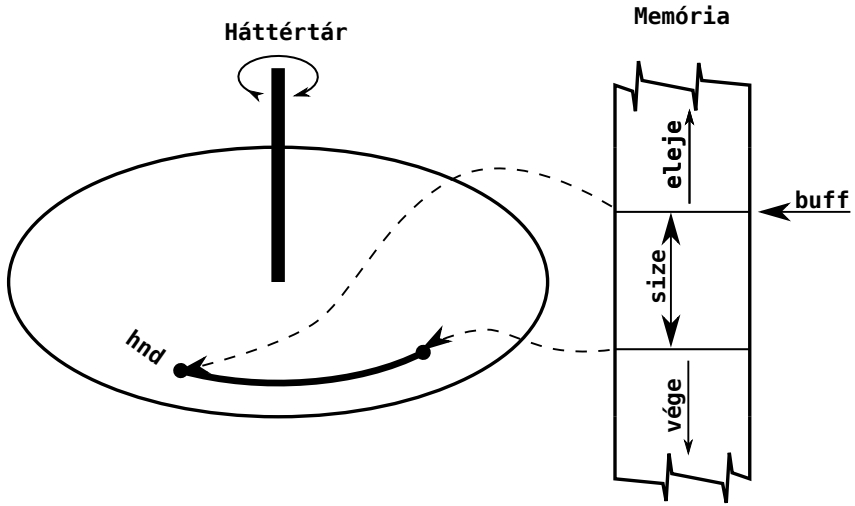
`write` függvény

A függvény a memóriából egy írásra megnyitott fájlba másol.
Deklarációja:

```
ssize_t write(int hnd, void *buff, size_t size);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`void *buff`** a memória cím, ahonnan írunk,
- **`size_t size`** a kiírandó bájtok száma.



`write` függvény

A függvény a memóriából egy írásra megnyitott fájlba másol.
Deklarációja:

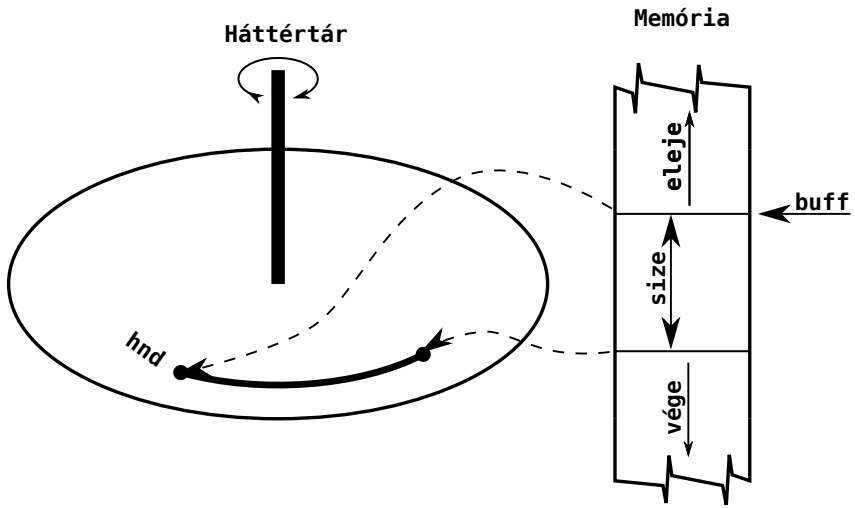
```
ssize_t write(int hnd, void *buff, size_t size);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`void *buff`** a memória cím, ahonnan írunk,
- **`size_t size`** a kiírandó bájtok száma.

Visszatérési értéke:

- a fájlba írt bájtok száma, ha az írás sikeres,
- -1, vagy **`EOF`**, ha sikertelen.



Standard fájlok

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

Standard fájlok

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 leíró értéke 0),
- **standard kimenet**, amely általában a képernyő (a leíró értéke 1),
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2).

Standard fájlok

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 leíró értéke 0),
- **standard kimenet**, amely általában a képernyő (a leíró értéke 1),
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2).

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

Standard fájlok

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 leíró értéke 0),
- **standard kimenet**, amely általában a képernyő (a leíró értéke 1),
- **stabdard hiba**, amely általában szintén a képernyő (a leíró értéke 2).

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

Egy üzenet kiírása a képernyőre a következőképpen történhet:

```
write(1,"Hello\n",6);
```

`lseek` függvény

A fájl pozíció mutató beállítására az **`lseek`** függvény szolgál.

`lseek` függvény

A fájl pozíció mutató beállítására az **`lseek`** függvény szolgál. Deklarációja:

```
off_t lseek(int hnd, off_t offset, int whence);
```

`lseek` függvény

A fájl pozíció mutató beállítására az **`lseek`** függvény szolgál. Deklarációja:

```
off_t lseek(int hnd, off_t offset, int whence);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`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).

`lseek` függvény

A fájl pozíció mutató beállítására az **`lseek`** függvény szolgál. Deklarációja:

```
off_t lseek(int hnd, off_t offset, int whence);
```

Paraméterei:

- **`int hnd`** a fájl leíró,
- **`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.

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
```

A megfelelő header fájlok beolvasása.

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.      {
7.      int r,hnd;
8.      char c;

18.     }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);




18.    }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {



15.       }



18.    }
```

A megfelelő header fájlok 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 alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {
12.       r=read(hnd,&c,1);


15.       }



18.    }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {
12.       r=read(hnd,&c,1);
13.       if(r<1) break;

15.       }


18.    }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {
12.       r=read(hnd,&c,1);
13.       if(r<1) break;
14.       write(1,&c,1);
15.       }



18.    }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {
12.       r=read(hnd,&c,1);
13.       if(r<1) break;
14.       write(1,&c,1);
15.       }

16.    close(hnd);

18.    }
```

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

Példa:

Egy fájl kiírása a standard outputra csak alacsonyszintű fájlkezeléssel.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    while(1)
11.       {
12.       r=read(hnd,&c,1);
13.       if(r<1) break;
14.       write(1,&c,1);
15.       }

16.    close(hnd);
17.    return 0;
18.    }
```

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

Példa:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.      {
7.      int hnd;
8.      char c;
```

```
16.     }
```

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

Példa:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int hnd;
8.     char c;

9.     hnd=open("./text.txt",O_WRONLY|O_CREAT,S_IWRITE|S_IREAD);

16.    }
```

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

Példa:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int hnd;
8.     char c;

9.     hnd=open("./text.txt",O_WRONLY|O_CREAT,S_IWRITE|S_IREAD);


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



13.       }



16.    }
```

A megfelelő header fájlok 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:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int hnd;
8.     char c;

9.     hnd=open("./text.txt",O_WRONLY|O_CREAT,S_IWRITE|S_IREAD);


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

12.       write(hnd,&c,1);

13.       }



16.    }
```

A megfelelő header fájlok 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:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int hnd;
8.     char c;

9.     hnd=open("./text.txt",O_WRONLY|O_CREAT,S_IWRITE|S_IREAD);


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

12.       write(hnd,&c,1);

13.       }

14.    close(hnd);

16.    }
```

A megfelelő header fájlok 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:

Fájl írása.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>

5.  int main(void)
6.     {
7.     int hnd;
8.     char c;

9.     hnd=open("./text.txt",O_WRONLY|O_CREAT,S_IWRITE|S_IREAD);


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

12.       write(hnd,&c,1);

13.       }

14.    close(hnd);
15.    return 0.
16.    }
```

A megfelelő header fájlok 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 <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);
```

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

```
20.    }
```

Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);
```

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

```
20.     }
```

Példa:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);

11.    while(1)
12.       {




17.       }



20.    }
```

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 <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);

11.    while(1)
12.       {
13.       r=read(hnd,&c,1);
14.       if(r<1) break;
15.       write(1,&c,1);



17.       }



20.    }
```

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 <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);

11.    while(1)
12.       {
13.       r=read(hnd,&c,1);
14.       if(r<1) break;
15.       write(1,&c,1);

16.       lseek(hnd,-2,SEEK_CUR);

17.       }



20.    }
```

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 <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);

11.    while(1)
12.       {
13.       r=read(hnd,&c,1);
14.       if(r<1) break;
15.       write(1,&c,1);

16.       lseek(hnd,-2,SEEK_CUR);

17.       }

18.     close(hnd);
19.     return 0;
20.     }
```

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:

Fájl kiírása fordított sorrendben a standard outputra.

```
1.  #include <sys/stat.h>
2.  #include <sys/types.h>
3.  #include <fcntl.h>
4.  #include <unistd.h>
5.  int main(void)
6.     {
7.     int r,hnd;
8.     char c;
9.     hnd=open("./text.txt",O_RDONLY);

10.    lseek(hnd,-1,SEEK_END);

11.    while(1)
12.       {
13.       r=read(hnd,&c,1);
14.       if(r<1) break;
15.       write(1,&c,1);

16.       lseek(hnd,-2,SEEK_CUR);

17.       }

18.     close(hnd);
19.     return 0;
20.     }
```

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.
Részletes magyarázat a következő oldalon.

Példa:

A 10. sorban az

```
lseek(hnd,-1,SEEK_END);
```

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:

A 10. sorban az

```
lseek(hnd,-1,SEEK_END);
```

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 16. sorban a

```
lseek(hnd,-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:

A 10. sorban az
`lseek(hnd,-1,SEEK_END);`

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 16. sorban a
`lseek(hnd,-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.

Példa:

A fájl pozíció mutató lekérdezése:

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **`lseek`** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **`lseek`** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

A függvény visszatérésiértéke a fájl hossza, paramétere a fájl leíró.

```
1.   long flength(int hnd)
2.      {
8.      }
```

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

```
1.   long flength(int hnd)
2.      {

3.      long tmp,r;

8.      }
```

A függvény visszatérésiértéke a fájl hossza, paramétere a fájl leíró.
A **tmp** változó átmenetileg tárolja az aktuális fájl pozíciót, az **r** változó a visszatérési érték.

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

```
1.   long flength(int hnd)
2.      {

3.      long tmp,r;

4.      tmp=lseek(hnd,0,SEEK_CUR);




8.      }
```

A függvény visszatérésiértéke a fájl hossza, paramétere a fájl leíró.
A **tmp** változó átmenetileg tárolja az aktuális fájl pozíciót, az **r** változó a visszatérési érték.
Elteszi az aktuális fájl pozíciót.

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

```
1.   long flength(int hnd)
2.      {
3.      long tmp,r;
4.      tmp=lseek(hnd,0,SEEK_CUR);
5.      r=lseek(hnd,0,SEEK_END);
8.      }
```

A függvény visszatérésiértéke a fájl hossza, paramétere a fájl leíró.
A **tmp** változó átmenetileg tárolja az aktuális fájl pozíciót, az **r** változó a visszatérési érték.
Elteszi az aktuális fájl pozíciót.
Elmegy a fájl végére és elteszi az ekkori fájl pozíciót.

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

```
1.   long flength(int hnd)
2.      {

3.      long tmp,r;

4.      tmp=lseek(hnd,0,SEEK_CUR);

5.      r=lseek(hnd,0,SEEK_END);

6.      lseek(hnd,tmp,SEEK_SET);



8.      }
```

A függvény visszatérésiértéke a fájl hossza, paramétere a fájl leíró.
A **tmp** változó átmenetileg tárolja az aktuális fájl pozíciót, az **r** változó a visszatérési érték.
Elteszi az aktuális fájl pozíciót.
Elmegy a fájl végére és elteszi az ekkori fájl pozíciót.
Visszaállítja a híváskori fájl pozíciót.

Példa:

A fájl pozíció mutató lekérdezése:

```
lgt=lseek(hnd,0,SEEK_CUR);
```

Adott fájl pozícióról nem módosítjuk a fájl pozíció mutatót. Ekkor az **lseek** függvény az aktuális fájl pozíció mutató értékkel tér vissza.

Egy függvény, amely megadja a fájl hosszát:

```
1.   long flength(int hnd)
2.      {
3.      long tmp,r;
4.      tmp=lseek(hnd,0,SEEK_CUR);
5.      r=lseek(hnd,0,SEEK_END);
6.      lseek(hnd,tmp,SEEK_SET);
7.      return r;
8.      }
```

A függvény visszatérésíértéke a fájl hossza, paramétere a fájl leíró.
A **tmp** változó átmenetileg tárolja az aktuális fájl pozíciót, az **r** változó a visszatérési érték.
Elteszi az aktuális fájl pozíciót.
Elmegy a fájl végére és elteszi az ekkori fájl pozíciót.
Visszaállítja a híváskori fájl pozíciót.
Visszatér a fájl hosszával.

Kérdések

Mi a fájl definíciója?

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Mi az a fájl pozíció mutató?

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Mi az a fájl pozíció mutató?

Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Mi az a fájl pozíció mutató?

Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.

Hogyan változik a a fájl pozíció mutató?

Kérdések

Mi a fájl definíciója?
A fájl egy logikailag és fizikailag összetartozó adatállomány.
Hogyan kezeli a C a fájlt?
Mint egy egydimenziós bájt tömböt.
Mi az a fájl pozíció mutató?
Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.
Hogyan változik a a fájl pozíció mutató?
A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.

Kérdések

Mi a fájl definíciója?
A fájl egy logikailag és fizikailag összetartozó adatállomány.
Hogyan kezeli a C a fájlt?
Mint egy egydimenziós bájt tömböt.
Mi az a fájl pozíció mutató?
Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.
Hogyan változik a a fájl pozíció mutató?
A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.
Hogyan azonosítunk egy fájlt a művelet során alacsonyszintű fájlkezelés esetén?

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Mi az a fájl pozíció mutató?

Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.

Hogyan változik a a fájl pozíció mutató?

A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.

Hogyan azonosítunk egy fájlt a művelet során alacsonyszintű fájlkezelés esetén?

Egy `int` típusú változó segítségével.

Kérdések

Mi a fájl definíciója?

A fájl egy logikailag és fizikailag összetartozó adatállomány.

Hogyan kezeli a C a fájlt?

Mint egy egydimenziós bájt tömböt.

Mi az a fájl pozíció mutató?

Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.

Hogyan változik a a fájl pozíció mutató?

A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.

Hogyan azonosítunk egy fájlt a művelet során alacsonyszintű fájlkezelés esetén?

Egy `int` típusú változó segítségével.

Mit jelent a fájl megnyitás fogalom?

Kérdések

Mi a fájl definíciója?
A fájl egy logikailag és fizikailag összetartozó adatállomány.
Hogyan kezeli a C a fájlt?
Mint egy egydimenziós bájt tömböt.
Mi az a fájl pozíció mutató?
Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.
Hogyan változik a a fájl pozíció mutató?
A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.
Hogyan azonosítunk egy fájlt a művelet során alacsonyszintű fájlkezelés esetén?
Egy `int` típusú változó segítségével.
Mit jelent a fájl megnyitás fogalom?
A fájlt a rendszer számára elérhetővé teszi.

Kérdések

Mi a fájl definíciója?
A fájl egy logikailag és fizikailag összetartozó adatállomány.
Hogyan kezeli a C a fájlt?
Mint egy egydimenziós bájt tömböt.
Mi az a fájl pozíció mutató?
Az a pozíció érték, amely azt mutatja meg, hogy a következő művelet a fájl hányadik bájtján hajtódik végre.
Hogyan változik a a fájl pozíció mutató?
A fájl pozíció mutató magától csak pozitív irányba változik a műveletek során, ha módosítani akarjuk, akkor külön függvény kell.
Hogyan azonosítunk egy fájlt a művelet során alacsonyszintű fájlkezelés esetén?
Egy `int` típusú változó segítségével.
Mit jelent a fájl megnyitás fogalom?
A fájlt a rendszer számára elérhetővé teszi.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az `open` függvény.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az `open` függvény.

Milyen értékkel tér vissza az `open` függvény hibás megnyitás esetén?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az `open` függvény.

Milyen értékkel tér vissza az `open` függvény hibás megnyitás esetén?

A visszatérési érték `-1`, vagy másnéven `EOF`.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az `open` függvény.

Milyen értékkel tér vissza az `open` függvény hibás megnyitás esetén?

A visszatérési érték `-1`, vagy másnéven `EOF`.

Mit jelent a `O_RDONLY` megnyitás?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az `open` függvény.

Milyen értékkel tér vissza az `open` függvény hibás megnyitás esetén?

A visszatérési érték `-1`, vagy másnéven `EOF`.

Mit jelent a `O_RDONLY` megnyitás?

A fájl csak olvasásra lesz megnyitva.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az **`open`** függvény.

Milyen értékkel tér vissza az **`open`** függvény hibás megnyitás esetén?

A visszatérési érték **`-1`**, vagy másnéven **`EOF`**.

Mit jelent a **`O_RDONLY`** megnyitás?

A fájl csak olvasásra lesz megnyitva.

Mi a hatása a **`O_APPEND`** megnyitásnak?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?

Az **open** függvény.

Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?

A visszatérési érték **-1**, vagy másnéven **EOF**.

Mit jelent a **O_RDONLY** megnyitás?

A fájl csak olvasásra lesz megnyitva.

Mi a hatása a **O_APPEND** megnyitásnak?

A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.
Nem létező fájl esetén milyen flag-eket kell használni, ha írni akarjuk?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.
Nem létező fájl esetén milyen flag-eket kell használni, ha írni akarjuk?
O_CREAT és **O_WRONLY**, továbbá szükséges a **S_IWRITE** attribútum.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.
Nem létező fájl esetén milyen flag-eket kell használni, ha írni akarjuk?
O_CREAT és **O_WRONLY**, továbbá szükséges a **S_IWRITE** attribútum.
Hogyan lehet lezárni egy megnyitott fájlt alacsony szinten?

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.
Nem létező fájl esetén milyen flag-eket kell használni, ha írni akarjuk?
O_CREAT és **O_WRONLY**, továbbá szükséges a **S_IWRITE** attribútum.
Hogyan lehet lezárni egy megnyitott fájlt alacsony szinten?
A **close** függvénnyel, melynek paramétere a fájlt azonosító leíró.

Kérdések

Milyen függvény nyit meg alacsony szinten egy fájlt?
Az **open** függvény.
Milyen értékkel tér vissza az **open** függvény hibás megnyitás esetén?
A visszatérési érték **-1**, vagy másnéven **EOF**.
Mit jelent a **O_RDONLY** megnyitás?
A fájl csak olvasásra lesz megnyitva.
Mi a hatása a **O_APPEND** megnyitásnak?
A fájl írásra lesz megnyitva és a fájl pozíció mutató a fájl utolsó bájtja után lesz beállítva.
Mire jók a **mode** paraméter?
A fájl attribútuma lesz beállítva.
Nem létező fájl esetén milyen flag-eket kell használni, ha írni akarjuk?
O_CREAT és **O_WRONLY**, továbbá szükséges a **S_IWRITE** attribútum.
Hogyan lehet lezárni egy megnyitott fájlt alacsony szinten?
A **close** függvénnyel, melynek paramétere a fájlt azonosító leíró.

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?

A **`read`** függvénnyel.

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?

A **`read`** függvénnyel.

Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.
Hogyan tudjuk a fájl pozíció mutatót beállítani?

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.
Hogyan tudjuk a fájl pozíció mutatót beállítani?
Az **`lseek`** függvénnyel.

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.
Hogyan tudjuk a fájl pozíció mutatót beállítani?
Az **`lseek`** függvénnyel.
Hogyan lehet megtudni a fájl pozíció mutató értékét?

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.
Hogyan tudjuk a fájl pozíció mutatót beállítani?
Az **`lseek`** függvénnyel.
Hogyan lehet megtudni a fájl pozíció mutató értékét?
Két lehetőség van, ha van akkor a **`tell`** függvény használatával, ha ez nincs, akkor az **`lseek`** függvénnyel: **`fpos=lseek(hnd,0,SEEK_CUR);`**

Kérdések

Melyik függvénnyel lehet adatot beolvasni a fájlból a memóriába alacsony szinten?
A **`read`** függvénnyel.
Hogyan lehet adatot kiíratni a memóriából fájlba alacsony szinten?
a **`write`** függvénnyel.
Hogyan tudjuk a fájl pozíció mutatót beállítani?
Az **`lseek`** függvénnyel.
Hogyan lehet megtudni a fájl pozíció mutató értékét?
Két lehetőség van, ha van akkor a **`tell`** függvény használatával, ha ez nincs, akkor az **`lseek`** függvénnyel: **`fpos=lseek(hnd,0,SEEK_CUR);`**