JEGYZŐKÖNYV

Operációs Rendszerek BSc

2021 tavaszi féléves feladat

Készítette: **Koics Marcell**
Neptunkód: **BYED0R**

A feladat leírása:

Írjon egy C programot, amely létrehoz egy osztott memória szegmenset és majd rácsatlakozik. Továbbá egy másik program olvasson be 3 számot egy file-ból (amik a háromszög oldalainak hosszát jelentik) az osztott memóriába és döntse el, hogy szerkeszthető-e belőlük háromszög A döntési eredmény a file kimeneten, ha készíthető háromszög van kerülete, illetve területe, ha nincs akkor ezekre -1-et ad vissza. Az adatokat és az eredményt egy fájl kimeneten adjuk vissza.

A feladat elkészítésének lépései:

1. Az shmcreate.c elkészítése.

Ez a program létrehozza a memória szegmenset, kiolvassa a szegmensből a 3 számot, amit egy másik program írt bele és meghatározza, hogy a 3 számból lehet-e háromszöget készíteni.

Először is, létrehoztuk a változókat, amikre szükség lehet:

- *shmid* – egész típus, az osztott memória szegmens azonosítója
- *key* – definiált típus (key_t), egy egyedi kulcs, aminek segítségével létrehozzuk a memória szegmenset; értéke az SHMKEY konstans által definiált érték
- *size* – egész típus, az osztott memória szegmens mérete byte-ban megadva; értéke 512 byte
- *shmflag* – egész típus, a flag határozza meg az osztott memória hozzáférhetőségét

Az osztott memória létrehozása előtt megnézzük létezik-e már 512 byte méretben ilyen szegmens.. Ezt úgy csináljuk, hogy az *shmflag*-et 0-ra állítjuk (a 0-s flag a readonly hozzáférést jelöli) , majd shmget() rendszerhívással lekérdezzük, hogy van-e ilyen osztott mem. szegmens. A függvénynek paraméterben átadjuk a *key*-t, a *size*-t és az *shmflag*-et. Az shmget() egy egész értékkel tér vissza és ha ez kisebb, mint 0, akkor az azt jelenti, hogy nincsen még ilyen szegmens létrehozva.
Így nyugodtan létre is hozzuk úgy, hogy először az *shmflag* értékét módosítjuk *00666* | *IPC_CREATE*-re. A 00666 a szegmenshez való hozzáférést határozza meg, az IPC_CREATE paranccsal pedig azt mondjuk, hogy létre akarjuk hozni ezt a szegmenset. Az shmget() rendszerhívással, paraméterben átadva a kulcsot (key), a méretet (size) és a flag-et (shmflag), létre is hozzuk az osztott memória szegmenset, a visszatérési értékét, pedig az *shmid* változóban tároljuk. Ez lesz az osztott mem. azonosítója A gyakorlatban ez így néz ki:

shmid = shmget(key,size,shmflag);

Ha az shmid kisebb, mint 0, akkor valami hiba történhetett az shmget() hívásakor. Miután létrehoztuk, kiírjuk a képernyőre az osztott memória azonosítóját (shmid).

Ezek után a programot, ami létrehozta ezt az osztott memóriát, hozzáadjuk a szegmenshez (attach). Az shmat() függvény, paraméterként átadva neki az *shmid*-t, egy NULL-t és a *shmflag*-et, visszaad egy memóriacímet és ezen a címen helyezkedik majd el a mem. szegmens. Erre a memóriacímre egy pointert állítunk, hogy a későbbiekben írni/olvasni tudjunk a memóriának ezen tartományán.. Ez az egész így néz ki a program kódjában:

int shm = (int*)shmat(shmid,NULL,shmflag);*

Az shmat() 2. paraméterének egy memóriacímet kell megadni (pointert, ami memóriacímre mutat), de mi NULL-t adtunk meg(nullpointer), ami azt jelenti, hogy az operációs rendszerre bízzuk az osztott memória szegmens elhelyezését a memóriában.

Int típusú pointert állítottunk rá a mem. címre, mert a későbbiekben egész számokat fogunk itt tárolni. Az shmat() egy void* típusú függvény, ezért kell elé egy cast, amivel int* típussá konvertáljuk a visszatérési értékét.

Mindezek után, egy ciklus segítségével várakoztatjuk a programot, míg egy másik program bele nem ír az osztott memóriába valamit. A mi esetünkben ez a valami 3 db egész szám. A ciklus a kódban így néz ki:

```
while(*shm == '\0')
{
        sleep(1);
}
```

Amíg a memóriaterület üres, addig a program várakozik. Ezt a várakozást a sleep() függvénnyel érjük el. Paraméternek másodpercet kell megadni.

Miután megtörtént a számok beolvasása a memória szegmensbe, ellenőrzésképpen egy for ciklussal kiiratjuk a képernyőre a szegmens tartalmát, majd ezután egy saját általunk írt függvénynek átadjuk ezt a 3 számot, ami meghatározza, hogy ezekből a számokból lehet-e háromszöget szerkeszteni. 1-et ad vissza a függvény, ha lehet és -1-et ad vissza, ha nem lehet. Az eredményt egy fájlba írja ki. Tehát, ha lehet háromszöget szerkeszteni, a fájlba kiírja a háromszög kerületét és területét. Ha nem lehet, akkor a fájlba a kerület és terület eredményének -1-et ír.

A program miután meghatározta a háromszög szerkeszthetőségét, shmdt() rendszerhívással lecsatlakoztatjuk a programot az osztott memóriáról és shmctl() hívással töröljük a szegmenset. Paraméternek az shmid-t, egy parancsot (IPC_RMID, ami azt mondja, hogy törli a memória azonosítot és a hozzátartozó memória szegmenset) és egy NULL.

```
shmctl(shmid,IPC_RMID,NULL);
```

2. A readFromFileToSHM.c elkészítése

Amíg a másik program várakozik, addig ez a program egy fájlból kiolvas 3 db egész számot, majd ezeket az osztott memóriában tárolja.

Első lépésekben létrehoztuk ugyanazokat a változókat, mint a másik program elején. Aztán az shmget()-el lokalizáljuk a létrehozott memória szegmenset, visszatérési értékét az shmid-ben tároljuk (kell az attach-nél). Gyakorlatban így néz ki:

```
shmflag = 00666;
shmid = shmget(key,size,shmflag);
```

Ugyanúgy leellenőrizzük shmid értékét (ugye ha kisebb, mint 0, az nem jó). Majd ezek után a programot hozzáadjuk ehhez a létrehozott szegmenshez. Pontosan így:

```
int* shm = (int*)shmat(shmid,NULL,shmflag);
```

Ugye eddig, ugyanúgy néz ki, mint az első program, annyi különbséggel, hogy ez nem hoz létre osztott memória szegmenst, hanem egy meglévőbe csatlakozik be. Most jön az a rész, mikor egy fájlból olvass be egy memória szegmensbe a következő módon:

```
FILE *fp;
fp = fopen(”adatok.txt”,”r”);
for(int i = 0; i < 3; i++)
{
        fscanf(fp,”%d ”,shm+i);
        printf(”%d. szam: %d”,shm[i]);
}
fclose(fp);
shmdt(shm);
```

Egyszerű fájlkezelés: megnyit, kiolvas, bezár. A for ciklusban lévő fscanf()-el egyenesen a memória szegmensbe ovassuk bele a 3 db egész számot. Miután vége a beolvasásnak, a fájlt bezárjuk és a shmdt() rendszerhívással lekapcsoljuk a programot az osztott memóriáról.

A futtatás eredménye:

szerkeszthető háromszög esetén

```
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/SHMcreate# gcc shmcre
ate.c -o shmcreate -lm
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/SHMcreate# ./shmcreat
e
Nincsen meg ilyen meretu shm szegmens, ezert most letrehozzuk
Az osztott memoria azonositoja: 491568

beolvasott 1. szam: 3
beolvasott 2. szam: 10
beolvasott 3. szam: 9

Igen, szerkesztheto haromszog ezekbol a szamokbol
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/SHMcreate#
```

```
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/readFromFileToSHM# gc
c readFromFileToSHM.c -o filetoshm
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/readFromFileToSHM# ./
filetoshm
1. szam: 3
2. szam: 10
3. szam: 9
fajlbol beolvasva, osztott memoriaba beleirva!
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/readFromFileToSHM#
```

eredmeny.txt ×

```
3 10 es 9 szamokbol lehet haromszoget keszíteni, így van kerulete es terulete
kerulet= 22
terulet = 13.266499
```

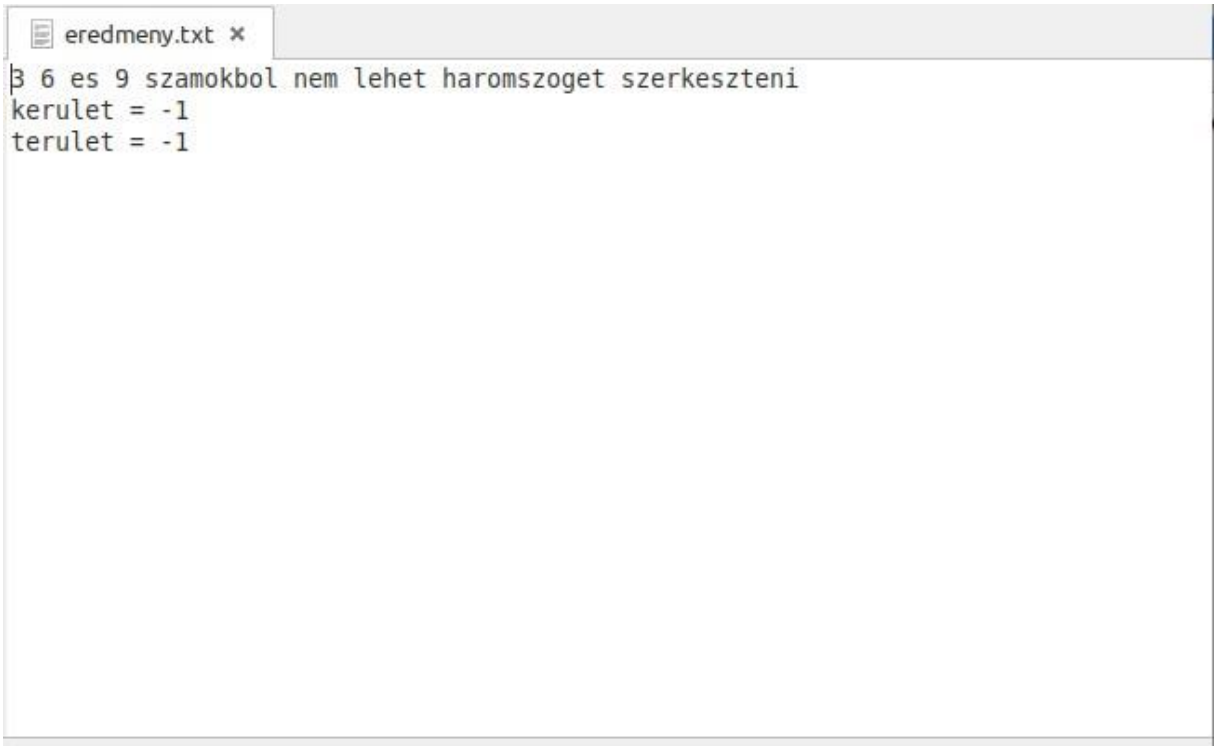
nem szerkeszthető háromszög esetén

```
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/SHMcreate# ./shmcreat
e
Nincsen meg ilyen meretu shm szegmens, ezert most letrehozzuk
Az osztott memoria azonositoja: 524300

beolvasott 1. szam: 3
beolvasott 2. szam: 6
beolvasott 3. szam: 9

Nem, nem szerkesztheto haromszog ezekbol a szamokbol
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/SHMcreate#
```

```
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/readFromFileToSHM# ./
filetoshm
1. szam: 3
2. szam: 6
3. szam: 9
fajlbol beolvasva, osztott memoriaba beleirva!
root@marcell-VirtualBox:/home/marcell/Dokumentumok/Projekt/readFromFileToSHM#
```


eredmeny.txt
3 6 es 9 szamokbol nem lehet haromszoget szerkeszteni
kerulet = -1
terulet = -1