

# JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: **Karczub Roland**

Neptunkód: **KJSPMW**

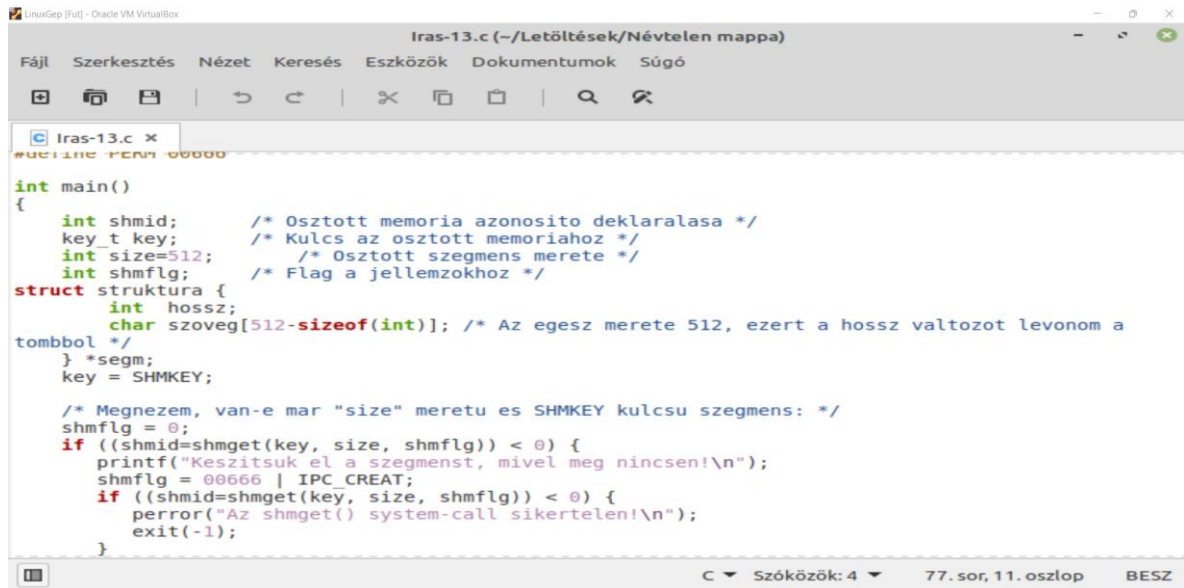
## **IPC FELADAT:**

### **A feladat leírása:**

Írjon C nyelvű programokat, ami létrehoz egy osztott memória szegmenst, az egyik program ír bele és vár pár másodpercet, bináris szemafor segítségével „védi” az írást, a másik program pedig kiolvas belőle.

## A feladat elkészítésének lépései:

Iras-13.c programkód:



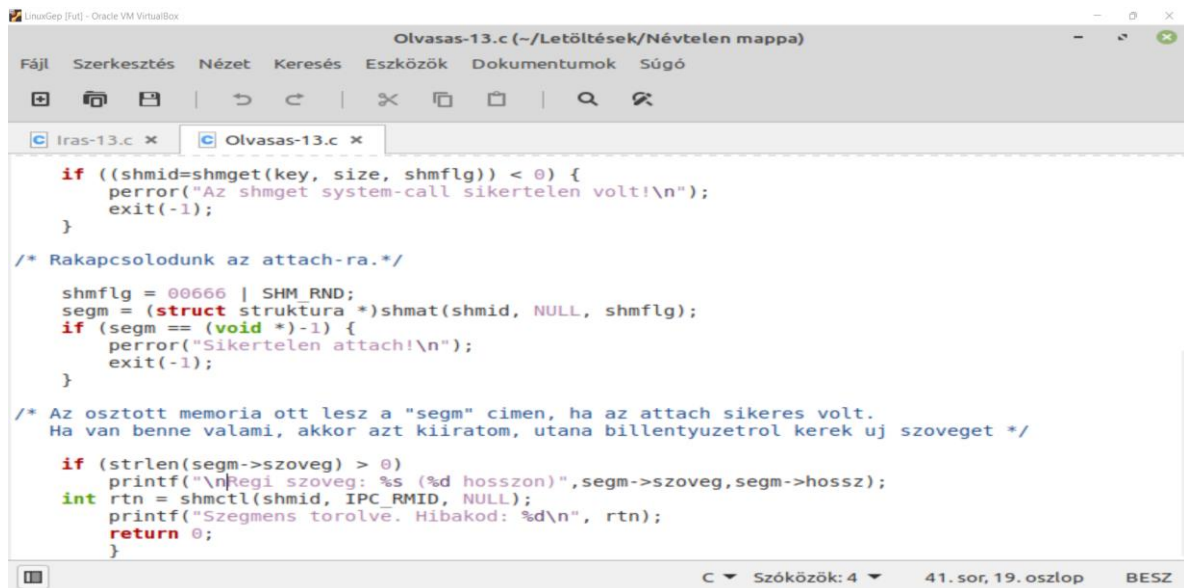
```
#define PERM 00000

int main()
{
    int shmid;          /* Osztott memoria azonosito deklaralasa */
    key_t key;          /* Kulcs az osztott memoriahoz */
    int size=512;       /* Osztott szegmens merete */
    int shmflg;         /* Flag a jellemzokhoz */
    struct struktura {
        int hossz;
        char szoveg[512-sizeof(int)]; /* Az egesz merete 512, ezert a hossz változtot levonom a
tombbol */
    } *segm;
    key = SHMKEY;

    /* Megnezem, van-e mar "size" meretu es SHMKEY kulcsu szegmens: */
    shmflg = 0;
    if ((shmid=shmget(key, size, shmflg)) < 0) {
        printf("Keszitsuk el a szegmenst, mivel meg nincs!\n");
        shmflg = 00666 | IPC_CREAT;
        if ((shmid=shmget(key, size, shmflg)) < 0) {
            perror("Az shmget() system-call sikertelen!\n");
            exit(-1);
        }
    }
}
```

C Szóközők: 4 77. sor, 11. oszlop BESZ

Olvasas-13.c programkód:



```
if ((shmid=shmget(key, size, shmflg)) < 0) {
    perror("Az shmget system-call sikertelen volt!\n");
    exit(-1);
}

/* Rakapcsolodunk az attach-ra.*/

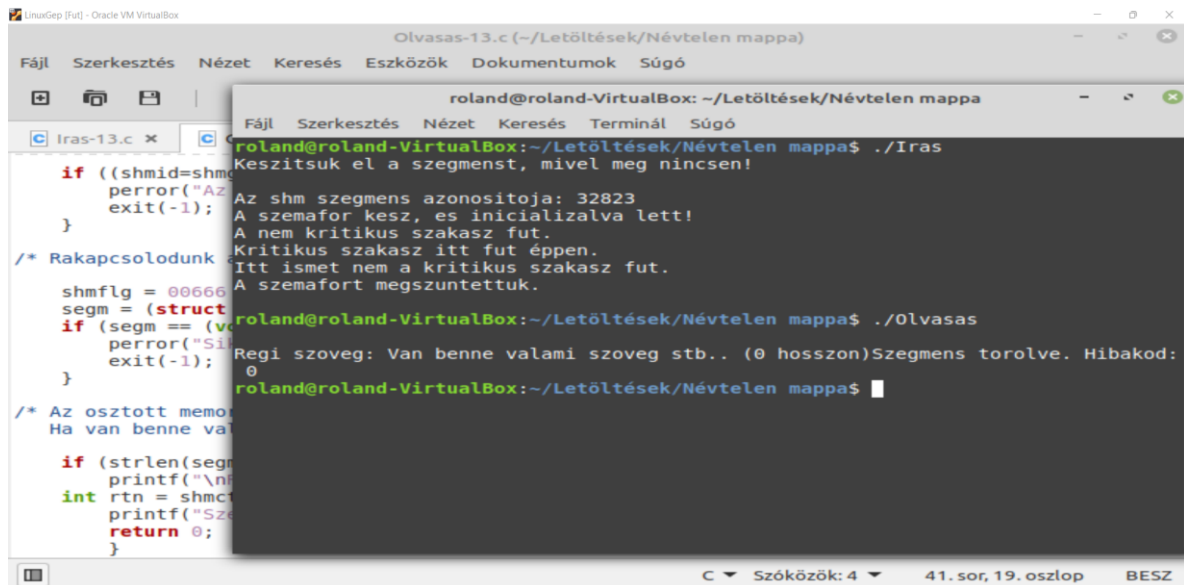
shmflg = 00666 | SHM_RND;
segm = (struct struktura *)shmat(shmid, NULL, shmflg);
if (segm == (void *)-1) {
    perror("Sikertelen attach!\n");
    exit(-1);
}

/* Az osztott memoria ott lesz a "segm" cimen, ha az attach sikeres volt.
Ha van benne valami, akkor azt kiiratom, utana billentyuzetrol kerek uj szoveget */

if (strlen(segm->szoveg) > 0)
    printf("\nRegi szoveg: %s (%d hossz)", segm->szoveg, segm->hossz);
int rtn = shmctl(shmid, IPC_RMID, NULL);
printf("Szegmens torolve. Hibakod: %d\n", rtn);
return 0;
}
```

C Szóközők: 4 41. sor, 19. oszlop BESZ

## A futtatás eredménye:



```
roland@roland-VirtualBox: ~/Letöltések/Névtelen mappa
roland@roland-VirtualBox:~/Letöltések/Névtelen mappa$ ./Iras
Készítsuk el a szegmenst, mivel meg nincsen!
Az shm szegmens azonosítója: 32823
A szemafor kész, és inicializálva lett!
A nem kritikus szakasz fut.
Kritikus szakasz itt fut éppen.
Itt ismét nem a kritikus szakasz fut.
A szemafort megszüntettük.
roland@roland-VirtualBox:~/Letöltések/Névtelen mappa$ ./Olvasas
Regi szoveg: Van benne valami szoveg stb.. (0 hosszon)Szegmens torolve. Hibakod:
0
roland@roland-VirtualBox:~/Letöltések/Névtelen mappa$
```

## ALGORITMUS FELADAT:

### A feladat leírása:

Adott egy számítógépes rendszer, melyben a szabad memória területek: 23KB, 64KB, 10KB, 80KB, 12KB, 50KB, és 40KB.

Foglalási igénye: 65KB, 21KB, 48KB, 13KB, 62KB.

Határozza meg a változó méretű partíció esetén a következő algoritmusok felhasználásával: next fit, worst fit a foglalási igényeknek megfelelő helyfoglalást – táblázatos formában!

### A feladat elkészítésének lépései:

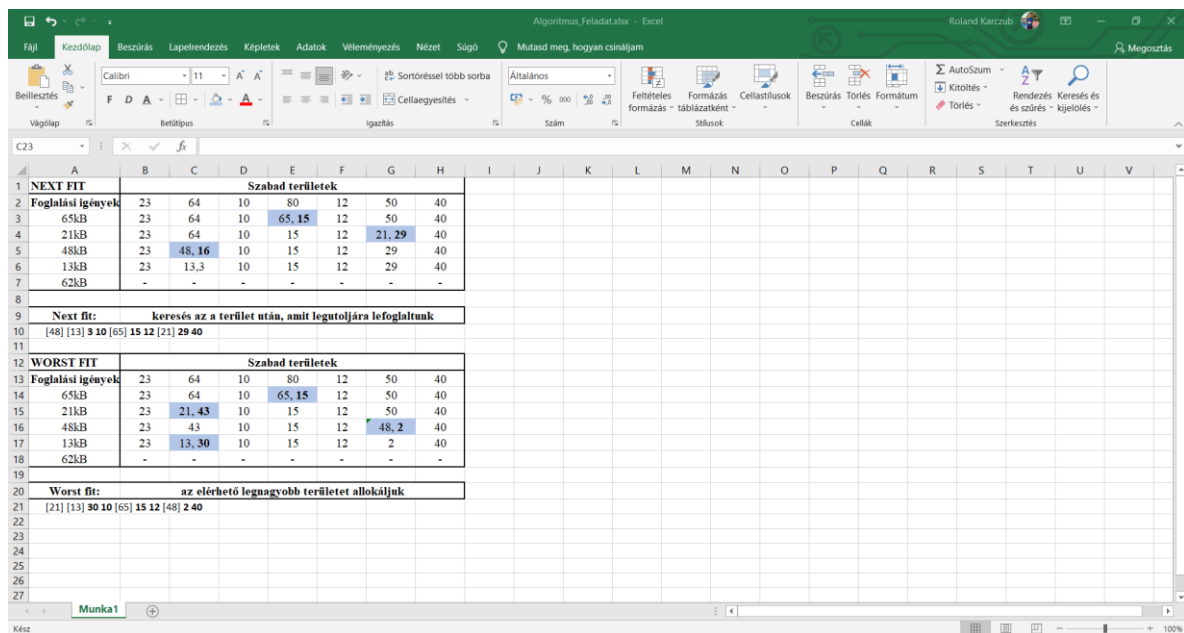
Első megfelelő (First fit): A rendelkezésre álló szabad területek közül a legelső elegendő méretűt foglaljuk le.

Következő megfelelő (Next Fit): A keresést nem a tábla elejétől kezdjük, hanem az után a terület után, amit legutoljára foglaltunk.

Legjobban megfelelő (Best fit): A legkisebbet foglaljuk le azon szabad területek közül, amelyek legalább akkorák, mint a lefoglalandó terület.

Legrosszabban illeszkedő (Worst fit): Az elérhető legnagyobb szabad területet allokáljuk. A maradék terület még talán elegendő lesz egy újabb foglalás számára.

## A futtatás eredménye:



	A	B	C	D	E	F	G	H
1	NEXT FIT	Szabad területek						
2	Foglalási igények	23	64	10	80	12	50	40
3	65kB	23	64	10	65, 15	12	50	40
4	21kB	23	64	10	15	12	21, 29	40
5	48kB	23	48, 16	10	15	12	29	40
6	13kB	23	13, 3	10	15	12	29	40
7	62kB	-	-	-	-	-	-	-
9	Next fit:	keresés az a terület után, amit legutoljára lefoglaltunk						
10		[48]	[13]	5	10	[65]	15	12
11								
12	WORST FIT	Szabad területek						
13	Foglalási igények	23	64	10	80	12	50	40
14	65kB	23	64	10	65, 15	12	50	40
15	21kB	23	21, 43	10	15	12	50	40
16	48kB	23	43	10	15	12	48, 2	40
17	13kB	23	13, 30	10	15	12	2	40
18	62kB	-	-	-	-	-	-	-
20	Worst fit:	az elérhető legnagyobb területet alkalmazzuk						
21		[21]	[13]	30	10	[65]	15	12
22								
23								
24								
25								
26								
27								

## Magyarázat:

Mindkét algoritmus esetén a bal első oszlopba kiírtam a Foglalási igényeket (kB), a táblázat első sorába egy fejlécezt adtam meg, ami a Szabad területeket foglalja össze. A „Szabad területek” alatt pedig a feladatban megadott szabad területekből kiszámoltam a foglalási igények alapján a hogy mennyi szabad terület marad, avagy eleget tud e tenni a foglalási igényeknek.

**Next fit algoritmus esetében:** A foglalási igényt megnézem melyik szabad területre fér be, majd beírom, és a fennmaradó területet vesszővel elválasztva leírom. Ezt követő sorban a foglalási igényt az előtte levő szerkesztett oszloptól indulva nézem, ami le lett foglalva, és úgy írom be egy olyan szabad területre, ahol megfelelő nagyságú hely van az igény számára. A 62kB foglalási igénynek nem volt akkora nagyságú szabad terület, mint amekkora kellett volna neki.

**Worst fit algoritmus esetén:** A foglalási igény abba a szabad területbe foglalom le, ahol a lehető legtöbb szabad terület fog maradni, a végén a 62kB-os igénynek nem volt megfelelő nagyságú szabad terület.

A kettő algoritmus közül a next fit alg. a jobb egyértelműen, mivel a worst fit az a legtöbb szabad területet hagyja a memóriában.

## Eredmények javítása:

Szemétygyűjtés (garbage collection) alkalmazása, vagyis a memória allokáció futás idejű átrendezése nem kielégíthető igények esetén. Hátránya, hogy erőforrás igényes.

Lapszervezés használata. Hátránya, hogy kell hozzá MMU (Memory Management Unit) támogatás.