JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: **Juhász Balázs**

Neptunkód: **ZUYISF**

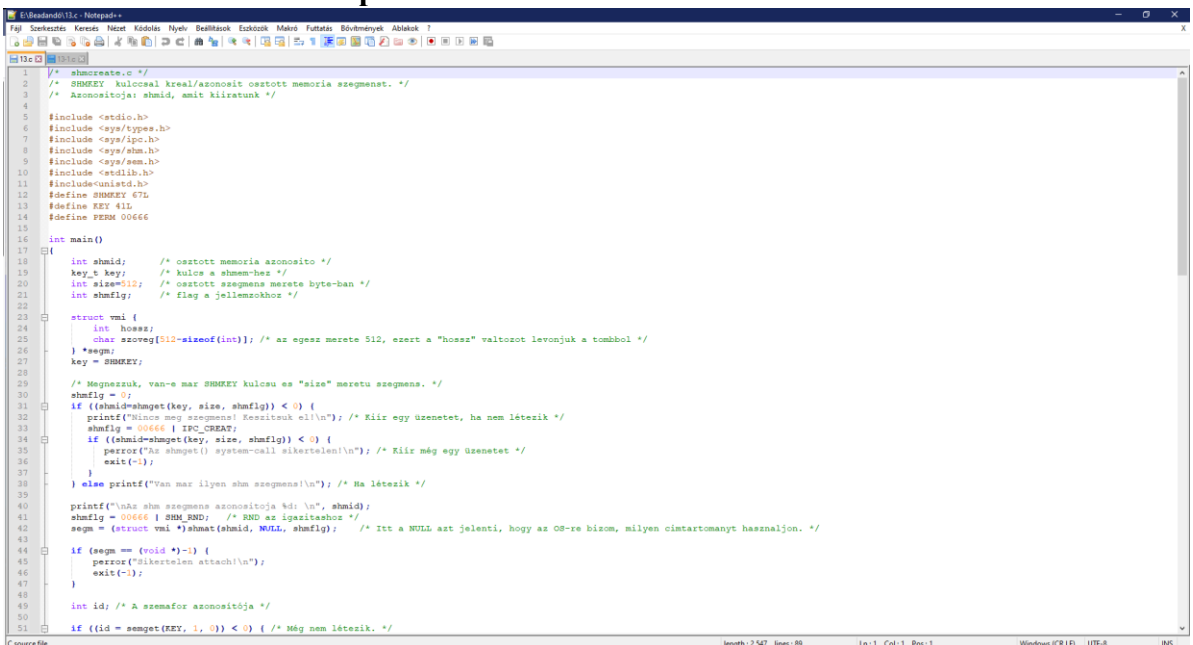
1. feladat:

IPC mechanizmus

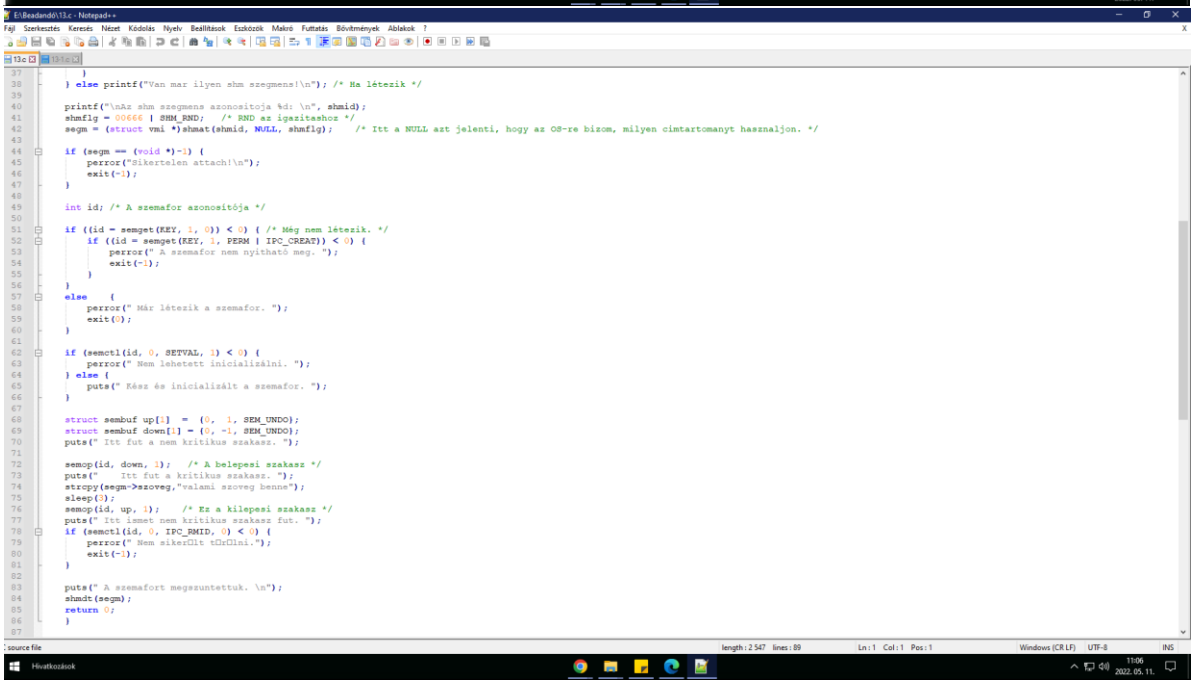
A feladat leírása:

13. feladat:
Írjon C nyelvű programokat, ami létrehoz egy osztott memória szegmenst az egyik program ír bele es vár pár másodpercet bináris szemafor segítségével "védi" az írást a másik program pedig kiolvas belőle.

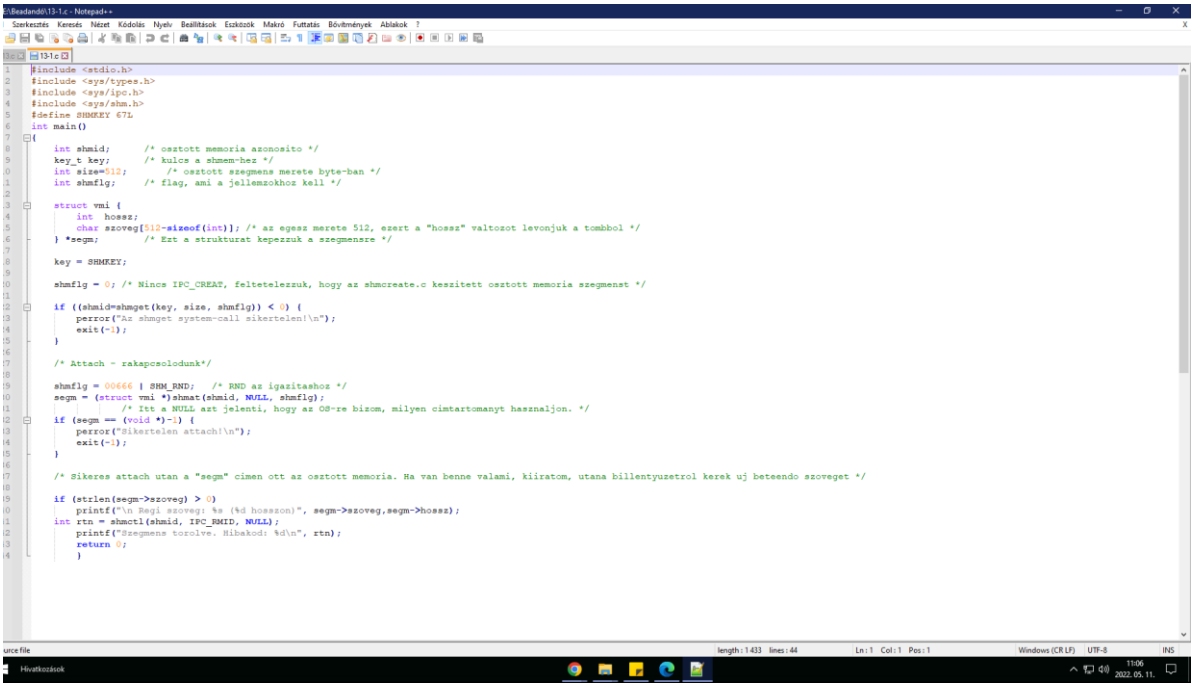
A feladat elkészítésének lépései:



```c
/*  shmcreate.c */
/*  SHMKEY  kulccsal kreal/azonosit osztott memoria szegmenst. */
/*  Azonositoja: shmid, amit kiiratunk */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <stdlib.h>
#include<unistd.h>
#define SHMKEY 67L
#define KEY 41L
#define PERM 00666

int main()
{
    int shmid;      /* osztott memoria azonosito */
    key_t key;      /* kulcs a shmem-hez */
    int size=512;   /* osztott szegmens merete byte-ban */
    int shmflg;     /* flag a jellemzokhoz */

    struct vmi {
        int  hossz;
        char szoveg[512-sizeof(int)]; /* az egesz merete 512, ezert a "hossz" valtozot levonjuk a tombbol */
    } *segm;
    key = SHMKEY;

    /* Megnezzuk, van-e mar SHMKEY kulcsu es "size" meretu szegmens. */
    shmflg = 0;
    if ((shmid=shmget(key, size, shmflg)) < 0) {
        printf("Nincs meg szegmens! Keszitsuk el!\n"); /* Kiir egy uzenetet, ha nem letezik */
        shmflg = 00666 | IPC_CREAT;
        if ((shmid=shmget(key, size, shmflg)) < 0) {
            perror("Az shmget() system-call sikertelen!\n"); /* Kiir még egy üzenetet */
            exit(-1);
        }
    } else printf("Van mar ilyen shm szegmens!\n"); /* Ha létezik */

    printf("\nAz shm szegmens azonositoja %d: \n", shmid);
    shmflg = 00666 | SHM_RND;   /* RND az igazitashoz */
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);    /* Itt a NULL azt jelenti, hogy az OS-re bizom, milyen cimtartomanyt hasznaljon. */

    if (segm == (void *)-1) {
        perror("Sikertelen attach!\n");
        exit(-1);
    }

    int id; /* A szemafor azonositója */

    if ((id = semget(KEY, 1, 0)) < 0) { /* Még nem létezik. */
```



```c
    } else printf("Van mar ilyen shm szegmens!\n"); /* Ha létezik */

    printf("\nAz shm szegmens azonositoja %d: \n", shmid);
    shmflg = 00666 | SHM_RND;   /* RND az igazitashoz */
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);    /* Itt a NULL azt jelenti, hogy az OS-re bizom, milyen cimtartomanyt hasznaljon. */

    if (segm == (void *)-1) {
        perror("Sikertelen attach!\n");
        exit(-1);
    }

    int id; /* A szemafor azonositója */

    if ((id = semget(KEY, 1, 0)) < 0) { /* Még nem létezik. */
        if ((id = semget(KEY, 1, PERM | IPC_CREAT)) < 0) {
            perror(" A szemafor nem nyithato meg. ");
            exit(-1);
        }
    }
    else    {
        perror(" Már létezik a szemafor. ");
        exit(0);
    }

    if (semctl(id, 0, SETVAL, 1) < 0) {
        perror(" Nem lehetett inicializálni. ");
    } else {
        puts(" Kész és inicializált a szemafor. ");
    }

    struct sembuf up[1]  =  {0,  1, SEM_UNDO};
    struct sembuf down[1] = {0, -1, SEM_UNDO};
    puts(" Itt fut a nem kritikus szakasz. ");

    semop(id, down, 1);   /* A belepesi szakasz */
    puts("    Itt fut a kritikus szakasz. ");
    strcpy(segm->szoveg,"valami szoveg benne");
    sleep(3);
    semop(id, up, 1);     /* Ez a kilepesi szakasz */
    puts(" Itt immet nem kritikus szakasz fut. ");
    if (semctl(id, 0, IPC_RMID, 0) < 0) {
        perror(" Nem sikerÜlt tÖrÖlni.");
        exit(-1);
    }

    puts(" A szemafort megszuntettuk. \n");
    shmdt(segm);
    return 0;
    }
```



```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SHMKEY 67L
int main()
{
    int shmid;      /* osztott memoria azonosito */
    key_t key;      /* kulcs a shmem-hez */
    int size=512;       /* osztott szegmens merete byte-ban */
    int shmflg;     /* flag, ami a jellemzokhoz kell */

    struct vmi {
        int  hossz;
        char szoveg[512-sizeof(int)]; /* az egesz merete 512, ezert a "hossz" valtozot levonjuk a tombbol */
    } *segm;        /* Ezt a strukturat kepezzuk a szegmensre */

    key = SHMKEY;

    shmflg = 0; /* Nincs IPC_CREAT, feltetelezzuk, hogy az shmcreate.c keszitett osztott memoria szegmenst */

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

    /* Attach - rakapcsolodunk*/

    shmflg = 00666 | SHM_RND;   /* RND az igazitashoz */
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);
                /* Itt a NULL azt jelenti, hogy az OS-re bizom, milyen cimtartomanyt hasznaljon. */
    if (segm == (void *)-1) {
        perror("Sikertelen attach!\n");
        exit(-1);
    }

    /* Sikeres attach utan a "segm" cimen ott az osztott memoria. Ha van benne valami, kiiratom, utana billentyuzetrol kerek uj beteendo szoveget */

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

A futtatás eredménye:

```
juhasz56@jerry:~/Desktop$ ./13.out
Nincs meg szegmens! Keszitsuk el!

Az shm szegmens azonositoja 622592:
 Kész és inicializált a szemafor.
 Itt fut a nem kritikus szakasz.
    Itt fut a kritikus szakasz.
 Itt ismet nem kritikus szakasz fut.
 A szemafort megszuntettuk.

juhasz56@jerry:~/Desktop$ ./13-1.out

 Regi szoveg: valami szoveg benne (0 hosszon)Szegmens torolve. Hibakod: 0
juhasz56@jerry:~/Desktop$
```

2. feladat:

OS algoritmusok

A feladat leírása:

5. feladat:

Adott négy processz (A, B, C, D) a rendszerbe, induláskor a p_cpu értéke A=0, B=0, C=0, D=0. A rendszerben a P_USER = 60. Az óraütés 1 indul, a befejezés 301-ig. Induláskor a p_usrpri A=60, B=60, C=65 és D=60. Induláskor a p_nice értéke A=0, B=0, C=5 és D=0.

a.) Határozza meg az ütemezést RR nélkül 301 óraütésig - táblázatba!

b.) Minden óraütem esetén határozza meg a processzek sorrendjét óraütés előtt/után.

c.) Igazolja a számítással a tanultak alapján.

A feladat elkészítésének lépései:

	A process		B process		C process		D process		Reschedule			
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	running before		running after	
Starting point	60	0	60	0	65	0	60	0	A		A	
1	60	1	60	0	65	0	60	0	A		A	
2	60	2	60	0	65	0	60	0	A		A	
3	60	3	60	0	65	0	60	0	A		A	
99	60	99	60	0	65	0	60	0	A		A	
100	85	50	60	0	65	0	60	0	A		B	
101	85	50	60	1	65	0	60	0	B		B	
199	85	50	60	99	65	0	60	0	B		B	
200	75	25	85	50	65	0	60	0	B		C	
201	75	25	85	50	65	1	60	0	C		C	
299	75	25	85	50	65	99	60	0	C		C	
300	63	12	75	25	95	50	60	0	C		D	
301	63	12	75	25	95	50	60	1	D		D	

A, B, D p_nice	0
C p_nice	5
p_user	60
p_uspri:	p_user+p_cpu/2+2*p_nice
p_cpu	p_cpu/2