

Operációs rendszerek Bsc

11. Gyak.

2022. 04. 29.

Készítette:

Flaskó Lilian Laura
Programtervező informatikus
GCNS8S

Miskolc, 2022.05

1.feladat:

Szabad: 30k, 35k, 15k, 25k, 75k, 45k Igény: 39k, 40k, 33k, 20k, 21k							
Foglalási igény		Memória terület, szabad terület					
		30	35	15	25	75	45
	39	30	35	15	25	75	45
	40	30	35	15	25	75	45
	33	30	35	15	25	75	45
	20	30	35	15	25	75	45
	21	30	35	15	25	75	45
First Fit		Memória terület, szabad terület					
Foglalási igény		30	35	15	25	75	45
	39	30	35	15	25	36, 39	45
	40	30	35	15	25	75	5, 40
	33	30	2, 33	15	25	75	45
	20	10, 20	35	15	25	75	45
	21	30	35	15	4, 21	75	45
Next Fit		Memória terület, szabad terület					
Foglalási igény		30	35	15	25	75	45
	39	30	35	15	25	39, 36	45
	40	30	35	15	25	75	40, 5
	33	30	33, 2	15	25	75	45
	20	30	35	15	20, 5	75	45
	21	30	35	15	25	21, 15	45
Best Fit		Memória terület, szabad terület					
Foglalási igény		30	35	15	25	75	45
	39	30	35	15	25	75	39, 6
	40	30	35	15	25	40, 35	45
	33	30	33, 2	15	25	75	45
	20	30	35	15	20, 5	75	45
	21	21, 9	35	15	25	75	45
Worst Fit		Memória terület, szabad terület					
Foglalási igény		30	35	15	25	75	45
	39	30	35	15	25	39, 36	45
	40	30	35	15	25	75	40, 5
	33	30	35	15	25	33, 3	40, 5
	20	30	20, 15	15	25	33, 3	40, 5
	21	21, 9	35	15	25	33, 3	40, 5

A legelső szabad helyre megy.

Helyet keres, lefoglal, majd ezt követően elindul és ha elfogy a szabad terület a kkor az elejétől kezd

Azt keresi ahol a legkevesebb és meghagyja a szabad területet

A legnagyobb területet lefoglalja és mindig csak azt

2.feladat

```
*semset.c (~\Asztal\GCNS850426)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x12

int semid,
    nsms,
    semnum,
    rtn;

int semflg;
struct sembuf sembuf, *sop;
union semun;

int cmd;

int main()
{
    int arg;

    nsms = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsms, semflg);
    if (semid < 0) { perror("semget() hibai\n"); exit(0); }
    else printf("\n Az azonosító: %d\n", semid);

    printf ("Kérem a semval értéket: ");
    semnum = 0;
    cmd = SETVAL;
    scanf("%d", &arg);
    rtn = semctl(semid, semnum, cmd, arg);
    printf("\nVisszatérési érték: %d\nSemval értéke: %d\n", rtn, arg);
}
```

```
semval.c (~/Asztal/GCNS850426)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semsetc x semvalc x

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define SEMKEY 0x12

int semid, nsems, rtn;
int semflg;
struct sembuf sembuf, *sop;
union semun arg;

union semun {
    int val;
    struct semid_ds *buf;
    unsigned short int *array;
};

int cmd;

int main()
{
    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("azonosító: %d\n", semid);
    cmd = GETVAL;
    rtn = semctl(semid, 0, cmd, NULL);
    printf("semval kiolvasott értéke: %d ", rtn);
    printf("\n");
    return 0;
}
```

```
semkill.c (~/Asztal/GCNS850426)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semsetc x semvalc x semkillc x

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x10

int semid, nsems, rtn;
int semflg;
struct sembuf sembuf, *sop;
union semun {
    int val;
    struct semid_ds *buf;
    unsigned short int *array;
};
int cmd;

int main()
{
    int arg;

    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("semid értéke: %d\n", semid);
    cmd = IPC_RMID;
    rtn = semctl(semid, 0, cmd, arg);
    printf("Kill visszatérés: %d\n", rtn);

    return 0;
}
```

```
semkill.c (~/Asztal/GCNS850426)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semsetc x semvalc x semkillc x

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x10

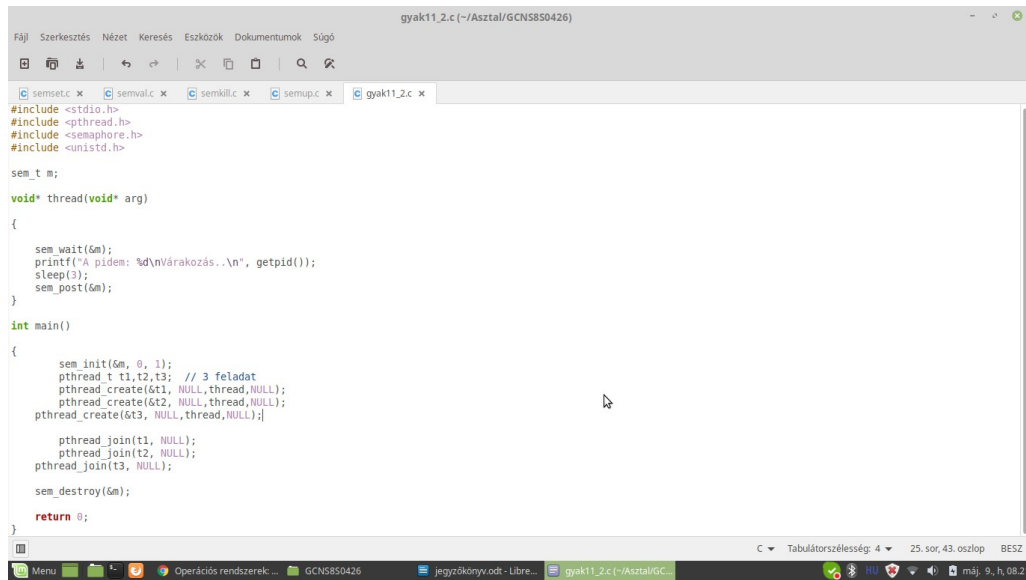
int semid, nsems, rtn;
int semflg;
struct sembuf sembuf, *sop;
union semun {
    int val;
    struct semid_ds *buf;
    unsigned short int *array;
};
int cmd;

int main()
{
    int arg;

    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("semid értéke: %d\n", semid);
    cmd = IPC_RMID;
    rtn = semctl(semid, 0, cmd, arg);
    printf("Kill visszatérés: %d\n", rtn);

    return 0;
}
```

2.a



The image shows a screenshot of a C code editor window titled "gyak11_2.c (~/Asztal/GCNS850426)". The editor contains the following C code:

```
Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semest.c x semval.c x semkill.c x semup.c x gyak11_2.c x

#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t m;

void* thread(void* arg)
{
    sem_wait(&m);
    printf("A pldem: %d\nVárakozás...\n", getpid());
    sleep(3);
    sem_post(&m);
}

int main()
{
    sem_init(&m, 0, 1);
    pthread_t t1,t2,t3; // 3 feladat
    pthread_create(&t1, NULL,thread,NULL);
    pthread_create(&t2, NULL,thread,NULL);
    pthread_create(&t3, NULL,thread,NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);

    sem_destroy(&m);

    return 0;
}
```

The code implements a semaphore-based thread synchronization. It defines a semaphore `m` with an initial value of 1. Three threads are created, each calling `thread`. Each thread acquires the semaphore (`sem_wait`), prints its PID and a message, sleeps for 3 seconds, and then releases the semaphore (`sem_post`). The `main` function initializes the semaphore, creates the three threads, joins them, destroys the semaphore, and returns 0.

Tabulátorszélesség: 4 25. sor, 43. oszlop BESZ

Menu 4: Operációs rendszerek GCNS850426 jegyzőkönyv.odt - Libre... gyak11_2.c (~/Asztal/GC maj. 9. h. 08.21

