JEGYZŐKÖNYV

Operációs rendszerek BSc

11. Gyakorlat

Készítette: **Szeszák Ádám**

Neptunkód: **AZCTJJ**

1. Feladat

First fit

First Fit	Szabad területek közül az első szabad méretű					
	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	36	40,5
33	30	33,2	15	25	36	5
20	20,1	2	15	25	36	5
21	10	2	15	21,4	36	5

Next fit

Next fit	Keresés azután a terület után, amit utoljára foglaltunk					
	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	36	40,5
33	30	33,2	15	25	36	5
20	30	2	15	20,5	36	5
21	30	2	15	5	39,21,15	5

Best fit

Best fit	Az elérhető legkisebb szabad területet allokáljuk					
	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	6
33	30	33,2	15	25	35	6
20	30	2	15	20,5	35	6
21	21,9	2	15	5	35	6

Worst fit

Worst fit	Az elérhető legnagyobb szabad területet allokáljuk					
	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	36	40,5
33	30	35	15	25	39,33,3	5
20	30	20,15	15	25	3	5
21	21,9	15	15	25	3	5

2. Feladat

semset.c

Létrehoz n db szemafort, majd inicializája 0 értékkel.
A semget hívással tudunk létrehozni szemafort, ahol szükség van egy kulcsra illetve jogosultságokra. A semctl hívással tudjuk kontrollálni, ebben a SETALL kapcsoló beállítja az arg union semun típusú array nevű tömbjében szereplő értékekre. A semun union tartalmazza a számot, amit az adott szemafor tartalmaz, descriptort, tömböt és egy információt tartalmazó buffert.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

union semun {
    int val;                    /* Value for SETVAL */
    struct semid_ds *buf;       /* Buffer for IPC_STAT, IPC_SE
    unsigned short *array;      /* Array for GETALL, SETALL */
    struct seminfo *__buf;      /* Buffer for IPC_INFO (Linux-
};

void main() {
    union semun arg;

    int n = 10;
    int semID = semget(KEY, n, IPC_CREAT | 0666);

    if (semID == -1) {
        perror("Nem sikerult szemaforokat letrehozni");
        exit(-1);
    }

    arg.array = (short *)calloc(n, sizeof(int));


    if (semctl(semID, 0, SETALL, arg)) {
        perror("Nem sikerult beallitani az erteket\n");
        exit(-1);
    }
}
```

semval.c

Lekérdezzük a tartalmat. Mivel nem hozunk létre új szemafort, az argumentumnak csak a key kell.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

union semun {
    int val;                    /* Value for SETVAL */
    struct semid_ds *buf;       /* Buffer for IPC_STAT, IPC_SE
    unsigned short *array;      /* Array for GETALL, SETALL */
    struct seminfo *__buf;      /* Buffer for IPC_INFO (Linux-
};

void main() {

    int semID = semget(KEY, 0, 0);
    int n = 10;
    if (semID == -1) {
        perror("Nem sikerult szemaforokat lekerdezni\n");
        exit(-1);
    }

    union semun arg;

    printf("Szemaforok tartalma: \n");
    arg.array = (short *)calloc(n, sizeof(int));

    semctl(semID, 0, GETALL, arg);

    for (int i = 0; i < n; i++)
        printf("%d ", arg.array[i]);
}
```

semkill.c

Szemaforok törlése a semctl hívás IPC_RMID kapcsoló használatával.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

void main() {
    int n = 5;
    int semID = semget(KEY, 0, 0);
    if (semID == -1) {
        perror("Nem sikerult szemaforokat lekerdezni\n");
        exit(-1);
    }

    for (int i = 0; i < n; i++)
        semctl(semID, i, IPC_RMID);
}
```

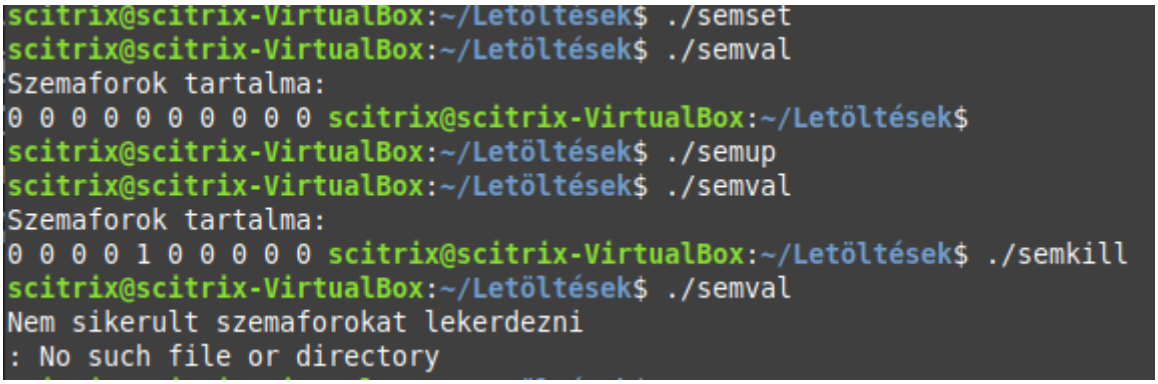
semup.c

A semop függvény feladata egy szemaforköteg egy elemének növelése és csökkentése. Az előző programok létrehoztak egy 10 elemű szemafort. Ez a program az 5. szemafort fogja növelni.

A sembuf struktúrában van egy sem_num változó, ami a szemaforok számát kapja. A sem_op növelés(1) vagy csökkentés(-1) lehet. A sem_flg a hozzáférési jogokat tartalmazza.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

void main() {
    int semID = semget(KEY, 0, 0);
    if (semID == -1) {
        perror("Nem sikerult szemaforokat lekerdezni\n");
        exit(-1);
    }

    struct sembuf buffer;

    buffer.sem_num = 5;
    buffer.sem_op = 1;
    buffer.sem_flg = 0666;

    if (semop(semID, &buffer, 1)) {
        perror("Sikertelen\n");
        exit(-1);
    }
}
```

A programok futási eredménye:



A semset programmal létrehozzuk a szemaforokat, a semval program kiírja a szemaforunk tartalmát, a semup program futtatása után jól látható, hogy az 5. értéket növeltük, majd a semkill futtatása után már nem tudjuk lekérdezni a szemaforokat, ugyanis töröltük.