Operációs rendszerek BSc

12. Gyak. 2022. 04. 25.

Készítette:

Szkárosi Szilárd Bsc Mérnökinformatikus DLWGQZ

Miskolc, 2022

- **1. feladat** Adott egy rendszer (foglalási stratégiák), melyben a következő
 - Szabad területek: 30k, 35k, 15k, 25k, 75k, 45k és
 - Foglalási igények: 39k, 40k, 33k, 20k, 21k állnak rendelkezésre.

A rendszerben a memória 4 kbyte-os blokkokban kerül nyilvántartásra, ennél kisebb méretű töredék igény esetén a teljes blokk lefoglalásra kerül. Határozza meg változó méretű partíció esetén a következő algoritmusok felhasználásával: first fit, next fit, best fit, worst fit a foglalási igényeknek megfelelő helyfoglalást – táblázatos formában (az ea. bemutatott mintafeladat alapján)! Hasonlítsa össze, hogy a teljes szabad memóriaterület hány százaléka vész el átlagosan az egyes algoritmusok esetén! A kapott eredményeket ábrázolja oszlop diagrammal! Magyarázza a kapott eredményeket és hogyan lehet az eredményeket javítani!

Szabad területek:	30k, 35k, 15k, 25k, 75k, 45k	first fit						
	39k, 40k, 33k, 20k, 21k		Memória terület - szabad terület					
		Foglalási igény	30	35	15	25	75	4
	first fit, next fit, best fit, worst fit	39					36 (75 - 39)	
		40						5 (45 - 40
		33		2 (35 - 33)				
		20				5 (25 - 20)		
		21	9 (30 - 31)					
		next fit						
			Memória terület - szabad terület					
		Foglalási igény	30		15			4
		39					36 (75 - 39)	
		40						5 (45 - 40
		33		2 (35 - 33)				
		20				5 (25 - 20)		
		21					15 (36 - 21)	
		best fit						
			Memória terület - szabad terület					
		Foglalási igény	30	35	15	25	75	4
		39						6 (45 - 39
		40					35 (75 - 40)	
		33		2 (35 - 33)				
		20				5 (25 - 20)		
		21	9 (30 - 31)					
		worst fit						
			Memória terület - szabad terület					
		Foglalási igény	30	35	15	25	75	4
		39					36 (75 - 39)	
		40						5 (45 - 40
		33					3 (36 - 33)	
		20		15 (35 - 20)				
			9 (30 - 31)					

2. feladat – A feladat megoldásához először tanulmányozza Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (6.4)., azaz Írjon C nyelvű programokat, ahol

• kreál/azonosít szemafor készletet, benne N szemafor-t. A kezdő értéket 0-ra állítja – semset.c,

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L
union semun {
                                 /* Value for SETVAL */
    int val;
    struct semid_ds *buf;  /* Buffer for IPC_STAT, IPC_SET */
unsigned short *array;  /* Array for GETALL, SETALL */
struct seminfo *__buf;  /* Buffer for IPC_INFO (Linux-specific) */
};
void main() {
     union semun arg;
     int n = 5;
    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);
     }
```

• kérdezze le és írja ki a pillanatnyi szemafor értéket – semval.c

```
c semval.c ×
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L
union semun {
                             /* Value for SETVAL */
    int val;
                            /* Buffer for IPC_STAT, IPC_SET */
/* Array for GETALL, SETALL */
    struct semid ds *buf;
    unsigned short *array;
    struct seminfo *_buf; /* Buffer for IPC_INFO (Linux-specific) */
};
void main() {
    int semID = semget(KEY, 0, 0);
    int n = 5;
    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 \n", arg.array[i]);
    }
}
szkarosi1@szkarosi1-VirtualBox:~/OS_GYAK/OS_12_Gyak$ ./semval.out
Szemaforok tartalma:
0 0 0 0 0 szkarosi1@szkarosi1-VirtualBox:~/OS_GYAK/OS_12_Gyak$
```

• szüntesse meg a példácskák szemafor készletét – semkill.c

```
c semkill.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);
```

sembuf.sem_op=1 értékkel inkrementálja a szemafort – semup.c

```
c semup.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;
                            //a 4.ik szemafort
    buffer.sem num = 4;
    buffer.sem op = 1;
                            //inkrementaljuk a szemaforokat
    buffer.sem_flg = 0666; //jogok
    if (semop(semID, &buffer, 1)) {
        perror("Sikertelen\n");
        exit(-1);
    }
}
```

2a. feladat – a. Írjon egy C nyelvű programot, melyben

- egyik processz létrehozza a szemafort (egyetlen elemi szemafort; inicializálja 1-re, vagy x-re, ha még nem létezik),
- másik processz használja a szemafort, belépési szakasz (down), a kritikus szakaszban alszik 2-3 sec-et, m pid-et kiír, kilépési szakasz (up), ezt ismételve 2x-3x (és a hallgató egyszerre indítson el 2-3 ilyen processzt),

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#define KEY 77777L
void up(int);
void down(int);
void main()
{
    int semID = semget(KEY, 0, 0);
    if (semID == -1)
        perror("Nem sikerult megnyitni\n");
        exit(-1);
    }
    //belepesi szakasz
    printf("Kritikus szakasz\n");
    down(semID);
    sleep(3);
    printf("pid : %d\n", getpid());
    printf("%d \n", semctl(semID, 0, GETVAL));
    up(semID);
    printf("kritikus szakasz vege\n");
}
```

```
void up(int semId) {
    struct sembuf buffer;
    buffer.sem_num = 0;
    buffer.sem_op = 1;
    buffer.sem_flg = 0;

    semop(semId, &buffer, 1);
}

void down(int semId) {
    struct sembuf buffer;
    buffer.sem_num = 0;
    buffer.sem_op = -1;
    buffer.sem_flg = 0;

    semop(semId, &buffer, 1);
}
```

• harmadik processzben, ha létezik a szemafor, akkor megszünteti".

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <svs/sem.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#define KEY 77777L
void main() {
    int semID = semget(KEY, 0, 0);
    if (semID == -1)
        perror("Nem sikerult megnyitni\n");
        exit(-1);
    if (semctl(semID, 0, IPC RMID) == -1)
        perror("Nem sikerult torolni\n");
        exit(-1);
    }
    printf("Torolve\n");
}
```

A futtatás eredményei:

```
szkarosil@szkarosil-VirtualBox:~$ cd OS_GYAK/OS_11_Gyak
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_11_Gyak$ gcc gyak11_2.c -o gyak11_2.out
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_11_Gyak$ ./gyak11_2.out
Szam: 2
Szem: 2
A szemafor erteke (1): 2
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_11_Gyak$ gcc gyak11_2._masik.c -o gyak11_2_masik.out
gcc: error: gyak11_2. masik.c: Nincs ilyen fájl vagy könyvtár
gcc: fatal error: no input files
compilation terminated.
szkarosil@szkarosil-VirtualBox:-/OS_GYAK/OS_11_Gyak$ gcc gyak11_2_masik.c -o gyak11_2_masik.out
szkarosil@szkarosil-VirtualBox:-/OS_GYAK/OS_11_Gyak$ ./gyak11_2_masik.out
kritikus szakasz vege
szkarosil@szkarosil-VirtualBox:-/OS_GYAK/OS_11_Gyak$ gcc gyak11_2_harmadik.c -o gyak11_2_harmadik.out
szkarosil@szkarosi1-VirtualBox:-/OS_GYAK/OS_11_Gyak$ ./gyak11_2_harmadik.out
Torolve szkarosi1-VirtualBox:~/OS_GYAK/OS_11_Gyak$ [
```