Operációs rendszerek BSc

12. Gyak.

2022. 04. 25.

Készítette:
Baráth Kristóf Bsc
Mérnökinformatikus
DQPDLY

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						
Foglalási igények:	39k, 40k, 33k, 20k, 21k		Memória terület - szabad terület					
		Foglalási igény	30	35	15	25	75	45
	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	35	15	25	75	45
		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	45
		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	45
		39					36 (75 - 39)	
		40						5 (45 - 40)
		33					3 (36 - 33)	
		20		15 (35 - 20)				
		21	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 0ra állítja – semset.c,

```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;
    struct semid_ds *buf;
    unsigned short *array;
    struct seminfo *__buf;
};

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]);
    }

}
```

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

```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);

}
```

- szüntesse meg a példácskák szemafor készletét – semkill.c

```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 = 4;
    buffer.sem_op = 1;
    buffer.sem_flg = 0666;

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

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

```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;
    struct semid_ds *buf;
    unsigned short *array;
    struct seminfo *__buf;
};

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);
    }

}
```

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),

```c
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”.

```c
#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 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");

}
```