Operációs rendszerek BSc

11. Gyak.

2022. 04. 25.

Készítette:
Nyíri Levente Bsc
Mérnökinformatikus
F023QC

Miskolc, 2022

1. feladat

Szabad területek:	30k, 35k, 15k, 25k, 75k, 45k
Foglalási igények:	39k, 40k, 33k, 20k, 21k
	first fit, next fit, best fit, worst fit

first 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	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

- semset.c,

```c
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);
    }

}
```

- semval.c

```c
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]);
    }

}
```

semkill.c

```c
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

```c
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);
    }
}
```

2a. feladat

```c
void up(int);
void down(int);

void main()
{
    int semID = semget(KEY, 0, 0);

    if (semID == -1)
    {
        perror("Nem sikerult megnyitni\n");
        exit(-1);
    }

    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);
}
```

```c
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");
}
```