

# **Operációs rendszerek BSc**

11.Gyak

2022.04.25.

**Készítette:**

Urbán Olivér BSc

Szak:

Mérnökinformatikus

Neptunkód: HEPMIU

**2022.04.25.**

## Feladatok

„1. 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 százalékavész el átlagosan az egyes algoritmusok esetén! A kapott eredményeket ábrázolja oszlop diagrammal!

Magyarázza a kapott eredményeket és hogyan lehet az eredményeket javítani!

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1 Szabad területek:	30k, 35k, 15k, 25k, 75k, 45k																										
2 Foglalási igények:	39k, 40k, 33k, 20k, 21k																										
3																											
4																											
5																											
6																											
7 first fit																											
8																											
9 Foglalási igény	39	30	35	15	25	75	45																				
10																											
11																											
12																											
13																											
14																											
15																											
16 next fit																											
17																											
18 Foglalási igény	39	30	35	15	25	75	45																				
19																											
20																											
21																											
22																											
23																											
24																											
25 best fit																											
26																											
27 Foglalási igény	39	30	35	15	25	75	45																				
28																											
29																											
30																											
31																											
32																											
33																											
34 worst fit																											
35																											
36 Foglalási igény	39	30	35	15	25	75	45																				
37																											
38																											

2. Gyakorló 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,

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

❓ szüntesse meg a példácskák szemafor készletét – semkill.c

❓ sembuf.sem\_op=1 értékkel inkrementálja a szemafort – semup.c

A futtatás eredményét is tartalmazza a jegyzőkönyv.

```
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/ipc.h>
4  #include <sys/sem.h>
5  #define KEY 126K
6
7  int semid, nsems, semnum, rtn;
8
9  int semflg;
10 struct sembuf sembuf, *sop;
11 union semun arg;
12 int cmd;
13
14 main()
15 {
16
17
18     nsems = 1;
19     semflg = 00666 | IPC_CREAT;
20     semid = semget (SEMKEY, nsems, semflg);
21     if (semid < 0 ) {perror(" semget hiba"); exit(0);}
22     else printf("\n semid: %d ", semid);
23     printf ("\n kerem a semval erteket ");
24
25     semnum = 0;
26
27     cmd = SETVAL;
28     scanf("%d", &arg.val);
29     rtn = semctl(semid, semnum, cmd, arg);
30
31     printf("\n set   rtn: %d ,semval: %d ", rtn, arg.val);
32     printf("\n");
33
34 }
35
```

```

1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/ipc.h>
4  #include <sys/sem.h>
5  #define KEY 126K
6
7
8  int semid,nsems,rtn;
9  int semflg;
10 struct sembuf sembuf, *sop;
11 union semun arg;
12 int cmd;
13
14 main()
15 {
16
17
18     nsems = 1;
19     semflg = 00666 | IPC_CREAT;
20     semid = semget (SEMKEY, nsems, semflg);
21     if (semid < 0 ) {perror(" semget hiba"); exit(0);}
22     else printf("\n semid: %d ",semid);
23     printf ("\n");
24
25     cmd = GETVAL;
26     rtn = semctl(semid,0, cmd, NULL);
27
28     printf("\n semval: %d ",rtn);
29     printf("\n");
30
31 }
32

```

```

1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/ipc.h>
4  #include <sys/sem.h>
5  #define KEY 126K
6
7
8      int semid,nsems,rtn;
9      int semflg;
10     struct sembuf sembuf, *sop;
11     union semun arg;
12     int cmd;
13
14     main()
15     {
16
17
18         nsems = 1;
19         semflg = 00666 | IPC_CREAT;
20         semid = semget (SEMKEY, nsems, semflg);
21         if (semid < 0 ) {perror(" semget hiba"); exit(0);}
22         else printf("\n semid: %d ",semid);
23         printf ("\n");
24
25         cmd = IPC_RMID;
26         rtn = semctl(semid,0, cmd, arg);
27
28         printf("\n kill rtn: %d ",rtn);
29         printf("\n");
30
31     }
32

```

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L

int semid,nsems,rtn;
unsigned nsops;
int semflg;
struct sembuf sembuf, *sop;

main()
{

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

    nsops = 1;
    sembuf.sem_num = 0;
    sembuf.sem_op = 1;
    sembuf.sem_flg = 0666;
    sop = &sembuf;
    rtn = semop(semid, sop, nsops);
    printf("\n up rtn: %d ",rtn);
    printf("\n");
}

```

2a. Írjon egy C nyelvű programot, melyben

- ❑ egyik processz létrehozza a szemafor (egyetlen elemi szemafor; inicializálja 1-re, vagy x-re, ha még nem létezik),
- ❑ másik processz használja a szemafor, 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),
- ❑ harmadik processzben, ha létezik a szemafor, akkor megszünteti”.

Mentés: gyak11\_2.c

```

1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/ipc.h>
4  #include <sys/sem.h>
5  #include <stdlib.h>
6  #include <errno.h>
7  #include <unistd.h>
8
9  #define KEY 126K
10
11  union semun {
12      int val;
13      struct semid_ds *buf;
14      unsigned short *array;
15      struct seminfo *__buf;
16  };
17
18  void main() {
19      union semun arg;
20
21      int semID = semget(KEY, 0, 0);
22      if (errno == ENOENT)
23      {
24          semID = semget(KEY, 1, IPC_CREAT | 0666);
25          printf("Szam: ");
26          scanf("%d", &(arg.val));
27      }
28      else
29      {
30          arg.val = 1;
31      }
32
33      semctl(semID, 0, SETVAL, arg);
34
35      printf("A szemafor értéke (1) : %d\n", semctl(semID, 0, GETVAL));
36
37  }
38

```

```

#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 126K

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");
}

```

```

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>

#define KEY 126K

void up(int);
void down(int);

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);
}

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

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

```



