Operációs rendszerek BSc

10.gyak.

2021. 04. 26.

Készítette:
Ábrahám Péter István

Gazdaságinformatikus
Neptunkód: **A4XIOV**

Miskolc, 2021

„**1.** Az előadáson bemutatott mintaprogram alapján készítse el a következő feladatot.

Adott egy rendszerbe az alábbi erőforrások: R (R1: 10; R2: 5; R3: 7)

A rendszerbe 5 processz van: P0, P1, P2, P3, P4

Kérdés: Kielégíthető-e P4 (3,3,0) ill. P0 (0,2,0) kérése úgy, hogy biztonságos legyen, holtpontmentesség szempontjából a rendszer - a következő *kiinduló állapot* alapján. Igazolja a processzek végrehajtásának sorrendjét – számolással.”

Az egyes részfeladatok lentebb a munkalapokban vannak.

R1 : 10 R2: 5 R3 : 7

	Max. igény			Foglal			Kielégítetlen igények			max igény - foglal
	R1	R2	R3	R1	R2	R3	R1	R2	R3	
p0	7	5	3	0	1	0	7	4	3	
p1	3	2	2	2	0	0	1	2	2	
p2	9	0	2	3	0	2	6	0	0	
p3	2	2	2	2	1	1	0	1	1	
p4	4	3	3	0	0	2	4	3	1	
			Foglaltak:	7	2	5	Készlet igény			
			Összesen:	10	5	7	-4	-1	-1	
			Szabad ef. Szám	3	3	2	2	1	0	Tehát a P1 igénye kielégíthető
							-3	3	2	
							3	2	1	
							-1	0	1	
							3	3	2	

P1 lefutása után

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	1	0	7	4	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	2	1	1	0	1	1
p4	4	3	3	0	0	2	4	3	1
			Foglaltak:	7	2	5	Készlet igény		
			Összesen:	10	5	7	-2	-1	-1
			Szabad ef. Szám	5	3	2	4	1	0 LEFUTOTT
							-1	3	2
							5	2	1 P3 igénye kielégíthető
							1	0	1
							5	3	2

P3 lefutása után

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	1	0	7	4	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	0	0	2	4	3	1

P0 lefutása után

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	**0**	**1**	**0**	7	4	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	0	0	2	4	3	1

Foglaltak:	7	2	5
Összesen:	10	5	7
Szabad ef. Szám	7	5	3

Készlet igény			
0	1	0	LEFUTOTT
6	3	1	LEFUTOTT
1	5	3	P2 igénye kielégíthető
7	4	2	LEFUTOTT
3	2	2	

P2 lefutása után

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	**0**	**1**	**0**	7	4	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	**3**	**0**	**2**	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	0	0	2	4	3	1

Foglaltak:	7	2	5
Összesen:	10	5	7
Szabad ef. Szám	10	5	5

Készlet igény			
0	1	0	LEFUTOTT
6	3	1	LEFUTOTT
1	5	3	LEFUTOTT
7	4	2	LEFUTOTT
3	2	2	P4 igénye kielégíthető

P4 lefutása után

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	**0**	**1**	**0**	7	4	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	**3**	**0**	**2**	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	**0**	**0**	**2**	4	3	1

Foglaltak:	7	2	5
Összesen:	10	5	7
Szabad ef. Szám	**10**	**5**	**7**

Készlet igény			
0	1	0	LEFUTOTT
6	3	1	LEFUTOTT
1	5	3	LEFUTOTT
7	4	2	LEFUTOTT
3	2	2	LEFUTOTT

A rendszer biztonságos állapotban van.

Processzek végrehajtásának sorrendje: P1 - P3 - P0 - P2 - P4

Kielégíthető-e P4 (3,3,0)

	R1 : 10	R2: 5	R3 : 7								
	Max. igény				**Foglal**			**Kielégítetlen igények**			
	R1	R2	R3		R1	R2	R3	R1	R2	R3	
p0	7	5	3		0	1	0	7	4	3	
p1	3	2	2		2	0	0	1	2	2	
p2	9	0	2		3	0	2	6	0	0	
p3	2	2	2		2	1	1	0	1	1	
p4	4	3	3		0	0	2	4	3	1	
				Foglaltak:	7	2	5	**Készlet igény**			
				Összesen:	10	5	7	-4	-1	-1	
				Szabad ef. Szám:	3	3	2	2	1	0	
								-3	3	2	
								3	2	1	
								-1	0	1	

P4 kérelme után

	R1 : 10	R2: 5	R3 : 7								
	Max. igény				**Foglal**			**Kielégítetlen igények**			
	R1	R2	R3		R1	R2	R3	R1	R2	R3	
p0	7	5	3		0	1	0	7	4	3	
p1	3	2	2		2	0	0	1	2	2	
p2	9	0	2		3	0	2	6	0	0	
p3	2	2	2		2	1	1	0	1	1	
p4	4	3	3		3	3	2	1	0	1	
				Foglaltak:	10	5	5	**Készlet igény**			
				Összesen:	10	5	7	-7	-4	-1	
				Szabad ef. Szám:	0	0	2	-1	-2	0	
								-6	0	2	
								0	-1	1	
								-1	0	1	

Egyik processz sem tud lefutni.

P0 (0,2,0)

R1 : 10 R2: 5 R3 : 7

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	1	0	7	4	3
p1	3	2	2	2	0	0	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	2	1	1	0	1	1
p4	4	3	3	0	0	2	4	3	1

				Készlet igény		
Foglaltak:	7	2	5			
Összesen:	10	5	7	-4	-1	-1
Szabad ef. Szám:	3	3	2	2	1	0
				-3	3	2
				3	2	1
				-1	0	1

P0 kérelme után

R1 : 10 R2: 5 R3 : 7

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	3	0	7	2	3
p1	3	2	2	2	0	0	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	2	1	1	0	1	1
p4	4	3	3	0	0	2	4	3	1

				Készlet igény			
Foglaltak:	7	4	5				
Összesen:	10	5	7	-4	-1	-1	
Szabad ef. Szám:	3	1	2	2	-1	0	
				-3	1	2	
				3	0	1	P3 igénye kielégíthető
				-1	-2	1	

P3 lefut

R1 : 10 R2: 5 R3 : 7

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	3	0	7	2	3
p1	3	2	2	2	0	0	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	0	0	2	4	3	1

				Készlet igény			
Foglaltak:	7	4	5				
Összesen:	10	5	7	-2	0	0	
Szabad ef. Szám:	5	2	3	4	0	1	P1 igénye teljesíthető
				-1	2	3	
				5	1	2	LEFUTOTT
				1	-1	2	

P1 lefut

R1 : 10 R2: 5 R3 : 7

	Max. igény			Foglal			Kielégítetlen igények		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
p0	7	5	3	0	3	0	7	2	3
p1	3	2	2	**2**	**0**	**0**	1	2	2
p2	9	0	2	3	0	2	6	0	0
p3	2	2	2	**2**	**1**	**1**	0	1	1
p4	4	3	3	0	0	2	4	3	1

				Készlet igény			
Foglaltak:	7	4	5				
Összesen:	10	5	7	0	0	0	P0 igénye teljesíthető
Szabad ef. Szám:	7	2	3	6	0	1	LEFUTOTT
				1	2	3	
				7	1	2	LEFUTOTT
				3	-1	2	

P0 lefut

	R1 : 10	R2: 5	R3 : 7
	Max. igény		
	R1	R2	R3
p0	7	5	3
p1	3	2	2
p2	9	0	2
p3	2	2	2
p4	4	3	3

	Foglal		
	R1	R2	R3
p0	**0**	**3**	**0**
p1	**2**	**0**	**0**
p2	3	0	2
p3	**2**	**1**	**1**
p4	0	0	2

	Kielégítetlen igények		
	R1	R2	R3
p0	7	2	3
p1	1	2	2
p2	6	0	0
p3	0	1	1
p4	4	3	1

Foglaltak:	7	4	5
Összesen:	10	5	7
Szabad ef. Szám:	7	5	3

Készlet igény			
0	3	0	LEFUTOTT
6	3	1	LEFUTOTT
1	5	3	P2 igénye teljesíthető
7	4	2	LEFUTOTT
3	2	2	

P2 lefut

	R1 : 10	R2: 5	R3 : 7
	Max. igény		
	R1	R2	R3
p0	7	5	3
p1	3	2	2
p2	9	0	2
p3	2	2	2
p4	4	3	3

	Foglal		
	R1	R2	R3
p0	**0**	**3**	**0**
p1	**2**	**0**	**0**
p2	**3**	**0**	**2**
p3	**2**	**1**	**1**
p4	0	0	2

	Kielégítetlen igények		
	R1	R2	R3
p0	7	2	3
p1	1	2	2
p2	6	0	0
p3	0	1	1
p4	4	3	1

Foglaltak:	7	4	5
Összesen:	10	5	7
Szabad ef. Szám:	10	5	5

Készlet igény			
3	3	2	LEFUTOTT
9	3	3	LEFUTOTT
4	5	5	LEFUTOTT
10	4	4	LEFUTOTT
6	2	4	P4 igénye teljesíthető

P4 lefut

	R1 : 10	R2: 5	R3 : 7
	Max. igény		
	R1	R2	R3
p0	7	5	3
p1	3	2	2
p2	9	0	2
p3	2	2	2
p4	4	3	3

	Foglal		
	R1	R2	R3
p0	**0**	**3**	**0**
p1	**2**	**0**	**0**
p2	**3**	**0**	**2**
p3	**2**	**1**	**1**
p4	**0**	**0**	**2**

	Kielégítetlen igények		
	R1	R2	R3
p0	7	2	3
p1	1	2	2
p2	6	0	0
p3	0	1	1
p4	4	3	1

Foglaltak:	7	4	5
Összesen:	10	5	7
Szabad ef. Szám:	10	5	7

Készlet igény			
3	3	4	LEFUTOTT
9	3	5	LEFUTOTT
4	5	7	LEFUTOTT
10	4	6	LEFUTOTT
6	2	6	LEFUTOTT

A P1 kérelme teljesíthető

2. Gyakorló feladat: Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (5.3)., azaz

Írjanak három C nyelvű programot, ahol készít egy üzenetsort és ebbe két üzenetet tesz bele – **msgcreate.c,** majd olvassa ki az üzenetet - **msgrcv.c**, majd szüntesse meg az üzenetsort (takarít) - **msgctl.c.**

A futtatás eredményét is tartalmazza a jegyzőkönyv.

Mentés: msgcreate.c; msgrcv.c; msgctl.c.

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: msgcreate.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
#define MSGKEY 654321L

struct msgbuf1 {
        long mtype;
        char mtext[512];
} sndbuf, *msgp;



int main()
{
        int msgid;
        key_t key;
        int msgflg;
        int rtn, msgsz;

        key = MSGKEY;
        msgflg = 00666 | IPC_CREAT;
        msgid = msgget( key, msgflg);
        if ( msgid == -1) {
                perror("\nMsgget error!");
                exit(-1);
        }
        printf("Msgid %d, %x : ", msgid,msgid);

        msgp        = &sndbuf;
        msgp->mtype = 1;
        strcpy(msgp->mtext,"Message 1");
        msgsz       = strlen(msgp->mtext) + 1;

        rtn = msgsnd(msgid,(struct msgbuf *) msgp, msgsz, msgflg);
        printf("\nFirst msgsnd return: %d", rtn);
        printf("\nSent message: %s", msgp->mtext);

        strcpy(msgp->mtext,"Message 2");
        msgsz       = strlen(msgp->mtext) + 1;
        rtn = msgsnd(msgid,(struct msgbuf *) msgp, msgsz, msgflg);
        printf("\nSecond msgsnd return: %d", rtn);
        printf("\nSent message: %s\n", msgp->mtext);

        exit (0);
}
```

```
abraham1@jerry:~$ ./msgcreate.out
Msgid 557056, 88000 :
First msgsnd return: 0
Sent message: Message 1
Second msgsnd return: 0
Sent message: Message 2
```

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: msgrcv.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSGKEY 654321L

struct msgbuf1 {
        long mtype;
        char mtext[512];
} rcvbuf, *msgp;

struct msqid_ds ds, *buf;

int main()
{
        int msgid;
        key_t key;
        int mtype, msgflg;
        int rtn, msgsz;

        key = MSGKEY;
        msgflg = 00666 | IPC_CREAT | MSG_NOERROR;

        msgid = msgget( key, msgflg);
        if ( msgid == -1) {
                perror("Msgget error!");
                exit(-1);
        }
        printf("\nMsgid: %d",msgid);

        msgp = &rcvbuf;
        buf = &ds;
        msgsz = 20;
        mtype = 0;
        rtn = msgctl(msgid,IPC_STAT,buf);
        printf("\nNumber of messages: %d",buf->msg_qnum);

        while (buf->msg_qnum) {

                rtn = msgrcv(msgid,(struct msgbuf *)msgp, msgsz, mtype, msgflg);
                printf("\nRtn: %d, Got message: %s\n",rtn, msgp->mtext);
                rtn = msgctl(msgid,IPC_STAT,buf);
        }

        exit (0);
}
```

[Read 48 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page

```
abraham1@jerry:~$ ./msgrcv.out

Msgid: 557056
Number of messages: 4
Rtn: 10, Got message: Message 1

Rtn: 10, Got message: Message 2

Rtn: 10, Got message: Message 1

Rtn: 10, Got message: Message 2
abraham1@jerry:~$
```

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: msgctl.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSGKEY 654321L

int main()
{
        int msgid, msgflg,  rtn;
        key_t key;
        key = MSGKEY;
        msgflg = 00666 | IPC_CREAT;
        msgid = msgget( key, msgflg);

        rtn = msgctl(msgid, IPC_RMID, NULL);
        printf ("Returned: %d\n", rtn);

        exit (0);
}
```

```
abraham1@jerry:~$ ./msgctl.out
Returned: -1
abraham1@jerry:~$
```

3. Gyakorló feladat: Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzetet - a témához kapcsolódó fejezetét (5.3.2), azaz

Írjon három C nyelvű programot, ahol

- készít egy osztott memóriát, melyben választott kulccsal kreál/azonosít osztott memória szegmenst - **shmcreate.c**.
- az **shmcreate.c** készített osztott memória szegmens *státusának lekérdezése* – **shmctl.c**
- opcionális: **shmop.c** shmid-del azonosít osztott memória szegmenst. Ezután a segm nevű pointervál-tozót használva a processz virtuális címtartomanyába kapcsolja (attach) a szegmest (shmat() rendszerhívás). Olvassa, irja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

shmcreate.c

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: shmcreate.c

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


int main()
{
        int shmid;
        key_t key;
        int size=512;
        int shmflg;

        key = SHMKEY;

        shmflg = 0;
        if ((shmid=shmget( key, size, shmflg)) < 0) {
           printf("\nSegment creation.");
           shmflg = 00666 | IPC_CREAT;
           if ((shmid=shmget( key, size, shmflg)) < 0) {
              perror("\nShmget error!");
              exit(-1);
           }
        } else printf("\nThere is already a segment.");

        printf("\nShmid ID: %d \n", shmid);

        exit (0);
}
```

```
abraham1@jerry:~$ ./shmcreate.out

There is already a segment.
Shmid ID: 327680
abraham1@jerry:~$
```

shmctl.c

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: shmctl.c

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

int main()
{
        int shmid;
        key_t key;
        int size=512;
        int shmflg;
        int rtn;
        int cmd;
        struct shmid_ds shmid_ds, *buf;

        buf = &shmid_ds;

        key = SHMKEY;
        shmflg = 0;

        if ((shmid=shmget( key, size, shmflg)) < 0) {
            perror("\nShmget error!");
            exit(-1);
        }


        do {
                printf("\nGive the number of the command ");
                printf("\n0 IPC_STAT (status)");
                printf("\n1 IPC_RMID (torles)");
                printf("\nEnter the number: ");
                scanf("%d",&cmd);
        } while (cmd < 0 && cmd > 1);

        switch (cmd){
                case 0: rtn = shmctl(shmid, IPC_STAT, buf);
                        printf("\nSegment size: %d",buf->shm_segsz);
                        printf("\nLast shmop process pid: %d\n ",buf->shm_lpid);
                        break;
                case 1: rtn = shmctl(shmid, IPC_RMID, NULL);
                        printf("\nSegment deleted.\n");
        }

        exit(0);

}
```

```
abraham1@jerry:~$ ./shmctl.out

Give the number of the command
0 IPC_STAT (status)
1 IPC_RMID (torles)
Enter the number: 1

Segment deleted.
abraham1@jerry:~$ ./shmctl.out

Give the number of the command
0 IPC_STAT (status)
1 IPC_RMID (torles)
Enter the number: 0

Segment size: 512
Last shmop process pid: 0
 abraham1@jerry:~$
```

shmop.c :

jerry.iit.uni-miskolc.hu - PuTTY

GNU nano 2.7.4 File: shmop.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SHMKEY 123456L


int main()
{
        int shmid;
        key_t key;
        int size=512;
        int shmflg;
        struct vmi {
                int  hossz;
                char szoveg[512-sizeof(int)];
        } *segm;

        key = SHMKEY;
        shmflg = 0;

        if ((shmid=shmget( key, size, shmflg)) < 0) {
            perror("\nShmget error!");
            exit(-1);
        }

        shmflg = 00666 | SHM_RND;
        segm = (struct vmi *)shmat(shmid, NULL, shmflg);

        if (segm == (void *)-1) {
                perror("Unsuccessful attach.");
                exit (-1);
        }

        if (strlen(segm->szoveg) > 0)
                printf("\nOld text: %s (with %d length)",segm->szoveg,segm->hossz);

        printf("\nEnter new text: ");
        scanf("%s", segm->szoveg);
        printf("\nNew text: %s\n",segm->szoveg);
        segm->hossz=strlen(segm->szoveg);
        shmdt(segm);

        exit(0);

}
```

```
abraham1@jerry:~$ ./shmop.out

Old text: New (with 3 length)
Enter new text: String

New text: String
abraham1@jerry:~$
```