Operációs rendszerek BSc

10. gyakorlat

2021. április 26. (hétfő) 16:00 – 18:00

Készítette:

Szeli Márk

Gazdaságinformatikus

B8VNQ7

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), illetve P0 (0;2;0) kérése úgy, hogy biztonságos legyen, holtpontmentesség szempontjából a rendszer - a következő *kiinduló állapot* alapján. Igazolja a processzek végrehajtásának sorrendjét – számolással.

	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	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

Az IGÉNY mátrixot úgy kapom meg, hogy a MAX.IGÉNY mátrixból kivonom a FOGLAL mátrix elemeit.

	MAX. IGÉNY - FOGLAL		
	R1	**R2**	**R3**
P0	7-0	5-1	3-0
P1	3-2	2-0	2-0
P2	9-3	0-0	2-2
P3	2-2	2-1	2-1
P4	4-0	3-0	3-2

	IGÉNY		
	R1	**R2**	**R3**
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

P0 rendelkezik (0;2;0) kéréssel.

$$P0(0;1;0) + P0(0;2;0) = \boldsymbol{P0(0;3;0)}$$

	ÚJ FOGLAL		
	R1	**R2**	**R3**
P0	0	3	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2
SUM	7	4	5

	ÚJ IGÉNY		
	R1	**R2**	**R3**
P0	7	2	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

Készlet:

$R1: 10 - 7 = 3$
$R2: 5 - 4 = 1$
$R3: 7 - 5 = 2$

Készlet(3;1;2)

A jelenlegi készlettel a P3 processz igénye elégíthető ki.

	FOGLAL		
	R1	**R2**	**R3**
P0	0	3	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

	IGÉNY		
	R1	**R2**	**R3**
P0	7	2	3
P1	1	2	2
P2	6	0	0
P3	**0**	**1**	**1**
P4	4	3	1

A P3 processz lefut, az új készlet: $(3; 1; 2) + (2; 1; 1) = (5; 2; 3)$

A jelenlegi készlettel a P1 processz igénye elégíthető ki.

	FOGLAL		
	R1	**R2**	**R3**
P0	0	3	0
P1	2	0	0
P2	3	0	2
P4	0	0	2

	IGÉNY		
	R1	**R2**	**R3**
P0	7	2	3
P1	**1**	**2**	**2**
P2	6	0	0
P4	4	3	1

A P1 processz lefut, az új készlet: $(5; 2; 3) + (2; 0; 0) = (7; 2; 3)$

A jelenlegi készlettel a P0 processz igénye elégíthető ki.

	FOGLAL		
	R1	**R2**	**R3**
P0	0	3	0
P2	3	0	2
P4	0	0	2

	IGÉNY		
	R1	**R2**	**R3**
P0	**7**	**2**	**3**
P2	6	0	0
P4	4	3	1

A P0 processz lefut, az új készlet:$(7; 2; 3) + (0; 3; 0) = (7; 5; 3)$

A jelenlegi készlettel a P2 processz igénye elégíthető ki.

	FOGLAL		
	R1	**R2**	**R3**
P2	3	0	2
P4	0	0	2

	IGÉNY		
	R1	**R2**	**R3**
P2	**6**	**0**	**0**
P4	4	3	1

A P2 processz lefut, az új készlet:$(7; 5; 3) + (3; 0; 2) = (10; 5; 5)$

A jelenlegi készlettel a P4 processz igénye elégíthető ki.

	FOGLAL		
	R1	**R2**	**R3**
P4	0	0	2

	IGÉNY		
	R1	**R2**	**R3**
P4	**4**	**3**	**1**

A P4 processz lefut, az új készlet:$(10; 5; 5) + (0; 0; 2) = (10; 5; 7)$

Lehetséges sorrend: P3 → P1 → P0 → P2 → P4

Ezáltal biztonságos állapotban van holtpontmentesség szempontjából a rendszer.

b, P4 rendelkezik (3;3;0) kéréssel.

$$P4(0; 0; 2) + P4(3; 3; 0) = \boldsymbol{P0(3; 3; 2)}$$

	ÚJ FOGLAL		
	R1	**R2**	**R3**
P0	0	3	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	3	3	2
SUM	10	7	5

Készlet:

$R1: 10 - 10 = 0$
$R2: 5 - 7 = -2$
$R3: 7 - 5 = 2$
Készlet(0;-2;2)

Ezáltal a rendszer holtpontmentesség szempontjából nincs biztonságos állapotban.

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*.

B8VNQ7_msgcreate.c:

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

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

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("Az msgget rendszerhivas nem sikerult!\n");
            exit(-1);
        }

    printf("Az msgid %d, %x : \n ", msgid, msgid);

    msgp = &sndbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext, "Nev: Szeli Mark\n");
    msgsz = strlen(msgp->mtext) + 1;

    rtn = msgsnd(msgid,(struct msgbuf *) msgp, msgsz, msgflg);
    printf("Az 1. msgsnd visszaadott %d-t\n", rtn);
    printf("A kikuldott uzenet: %s\n ", msgp->mtext);

    strcpy(msgp->mtext, "NEPTUN kod: B8VNQ7\n ");
    msgsz = strlen(msgp->mtext) + 1;
    rtn = msgsnd(msgid,(struct msgbuf *) msgp, msgsz, msgflg);
    printf("A 2.  msgsnd visszaadott %d-t\n ", rtn);
    printf("A kikuldott uzenet: %s\n ", msgp->mtext);
    exit (0);
}
```

Eredmény:

```
mark@mark-K72Jr:~$ ./B8VNQ7_msgcreate.out
Az msgid 3, 3 :
 Az 1. msgsnd visszaadott 0-t
A kikuldott uzenet: Nev: Szeli Mark

 A 2.  msgsnd visszaadott 0-t
 A kikuldott uzenet: NEPTUN kod: B8VNQ7
```

B8VNQ7_msgrcv.c:

```c
#include <stdio.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;

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("Az msgget rendszerhivas nem sikerult!\n");
            exit(-1);
        }

    printf("Az msgid: %d\n", msgid);

    msgp = &rcvbuf;
    buf = &ds;
    msgsz = 20;
    mtype = 0;
    rtn = msgctl(msgid, IPC_STAT, buf);

    printf("Az uzenetek szama: %d\n", buf->msg_qnum);

    while (buf->msg_qnum)
        {
            rtn = msgrcv(msgid,(struct msgbuf *)msgp, msgsz, mtype, msgflg);
            printf("Az rtn: %d, a vett uzenet: %s\n", rtn, msgp->mtext);
            rtn = msgctl(msgid,IPC_STAT,buf);
        }
    exit (0);
}
```

B8VNQ7_msgctl.c:

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

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 ("Vissztert: %d\n", rtn);

    exit (0);
}
```

Eredményei:

```
mark@mark-K72Jr:~$ ./B8VNQ7_msgctl.out
Vissztert: 0
mark@mark-K72Jr:~$ ./B8VNQ7_msgrcv.out
Az msgid: 5
Az uzenetek szama: 0
```

2a. Írjon egy C nyelvű programot, melyben az egyik processz létrehozza az üzenetsort, és szövegeket küld bele, exit üzenetre kilép, másik processzben lehet választani a feladatok közül: üzenetek darabszámának lekérdezése, 1 üzenet kiolvasása, összes üzenet kiolvasása, üzenetsor megszüntetése, kilépés.

gyak10_2.c:

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

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

struct msqid_ds ds, *buf;

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

    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;
    msgid = msgget( key, msgflg);

    if ( msgid == -1)
    {
        perror("Az msgget rendszerhívás sikertelen!\n");
        exit(-1);
    }


    msgp = &sndbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext,"Szeli Mark, B8VNQ7");
    msgsz = strlen(msgp->mtext) + 1;
    rtn = msgsnd(msgid,(struct msgbuf *) msgp, msgsz, msgflg);

    if (msgp->mtext == "exit")
    {
        exit(0);
    }
```

```
47      key_t kulcs;
48      int tipus;
49      int meret;
50
51      kulcs = MSGKEY;
52      msgflg = 00666 | IPC_CREAT | MSG_NOERROR;
53
54      msgid = msgget(kulcs, msgflg);
55      if ( msgid == -1)
56      {
57          perror("Az msgget rendszerhívás sikertelen!\n");
58          exit(-1);
59      }
60
61      printf("Az uzenet ID-je: %d\n",msgid);
62
63      msgp = &sndbuf;
64      buf = &ds;
65      meret = 20;
66      tipus = 0;
67      rtn = msgctl(msgid,IPC_STAT,buf);
68
69      rtn = msgrcv(msgid,(struct msgbuf *)msgp, meret, tipus, msgflg);
70      rtn = msgctl(msgid,IPC_STAT,buf);
71
72      do{
73          printf("Kerem, valasszon a menusorbol!\n");
74          printf("0. Uzenet darabszamanak meghatarozasa\n");
75          printf("1. Az elso (1.) uzenet kiolvasasa\n");
76          printf("2. Az uzenetsor megszuntetese\n");
77          printf("3. Kilepes\n");
78          scanf("%d", &msgid);
79      }while(msgid < 0 || msgid > 3);
```

```
81      switch(msgid)
82      {
83        case 0:
84            printf("Az uzenetek szama: %ld\n",buf->msg_qnum);
85            break;
86
87        case 1:
88            printf("A kikuldott uzenet: %s\n", msgp->mtext);
89            break;
90
91        case 2:
92            rtn = msgctl(msgid, IPC_RMID, NULL);
93            printf("Az uzenetsor megszuntetesre kerult!\n");
94            break;
95
96        case 3:
97            exit(0);
98      }
99    }
```

Eredmények:

```
~$ ./gyak10_2.out
Az uzenet ID-je: 1
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
1
A kikuldott uzenet: Szeli Mark, B8VNQ7
```

```
~$ ./gyak10_2.out
Az uzenet ID-je: 1
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
2
Az uzenetsor megszuntetesre kerult!
```

```
~$ ./gyak10_2.out
Az uzenet ID-je: 1
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
3
~$
```

```
~$ ./gyak10_2.out
Az uzenet ID-je: 1
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
-1
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
4
Kerem, valasszon a menusorbol!
0. Uzenet darabszamanak meghatarozasa
1. Az elso (1.) uzenet kiolvasasa
2. Az uzenetsor megszuntetese
3. Kilepes
```

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 pointerváltozót használva a processz virtuális címtartomanyába kapcsolja (attach) a szegmest (shmat() rendszerhívás). Olvassa, írja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

B8VNQ7_shmcreate.c:

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

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

    key = SHMKEY;
    shmflg = 0;
    if ((shmid=shmget(key, size, shmflg)) < 0)
    {
        printf("Nincs meg szegmens! El kell kesziteni!\n");
        shmflg = 00666 | IPC_CREAT;

        if ((shmid=shmget(key, size, shmflg)) < 0)
            {
                perror("Az shmget system-call sikertelen!\n ");
                exit(-1);
            }
    }
    else
    {
        printf("Letezik mar szegmens!\n ");
    }

    printf("Az shmid azonositoja: %d\n", shmid);
    exit (0);
}
```

Eredmény:

```
mark@mark-K72Jr:~$ ./B8VNQ7_shmcreate.out
Nincs meg szegmens! El kell kesziteni!
Az shmid azonositoja: 32815
```

B8VNQ7_shmctl.c:

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

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("Az shmget system-call sikertelen!\n");
        exit(-1);
    }

    do {
        printf("Kerem, adja meg a parancs szamat!\n");
        printf("0 IPC_STAT (status)\n");
        printf("1 IPC_RMID (torles)\n");
        scanf("%d", &cmd);
        } while (cmd < 0 && cmd > 1);

    switch (cmd)
    {
        case 0: rtn = shmctl(shmid, IPC_STAT, buf);
            printf("Szegmens meret: %d", buf->shm_segsz);
            printf("Utolso shmop-os processzor PID: %d\n ",buf->shm_lpid);
            break;
        case 1: rtn = shmctl(shmid, IPC_RMID, NULL);
            printf("Szegmens torolve!\n");
    }
    exit(0);

}
```

Eredmények:

```
mark@mark-K72Jr:~$ ./B8VNQ7_shmctl.out
Kerem, adja meg a parancs szamat!
0 IPC_STAT (status)
1 IPC_RMID (torles)
0
Szegmens meret: 512Utolso shmop-os processzor PID: 0
```

```
mark@mark-K72Jr:~$ ./B8VNQ7_shmctl.out
Kerem, adja meg a parancs szamat!
0 IPC_STAT (status)
1 IPC_RMID (torles)
1
Szegmens torolve!
```

B8VNQ7_shmop.c:

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

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("\n Az shmget system-call sikertelen!");
        exit(-1);
    }

    shmflg = 00666 | SHM_RND;
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);
    if (segm == (void *)-1)
    {
        perror(" Sikertelen attach");
        exit (-1);
    }

    if (strlen(segm->szoveg) > 0)
    {
        printf("Regi szoveg: %s (%d hosszon)\n ", segm->szoveg, segm->hossz);
    }

    printf("Uj szoveget kerek!\n");
    gets(segm->szoveg);
    printf("Az uj szoveg: %s\n",segm->szoveg);
    segm->hossz=strlen(segm->szoveg);

    shmdt(segm);
    exit(0);
}
```

Eredmény:

```
mark@mark-K72Jr:~$ ./B8VNQ7_shmop.out
Uj szoveget kerek!
Szeli Mark, B8VNQ7
Az uj szoveg: Szeli Mark, B8VNQ7
```

3a. Írjon egy C nyelvű programot, melyben egyik processz létrehozza az osztott memóriát, másik processz rácsatlakozik az osztott memóriára, ha van benne valamilyen szöveg, akkor kiolvassa, majd beleír új üzenetet, harmadik processznél lehet választani a feladatok közül: status lekérése (szegmens mérete, utolsó shmop-os processzor pid-je), osztott memória megszüntetése, kilépés (2. és 3. proc. lehet egyben is).

gyak10_3.c:

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

int main() {
    int child = 0;

    if ((child = fork()) == 0) {
        int shmid;
        key_t key;
        int size=512;
        int shmflg;

        key = SHMKEY;
        shmflg = 0;
        if ((shmid=shmget( key, size, shmflg)) < 0)
        {
            printf("Nincs meg szegmens! El kell keszitenI!\n");
            shmflg = 00666 | IPC_CREAT;
            if ((shmid=shmget( key, size, shmflg)) < 0)
            {
                perror("Az shmget rendszerhivas sikertelen!\n");
                exit(-1);
            }
        }
        else
        {
            printf("Letezik mar szegmens!\n");
        }

        printf("Az shmid azonositoja %d:\n", shmid);

        exit (0);
    }
```

```
41      else
42      {
43          if (child = fork() == 0) {
44              int shmid;
45              key_t key;
46              int size=512;
47              int shmflg;
48              struct vmi {
49                  int  hossz;
50                  char szoveg[512-sizeof(int)];
51              } *segm;
52
53              key = SHMKEY;
54              shmflg = 0;
55              if ((shmid=shmget( key, size, shmflg)) < 0)
56              {
57                  perror("Az shmget rendszerhivas sikertelen!\n");
58                  exit(-1);
59              }
60
61              shmflg = 00666 | SHM_RND;
62              segm = (struct vmi *)shmat(shmid, NULL, shmflg);
63              if (segm == (void *)-1) {
64                  perror("Sikertelen attach!\n");
65                  exit (-1);
66              }
67
68              if (strlen(segm->szoveg) > 0)
69              {
70                  printf("Regi szoveg: %s (%d hosszon)\n",segm->szoveg,segm->hossz);
71              }
72
73              printf("Kerem, adjon meg uj szoveget!\n");
74              gets(segm->szoveg);
75              printf("Az uj szoveg: %s\n",segm->szoveg);
76              segm->hossz=strlen(segm->szoveg);
77
78              shmdt(segm);
79
80              exit(0);
81          }
```

```
83          else
84          {
85              int shmid;
86              key_t key;
87              int size=512;
88              int shmflg;
89              int rtn;
90              int cmd;
91              struct shmid_ds shmid_ds, *buf;
92              buf = &shmid_ds;
93
94              key = SHMKEY;
95              shmflg = 0;
96              if ((shmid=shmget( key, size, shmflg)) < 0)
97              {
98                  perror("Az shmget rendszerhivas sikertelen!\n");
99                  exit(-1);
100             }
101
102         do {
103             printf("Kerem, adja meg a parancs szamat!\n");
104             printf("0 IPC_STAT (status)\n");
105             printf("1 IPC_RMID (torles)   >  ");
106             scanf("%d",&cmd);
107             } while (cmd < 0 && cmd > 1);
108
109             switch (cmd)
110             {
111             case 0: rtn = shmctl(shmid, IPC_STAT, buf);
112                 printf("Segm. meret: %ld\n",buf->shm_segsz);
113                 printf("Utolso shmop-os processz PID: %d\n ",buf->shm_lpid);
114                 break;
115             case 1: rtn = shmctl(shmid, IPC_RMID, NULL);
116                 printf("Szegmens sikeresen torolve!\n");
117             }
118
119             exit(0);
120         }
121     }
122 }
```

Eredmények:

```
~$ ./gyak10_3.out
Letezik mar szegmens!
Az shmid azonositoja 2:
Kerem, adja meg a parancs szamat!
0 IPC_STAT (status)
1 IPC_RMID (torles)   >  Kerem, adjon meg uj szoveget!
0
Segm. meret: 512
Utolso shmop-os processz PID: 709
```

```
Letezik mar szegmens!
Kerem, adja meg a parancs szamat!
0 IPC_STAT (status)
Az shmid azonositoja 2:                (3 hosszon)
Kerem, adjon meg uj szoveget! szoveg:
1
Szegmens sikeresen torolve!
```