

Operációs rendszerek BSc

10. Gyak.

2022. 04. 11.

Készítette:

Szkárosi Szilárd Bsc

Mérnökinformatikus

DLWGQZ

Miskolc, 2022

1.feladat –

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 P1 (1,0,2), P4 (3,3,0) ill. P0 (0,2,0) kérése úgy, hogy biztonságos legyen, holtpontmentesség szempontjából a rendszer - a következő *kiinduló állapot* alapján.

Külön-külön táblázatba oldja meg a feladatot!

a) Határozza meg a processzek által igényelt erőforrások mátrixát?

b) Határozza meg *pillanatnyilag szabad erőforrások számát*?

c) Igazolja, magyarázza az egyes *processzek* végrehajtásának *lehetséges sorrendjét* - számolással?"

	MAX. IGÉNY				FOGLALÁS				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			
									R1	R2	R3	
Foglaltak					7	2	5		-4	-1	-1	p0
Összesen					10	5	7		2	1	0	p1
Szabad erőforrás szám					3	3	2		-3	3	2	p2
									3	2	1	p3
									-1	0	1	p4

2. feladat – Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetékét, a gyerek processz beleír egy szöveget a csővezetékbe (A kiírt szöveg: XY neptunkod), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.

Mentés: neptunkod_unnamed.c

```

1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     int fd[2];
7     int child;
8
9     if (pipe(fd))
10     {
11         perror("pipe");
12         return 1;
13     }
14
15     child = fork();
16
17     if (child > 0)
18     {
19         char s[1024];
20         close(fd[1]);
21         read(fd[0], s, sizeof(s));
22         printf("%s", s);
23         close(fd[0]);
24     }
25
26     else if (child == 0)
27     {
28         close(fd[0]);
29         write(fd[1], "SZSZ DLWGQZ\n", 17);
30         close(fd[1]);
31     }
32
33     return 0;
34 }
35
36 }

```

```

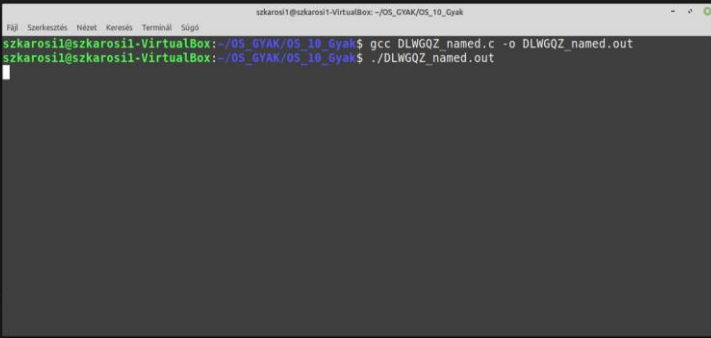
szkarosil@szkarosil-VirtualBox: ~/OS_GYAK/OS_10_Gyak
$ cd OS_GYAK/OS_10_Gyak
szkarosil@szkarosil-VirtualBox: ~/OS_GYAK/OS_10_Gyak$ gcc DLWGQZ_unnamed.c -o DLWGQZ_unnamed.out
szkarosil@szkarosil-VirtualBox: ~/OS_GYAK/OS_10_Gyak$ ./DLWGQZ_unnamed.out
SZSZ DLWGQZ
szkarosil@szkarosil-VirtualBox: ~/OS_GYAK/OS_10_Gyak$

```

3. feladat – Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetékét (neve: neptunkod), a gyerek processz beleír egy szöveget a csővezetékbe (A hallgató neve: pl.: Keszérű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.

Mentés: neptunkod_named.c

```
1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6
7 int main()
8 {
9     int child;
10
11     mkfifo("Keszaru Otto", S_IRUSR | S_IWUSR);
12     child = fork();
13
14     if (child > 0)
15     {
16         char s[1024];
17         int fd;
18
19         fd = open("Keszaru Otto", O_RDONLY);
20         read(fd, s, sizeof(s));
21         printf("%s", s);
22         close(fd);
23         unlink("Keszaru Otto");
24     }
25     else if (child == 0)
26     {
27         int fd = open("Keszaru Otto", O_WRONLY);
28         write(fd, "SZSZ DLWGQZ\n", 17);
29         close(fd);
30     }
31
32     return 0;
33 }
```



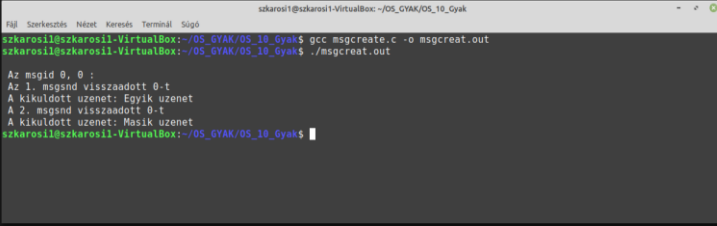
4. Gyakorló feladat – Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (5.3.), azaz

Írjon 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.

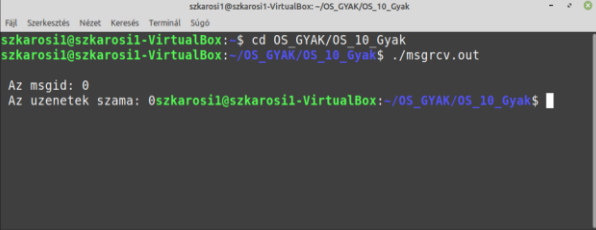
Mentés: msgcreate.c; msgrcv.c; msgctl.c.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/types.h>
5 #include <sys/msg.h>
6 #include <sys/stat.h>
7 #define MSGKEY 0543211
8
9 struct msgbuf {
10     long mtype;
11     char mtext[1024];
12 } msgbuf;
13
14 int main()
15 {
16     int msgid;
17     key_t key;
18     int msgflg;
19     int rtn;
20
21     key = MSGKEY;
22     msgflg = 0666 | IPC_CREAT;
23     msgid = msgget(key, msgflg);
24     if (msgid == -1)
25     {
26         perror("The msgget system call failed!");
27         exit(1);
28     }
29     printf("\n Az msgid %d, key : ", msgid, key);
30
31     msgp = (struct msgbuf *) malloc(sizeof(msgbuf));
32     msgp->mtype = 1;
33     strcpy(msgp->mtext, "Egyik üzenet");
34     msgp->mtext[strlen(msgp->mtext)] = '\0';
35     /* Az üzenet hossza */
36     /* Az üzenet tartalma */
37     rtn = msgsnd(msgid, (struct msgbuf *) msgp, sizeof(msgp->mtext), 0);
38     printf("\n Az 1. msgsnd visszaadott %d", rtn);
39     printf("\n A kiküldött üzenet: %s", msgp->mtext);
40
41     strcpy(msgp->mtext, "Másik üzenet");
42     msgp->mtext[strlen(msgp->mtext)] = '\0';
43     rtn = msgsnd(msgid, (struct msgbuf *) msgp, sizeof(msgp->mtext), 0);
44     printf("\n A 2. msgsnd visszaadott %d", rtn);
45     printf("\n A kiküldött üzenet: %s", msgp->mtext);
46     printf("\n");
47     exit(0);
48 }
```



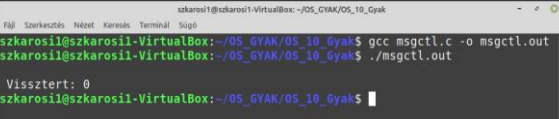
msgcreate.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/ipc.h>
5 #include <sys/msg.h>
6 #define MSGKEY 654321L
7
8 struct msgbuf {
9     long mtype;
10     char mtext[1024];
11 }
12
13 } rdbuf; /* message buffer as pointer */
14
15 struct msgid {
16     /* message identifier */
17     key_t key;
18     /* key */
19     int mtype; /* message type */
20     int rtn; /* return value */
21 }
22
23 key = MSGKEY; /* key */
24 msgflg = 0666 | IPC_CREAT; /* flags */
25 msgid = msgget(key, msgflg);
26 if (msgid == -1) {
27     perror("The msgget system call failed");
28     exit(1);
29 }
30 printf("Az msgid: %d\n", msgid);
31
32 rdbuf = (struct msgbuf *) malloc(sizeof(struct msgbuf));
33 if (rdbuf == NULL) {
34     perror("The malloc system call failed");
35     exit(1);
36 }
37 mtype = 0; /* message type */
38 rtn = msgctl(msgid, IPC_RMID, 0); /* remove the message queue */
39 printf("Az azonosító: %d\n", rtn);
40
41 while (1) {
42     /* read a message */
43     rtn = msgrecv(msgid, (struct msgbuf *) rdbuf, mtype, msgflg); /* receive a message */
44     printf("Az azonosító: %d, a küldő: %d, a fogadó: %d\n", rdbuf->mtype, rdbuf->key, rdbuf->key);
45     rtn = msgctl(msgid, IPC_STAT, rdbuf); /* get the message queue status */
46     printf("Az azonosító: %d, a fogadó: %d\n", rtn, rdbuf->key);
47 }
48 exit(0);
```



msgrcv.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/ipc.h>
5 #include <sys/msg.h>
6 #define MSGKEY 654321L
7
8 int main()
9 {
10     int msgid, msgflg, rtn;
11     key_t key;
12     key = MSGKEY;
13     msgflg = 0666 | IPC_CREAT;
14     msgid = msgget(key, msgflg);
15
16     rtn = msgctl(msgid, IPC_RMID, 0); /* remove the message queue */
17     printf("Az azonosító: %d\n", rtn);
18     exit(0);
19 }
```



msgctl.c

4a. Gyakorló feladat – 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.

Mentés: **gyak10_4.c**

```

struct msgbuf1
{
    long mtype;
    char mtext[256];
} sndbuf, *msgp;

int main()
{
    int id;
    key_t key;
    int flag;
    int rtn, size;
    int ok = 1, count = 1;

    char teszt[256];
    key = MSGKEY;
    flag = 00666 | IPC_CREAT;
    id = msgget( key, flag);
    if ( id == -1)
    {
        perror("\n Az msgget hivas nem valosult meg");
        exit(-1);
    }

    do
    {
        scanf("%s",teszt);
        msgp = &sndbuf;
        msgp->mtype = 1;                /*tipus = text*/
        size = strlen(msgp->mtext) + 1; /* az uzenet hossza */

        if(strcmp("exit",teszt) != 0)
        {
            rtn = msgsnd(id,(struct msgbuf *) msgp, size, flag);
            printf("\n Az %d. msgsnd visszaadott %d-t", count,rtn);
            printf("\n A kikuldott uzenet: %s\n", msgp->mtext);
            count++;
        }
        else
        {
            ok = 0;
            printf("\nKilepes\n");
        }
    } while (ok == 1);

    return 0;
}

```

szkarosi1@szkarosi1-VirtualBox: ~/OS_GYAK/OS_10_Gyak

Fájl Szerkesztés Nézet Keresés Terminál Súgó

szkarosi1@szkarosi1-VirtualBox:~/OS_GYAK/OS_10_Gyak\$./gyak10_4.out
Szia!

Az 1. msgsnd visszaadott 1-t
A kikuldott uzenet:
Hali

Az 2. msgsnd visszaadott 1-t
A kikuldott uzenet:

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

Írjon három C nyelvű programot, ahol

- készítsen 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 címtartományába kapcsolja (attach) a szegmest (shmat() rendszerhívás). Olvassa, írja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

#define KEY 2022

int main()
{
    int sharedMemoryId = shmget(KEY, 256, IPC_CREAT | 0666);

    return 0;
}
```

shmcreate.c

```

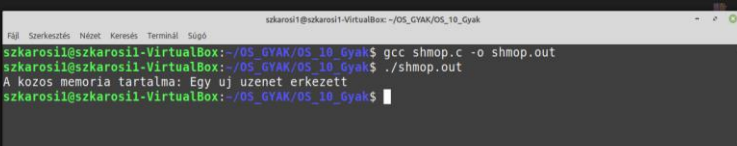
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

#define KEY 2022

void main()
{
    int sharedMemoryId = shmget(KEY, 0, 0);
    struct shmid_ds buffer;
    if (shmctl(sharedMemoryId, IPC_STAT, &buffer) == -1 )
    {
        perror("Nem sikerult az adatokat lekerdezni");
        exit(-1);
    }
}

```

shmctl.c



```

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

#define KEY 2022

void main()
{
    int sharedMemoryId = shmget(KEY, 0, 0);
    char *segn = shmat(sharedMemoryId, NULL, SHM_RDONLY);
    strcpy(segn, "Egy uj uzenet érkezett");
    printf("A kozos memoria tartalma: %s\n", segn);
    shmdt(segn);
}

```

shmop.c

5a. Gyakorló feladat – 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: státus lekérése (szegmens mérete, utolsó shmop-os proc. pid-je), osztott memória megszüntetése, kilépés (2. és 3. proc. lehet egyben is)”

```

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
#include <unistd.h>

#define KEY 777777

void main()
{
    pid_t process1; //foglalja le az osztott memoriát
    pid_t process2;
    pid_t process3;

    process1 = fork();
    if (process1 == 0)
    {
        int sharedMemoryId = shmget(KEY, 256, IPC_CREAT | 0666);
        if (sharedMemoryId == -1)
        {
            perror("Nem sikerült lefoglalni a memoriát\n");
            exit(-1);
        }
        printf("Process1 lefoglalta a memoriát!\n");
    }
    else
    {
        process2 = fork();
        if (process2 == 0)
        {
            printf("Process 2 olvas\n");
            int sharedMemoryId = shmget(KEY, 0, 0);
            char *s = shmat(sharedMemoryId, NULL, SHM_RND);
            strlen(s) > 0 ? printf("osztott memoriában szereplo szoveg : %s\n", s)
                : printf("Nincs benne szoveg\n");
            //beleirunk
            strcpy(s, "Ez egy uj szoveg");
            printf("process2 kuldté az üzenetet.\n");
        }
    }
}

```



```

else
{
    process2 = fork();
    if (process2 == 0)
    {
        printf("Process 2 olvas\n");
        int sharedMemoryId = shmget(KEY, 0, 0);
        char *s = shmat(sharedMemoryId, NULL, SHM_RND);
        strlen(s) > 0 ? printf("osztott memoriaban szereplo szoveg : %s\n", s)
        : printf("Nincs benne szoveg\n");
        //beleirunk
        strcpy(s, "Ez egy uj szoveg");
        printf("process2 kuldtte az uzenetet.\n");
    }
    else
    {
        process3 = fork();
        if (process3 == 0)
        {
            printf("process3: \n");
            int sharedMemoryId = shmget(KEY, 0, 0);
            struct shmid_ds buffer;
            if (shmctl(sharedMemoryId, IPC_STAT, &buffer) == -1)
            {
                perror("Nem sikerult lekerdezni.\n");
                exit(-1);
            }
            printf("Szegmens merete: %ld\n", buffer.shm_segsz);
            printf("utolso operaciot kiado processz pidje : %d\n", buffer.shm_lpid);
        }
    }
}
}
}

```

Gyak10_5.c futtatásakor:

```

szkarosil@szkarosil-VirtualBox:~$ cd OS_GYAK/OS_10_Gyak
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_10_Gyak$ gcc gyak10_5.c -o gyak10_5.out
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_10_Gyak$ ./gyak10_5.out
bash: ./: Ez egy könyvtár
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_10_Gyak$ ./gyak10_5.out
szkarosil@szkarosil-VirtualBox:~/OS_GYAK/OS_10_Gyak$ process3:
Szegmens merete: 256
utolso operaciot kiado processz pidje : 4397
Process 2 olvas
osztott memoriaban szereplo szoveg : Ez egy uj szoveg
process2 kuldtte az uzenetet.
Process1 lefoglalta a memoriát!

```