

Operációs rendszerek BSc

10. Gyak.

2022. 04. 11.

Készítette:

Juhász Balázs BSc

Mérnökinformatika

ZUYISF

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				-4	-1	-1	p0
Összesen				10				2	1	0	p1
Szabad erőforrás szám				3				-3	3	2	p2
								3	2	1	p3
								-1	0	1	p4

2. feladat – Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetékot, 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

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    int fd[2];
    int child;

    if (pipe(fd))
    {
        perror("pipe");
        return 1;
    }

    child = fork();

    if (child > 0)
    {
        char s[1024];
        close(fd[1]);
        read(fd[0], s, sizeof(s));
        printf("%s", s);

        close(fd[0]);
    }

    else if (child == 0)
    {
        close(fd[0]);
        write(fd[1], "JB ZUYISF\n", 17);
        close(fd[1]);
    }

    return 0;
}
```

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.: Keserű 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("Keseru 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("Keseru Otto", O_RDONLY);
20         read(fd, s, sizeof(s));
21         printf("%s", s);
22         close(fd);
23         unlink("Keseru Otto");
24     }
25     else if (child == 0)
26     {
27         int fd = open("Keseru Otto", O_WRONLY);
28         write(fd, "JB ZUVISF\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.

msgcreate.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/types.h>
5 #include <sys/ipc.h>
6 #include <sys/msg.h>
7 #define MSGKEY 654321
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, mtype;
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("In Az msgid %d, key : ", msgid, key);
30
31     msg = &msgbuf;
32     mtype = 1;
33     strcpy(msg->mtext, "Egik üzenet");
34     mtype = strlen(msg->mtext) + 1;
35     rtn = msgsnd(msgid, (struct msgbuf *) msg, mtype, msgflg);
36     printf("In Az 1. msgend visszaszolgált %d", rtn);
37     printf("In A kiküldött üzenet: %s", msg->mtext);
38
39     strcpy(msg->mtext, "Másik üzenet");
40     mtype = strlen(msg->mtext) + 1;
41     rtn = msgsnd(msgid, (struct msgbuf *) msg, mtype, msgflg);
42     printf("In Az 2. msgend visszaszolgált %d", rtn);
43     printf("In A kiküldött üzenet: %s", msg->mtext);
44     printf("\n");
45     exit(0);
46 }

```

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 struct msgbuf
9 {
10     long mtype;
11     char mtext[1024];
12 } rcvbuf, *mapp;
13
14 struct msgid_da da;
15
16
17
18
19 int main()
20 {
21     int msgid;
22     key_t key;
23     int mtype, msgflg;
24     int rtn, msize;
25
26     key = MSGKEY;
27     msgflg = 0666 | IPC_CREAT | MSG_NOERROR;
28
29     msgid = msgget(key, msgflg);
30     if (msgid == -1)
31     {
32         perror("\n The msgget system call failed!");
33         exit(-1);
34     }
35     printf("\n As msgid: %d", msgid);
36
37     mapp = &rcvbuf;
38     buf = &da;
39     msize = 20;
40     mtype = 0;
41     rtn = msgctl(msgid, IPC_STAT, buf);
42     printf("\n As msgstat: msize: %d\n", buf->msg_qnum);
43
44     while (buf->msg_qnum)
45     {
46         rtn = msgrcv(msgid, (struct msgbuf *)mapp, msize, mtype, msgflg);
47         printf("\n As rtn: %d, a text message: %s\n", rtn, mapp->mtext);
48         rtn = msgctl(msgid, IPC_STAT, buf);
49     }
50
51     exit(0);
52 }
```

msgctl.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, NULL);
17     printf ("\n Visszert: %d\n", rtn);
18
19     exit (0);
20 }
```

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

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
#define NKEYS 4096

struct magbui
{
    long mtype;
    char mtext[1024];
} *magbui, *magp;

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

    char test[1024];
    key = NKEYS;
    flag = 0666 | IPC_CREAT;
    id = shmget(key, flag);
    if (id == -1)
    {
        perror("Az shmget hívás nem valósult meg");
        exit(-1);
    }

    do
    {
        scanf("%s", test);
        magp = shmat(id, test, 0);
        size = strlen(magp->mtext) + 1;
        if (strlen(test) != 0)
        {
            rtn = shmctl(id, SHM_REM, magp, size, flag);
            printf("A %d. megadott válasz: %s", count, id);
            printf("A kibőlött üzenet: %s", magp->mtext);
            count++;
        }
        else
        {
            ok = 0;
            printf("Véglegesítés");
        }
    } while (ok == 1);

    return 0;
}
```

5. Gyakorló feladat – Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzet - 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 szegmenst (shmat()) rendszerhívás). Olvassa, írja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

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

int main()
{
    int sharedMemoryId = shmget(KEY, 256, IPC_CREAT | 0666);
    return 0;
}
```

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);
    struct shmctl_data buffer;
    if (shmctl(sharedMemoryId, IPC_STAT, &buffer) == -1)
    {
        perror("Ün siferult az adatokat lehetek szerezni");
        exit(-1);
    }
}
```

shmop.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 *sema = shmctl(sharedMemoryId, NMFL, SHM_RND);
    strcpy(sema, "Ez az közös szöveg");
    printf("A közös memória tartalma: %s\n", sema);
    shmctl(sema);
}
```