

Operációs rendszerek BSc

10.Gyak

2022.04.11.

Készítette:

Urbán Olivér BSc

Szak:

Mérnökinformatikus

Neptunkód: HEPMIU

2022.04.11.

Feladatok

„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 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?”

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1																						
2																						
3																						
4	P0																					
5	P1																					
6	P2																					
7	P3																					
8	P4																					
9																						
10																						
11																						
12																						
13																						
14																						
15																						
16																						
17																						
18																						
19																						
20																						
21																						
22																						
23																						
24																						
25																						
26																						
27																						
28																						
29																						
30																						
31																						

2. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezeték, 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 <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5
6 int main() {
7     int pipefd[2];
8     pid_t opid;
9     char buf;
10    char srecv[32];
11    int ret;
12
13    if (pipe(pipefd) == -1) {
14        perror("pipe");
15        exit(-1);
16    }
17
18    printf("fd: %d, fd: %d\n", getpid(), pipefd[0], pipefd[1]);
19
20    opid = fork();
21    if (opid == -1) {
22        perror("fork");
23        exit(-1);
24    }
25
26    if (opid == 0) {
27        printf("fd: I'm a children\n", getpid());
28        close(pipefd[1]);
29
30        printf("fd: what's int the pipe\n", getpid(), getfd());
31        while (read(pipefd[0], sbuf, 1) > 0) {
32            printf("%s", sbuf);
33        }
34        printf("\nthe pipe's other side is closed\n", getpid());
35        close(pipefd[0]);
36        exit(0);
37    } else {
38        printf("fd: I'm the parent\n", getpid());
39    }
40 }

```

3. 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 <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <fcntl.h>
6 #include <sys/stat.h>
7 #include <sys/types.h>
8 #include <sys/wait.h>
9
10 int main() {
11     int fd, ret;
12     char buf[32];
13     buf[0] = 0;
14
15     ret = mkfifo("fifo", 0666);
16     if (ret == -1) {
17         perror("mkfifo");
18         exit(-1);
19     }
20
21     fd = open("fifo", O_RDONLY);
22     if (fd == -1) {
23         perror("open");
24         exit(-1);
25     }
26
27     strcpy(buf, "I put this in the fifo\n");
28     printf("write in the fifo: %s\n", buf, strlen(buf));
29     write(fd, buf, strlen(buf));
30
31     ret = read(fd, buf, 32);
32     printf("read() read %d byte, these are: %s\n", ret, buf);
33     close(fd);
34     unlink("fifo");
35 }

```

4. Gyakorló feladat

Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó

fejezetét (5.3)., azaz

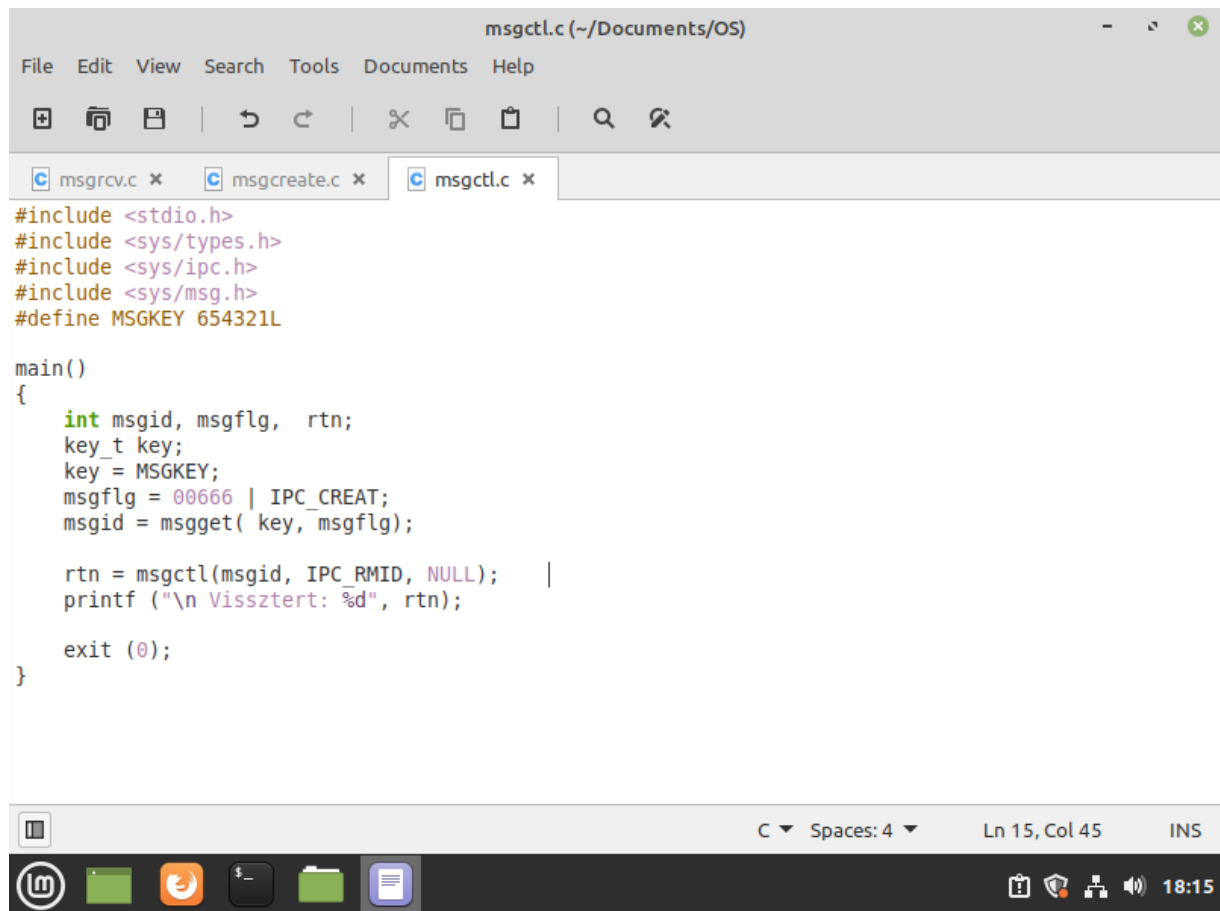
Írjon három C nyelvű programot, ahol készít egy üzenetsort és ebbe két üzenetet tesz bele –

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.



```
msgctl.c (~/Documents/OS)
File Edit View Search Tools Documents Help
+ [ ] [ ] | ↶ ↷ | ✂ [ ] [ ] | 🔍 [ ]
[ ] msgrcv.c x [ ] msgcreate.c x [ ] msgctl.c x

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

    exit (0);
}
```

C Spaces: 4 Ln 15, Col 45 INS

18:15

msgrcv.c (~/Documents/OS)

File Edit View Search Tools Documents Help

msgrcv.c x

msgcreate.c x

```
#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 msgid_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("\n A msgget rendszer hívás sikertelen!");
        exit(-1);
    }
    printf("\n Az msgid: %d", msgid);

    msgp = &rcvbuf;
    buf = &ds;
    msgsz = 20;
    mtype = 0;
    rtn = msgctl(msgid, IPC_STAT, buf);
    printf("\n Az üzenetek száma: %d", buf->msg_qnum);

    while (buf->msg_qnum) {
        rtn = msgrcv(msgid, (struct msgbuf *)msgp, msgsz, mtype, msgflg);
        printf("\n Az rtn: %d, a vett üzenet: %s\n", rtn, msgp->mtext);
        rtn = msgctl(msgid, IPC_STAT, buf);
    }
    exit(-1);
}
```

C Spaces: 4 Ln 27, Col 63 INS



msgcreate.c (~/Documents/OS)

File Edit View Search Tools Documents Help

msgrcv.c x

msgcreate.c x

```
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("\n A msgget rendszer hívás sikertelen!");
        exit(-1);
    }
    printf("\n Az msgid %d, %x : ", msgid, msgid);

    msgp = &sndbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext, "Egyik üzenet");
    msgsz = strlen(msgp->mtext) + 1;
    /* es elkuldom: */
    rtn = msgsnd(msgid, (struct msgbuf *)msgp, msgsz, msgflg);
    printf("\n Az 1. msgsnd visszaadott %d-t", rtn);
    printf("\n A kikuldott üzenet: %s", msgp->mtext);

    strcpy(msgp->mtext, "Masik üzenet");
    msgsz = strlen(msgp->mtext) + 1;
    rtn = msgsnd(msgid, (struct msgbuf *)msgp, msgsz, msgflg);
    printf("\n A 2. msgsnd visszaadott %d-t", rtn);
    printf("\n A kikuldott üzenet: %s", msgp->mtext);
    printf("\n");

    exit(0);
}
```

C Spaces: 4 Ln 25, Col 63 INS



4a. 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

A futtatás eredményét is tartalmazza a jegyzőkönyv.

5. Gyakorló feladat: Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzet -

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.

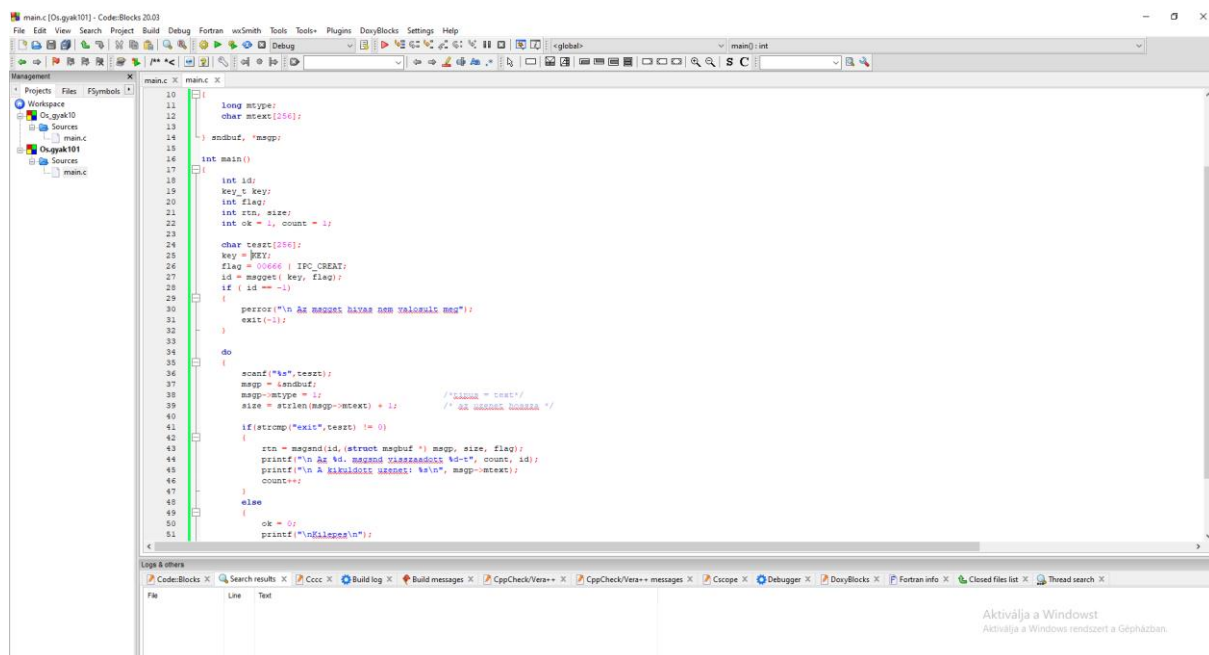
az shmcreate.c készített osztott memória szegmens státuszának lekérdezése – shmctl.c

opcionális: shmop.c shmid-del azonosít osztott memória szegmenst. Ezután a segm

nevű pointintervál-tozót használva a processz virtuális címtartományába kapcsolja

(attach) a szegmest (shmat()) rendszerhívás). Olvassa, írja ezt a címtartományt, végül

lekapcsolja (detach) a shmdt()) rendszerhívással).



```
10 // 10. Gyakorlat: Üzenetsorok és osztott memória
11 // 4a. Feladat: Írjon egy C nyelvű programot, melyben
12 // az egyik processz létrehozza az üzenetsort, és szövegeket küld bele, exit üzenetre
13 // kilép,
14 // másik processzben lehet választani a feladatok közül: üzenetek darabszámának
15 // lekérdezése, 1 üzenet kiolvasása, összes üzenet kiolvasása, üzenetsor megszüntetése,
16 // kilépés.
17 // Mentés: gyak10_4.c
18 // A futtatás eredményét is tartalmazza a jegyzőkönyv.
19
20 #include <stdio.h>
21 #include <stdlib.h>
22 #include <unistd.h>
23 #include <sys/types.h>
24 #include <sys/ipc.h>
25 #include <sys/shm.h>
26 #include <semaphore.h>
27
28 // Definíciók
29 #define MSG_SIZE 256
30 #define SEM_NAME "/shmsem"
31 #define SHM_NAME "/shm"
32
33 // Globális változók
34 long mtype;
35 char mtext[MSG_SIZE];
36 struct msgbuf {
37     long mtype;
38     char mtext[MSG_SIZE];
39 } msgbuf;
40
41 // Függvények
42 int main()
43 {
44     int id;
45     key_t key;
46     int flag;
47     int rtm, size;
48     int ok = 1, count = 1;
49
50     // Üzenetsor létrehozása
51     key = ftok(SHM_NAME, 1);
52     flag = IPC_CREAT | 0666;
53     id = msgget(key, flag);
54     if (id == -1)
55     {
56         perror("Nem lehet létrehozni az üzenetsortot");
57         exit(-1);
58     }
59
60     // Üzenet küldése
61     do
62     {
63         printf("Küldök üzenetet: %s\n", mtext);
64         if (strncpy(msgbuf.mtext, mtext, MSG_SIZE) != 0)
65             msgbuf.mtype = mtype;
66         if (sendmsg(id, (struct msgbuf *) &msgbuf, 0) != 0)
67             perror("Nem lehet küldeni az üzenetet");
68         count++;
69     } while (count < 10);
70
71     // Üzenetsor megszüntetése
72     ok = 0;
73     printf("Kilépés\n");
74     return 0;
75 }
```

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

A futtatás eredményét is tartalmazza a jegyzőkönyv.

Mentés: gyak10_5.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/ipc.h>
5 #include <sys/shm.h>
6 #include <string.h>
7 #include <unistd.h>
8
9 #define KEY 12345
10
11 void main()
12 {
13     pid_t process1;
14     pid_t process2;
15     pid_t process3;
16
17     process1 = fork();
18     if (process1 == 0)
19     {
20         int sharedMemoryId = shmop(KEY, 1024, IPC_CREAT | 0660);
21         if (sharedMemoryId == -1)
22         {
23             perror("Nem sikerült lefoglalni a memóriát");
24             exit(-1);
25         }
26         printf("Process1 lefoglalta a memóriát\n");
27     }
28     else
29     {
30         process2 = fork();
31         if (process2 == 0)
32         {
33             printf("Process 2 állapota\n");
34             int sharedMemoryId = shmop(KEY, 0, 0);
35             char *a = shmat(sharedMemoryId, NULL, SHM_RDONLY);
36             strlen(a) > 0 ? printf("Aktuális memóriában szereplő szöveg : %s\n", a) : printf("Ez az a memóriában szereplő szöveg : %s\n", a);
37             strcpy(a, "Ez az új szöveg");
38             printf("Process2 beírta az új szöveget\n");
39         }
40         else
41         {
42             process3 = fork();
43             if (process3 == 0)
44             {
45                 printf("Process3 állapota\n");
46                 int sharedMemoryId = shmop(KEY, 0, 0);
47                 printf("Process3 kiírta az új szöveget\n");
48             }
49         }
50     }
51 }
```