Operációs rendszerek BSc

10. Gyak.

2022. 04. 11.

Készítette:

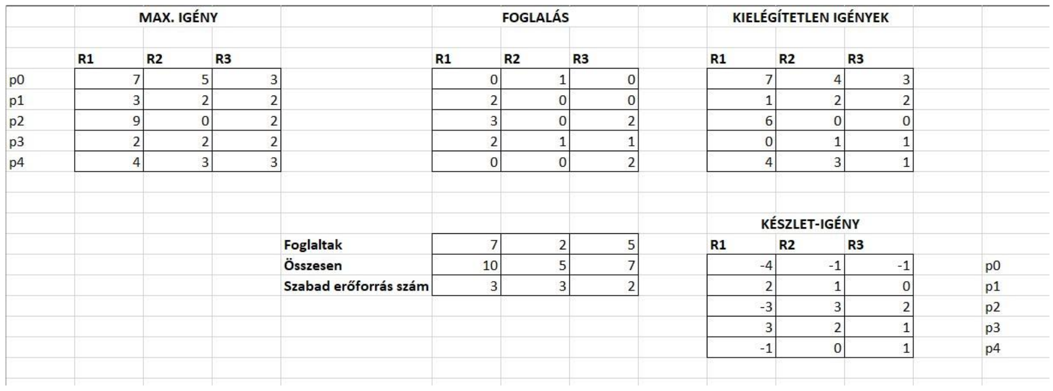
Nyíri Levente Bsc

Mérnökinformatikus

F023QC

Miskolc, 2022

1.feladat

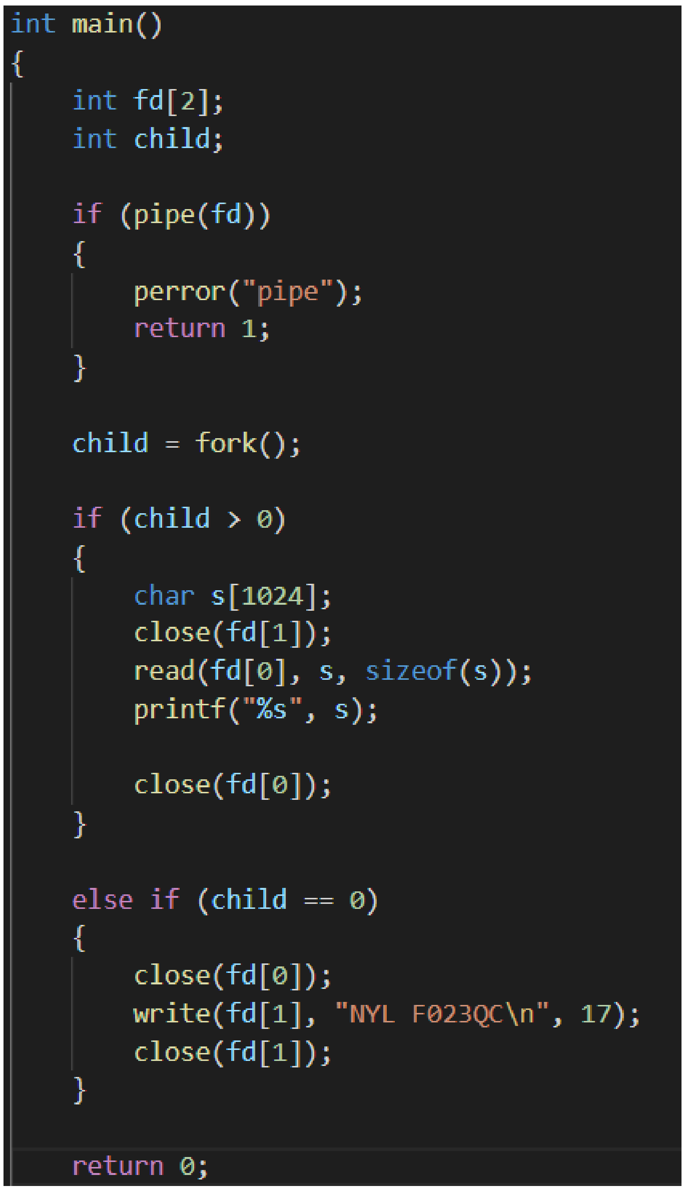


	MAX. IGÉNY				FOGLALÁS			KIELÉGÍTETLEN IGÉNYEK		
	R1	**R2**	**R3**		**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
p0	7	5	3		0	1	0	7	4	3
p1	3	2	2		2	0	0	1	2	2
p2	9	0	2		3	0	2	6	0	0
p3	2	2	2		2	1	1	0	1	1
p4	4	3	3		0	0	2	4	3	1

Foglaltak	7	2	5
Összesen	10	5	7
Szabad erőforrás szám	3	3	2

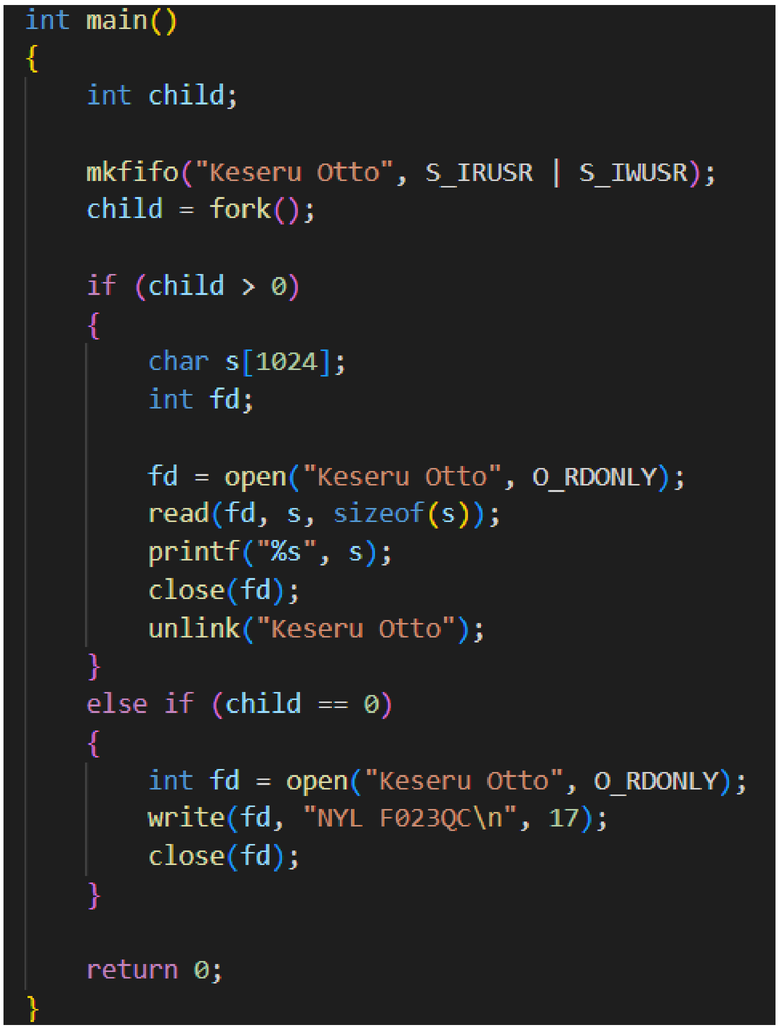
KÉSZLET-IGÉNY			
R1	**R2**	**R3**	
-4	-1	-1	p0
2	1	0	p1
-3	3	2	p2
3	2	1	p3
-1	0	1	p4

2. feladat



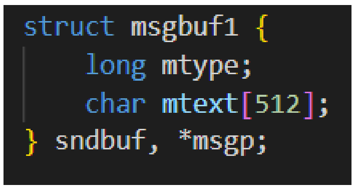
```c
int main()
{
    int fd[2];
    int child;

    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], "NYL F023QC\n", 17);
        close(fd[1]);
    }

    return 0;
```

3. feladat

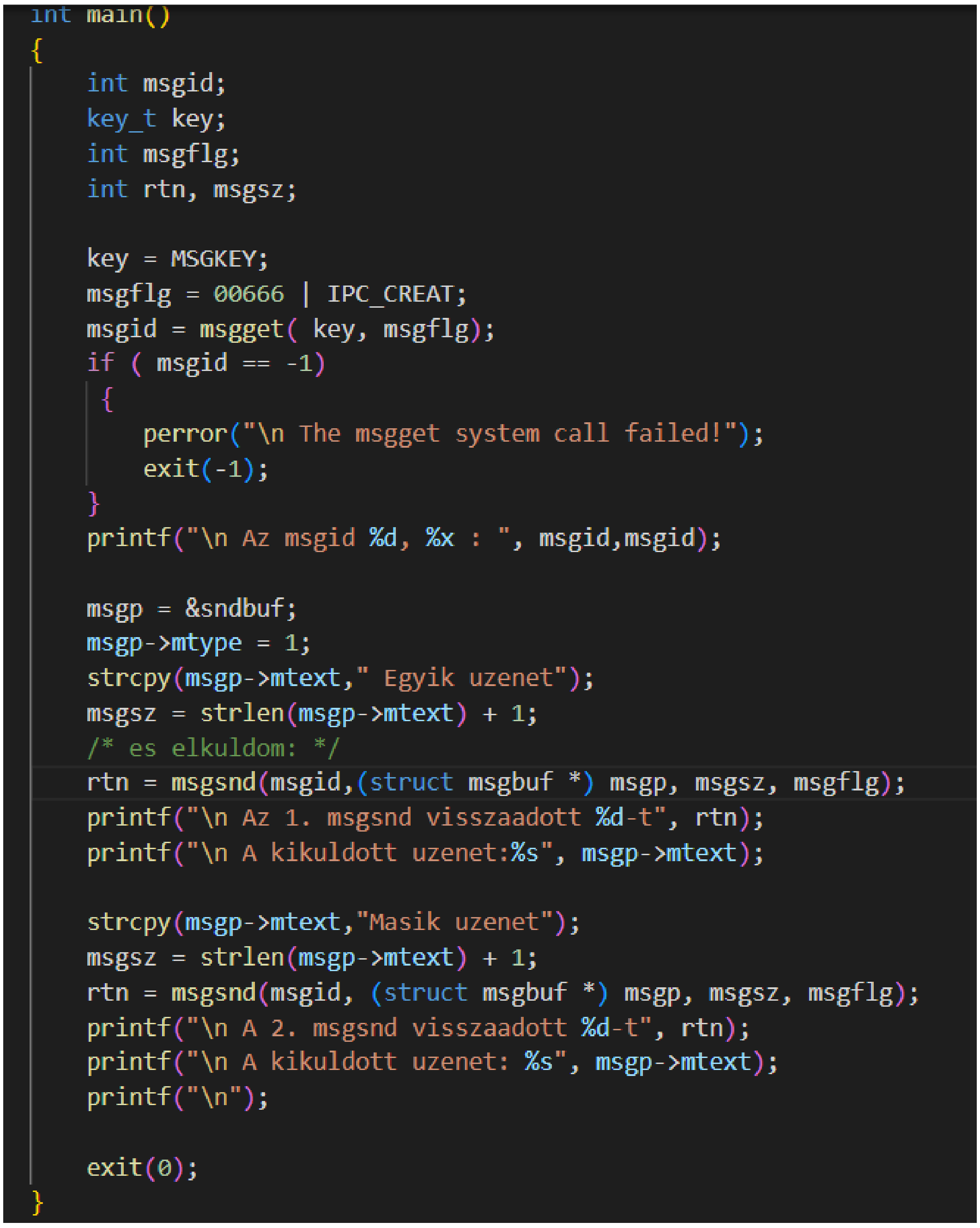


```c
int main()
{
    int child;

    mkfifo("Keseru Otto", S_IRUSR | S_IWUSR);
    child = fork();

    if (child > 0)
    {
        char s[1024];
        int fd;

        fd = open("Keseru Otto", O_RDONLY);
        read(fd, s, sizeof(s));
        printf("%s", s);
        close(fd);
        unlink("Keseru Otto");
    }
    else if (child == 0)
    {
        int fd = open("Keseru Otto", O_RDONLY);
        write(fd, "NYL F023QC\n", 17);
        close(fd);
    }

    return 0;
}
```

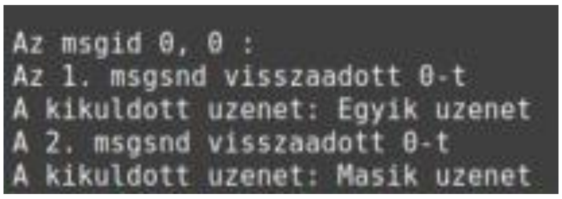
4. felada



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

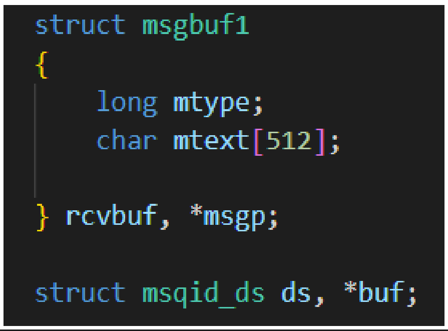


```c
int main()
{
    int msgid;
    key_t key;
    int msgflg;
    int rtn, msgsz;

    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;
    msgid = msgget( key, msgflg);
    if ( msgid == -1)
     {
        perror("\n The msgget system call failed!");
        exit(-1);
    }
    printf("\n Az msgid %d, %x : ", msgid,msgid);

    msgp = &sndbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext," Egyik uzenet");
    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 uzenet:%s", msgp->mtext);

    strcpy(msgp->mtext,"Masik uzenet");
    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 uzenet: %s", msgp->mtext);
    printf("\n");

    exit(0);
}
```

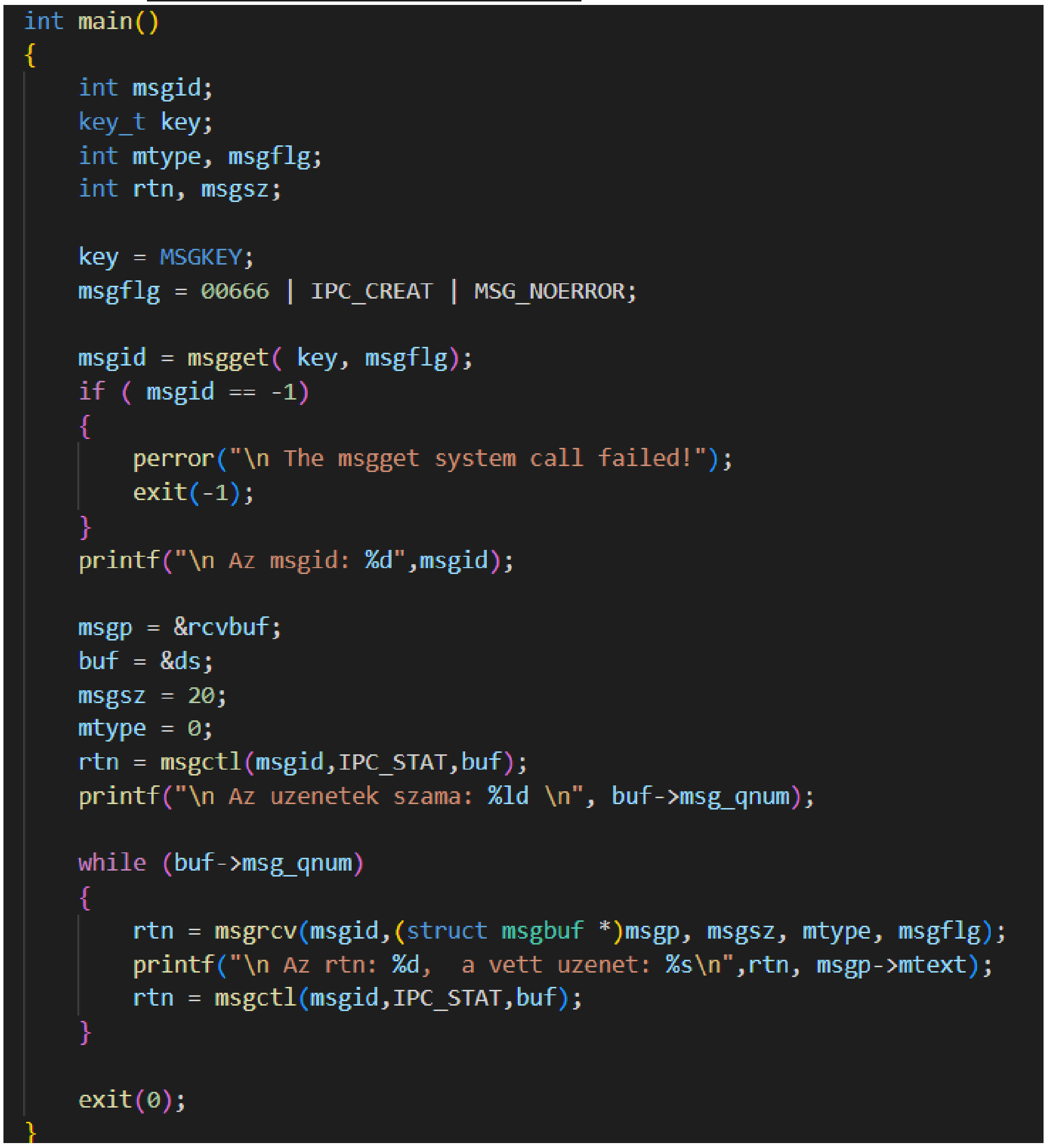


```
Az msgid 0, 0 :
Az 1. msgsnd visszaadott 0-t
A kikuldott uzenet: Egyik uzenet
A 2. msgsnd visszaadott 0-t
A kikuldott uzenet: Masik uzenet
```

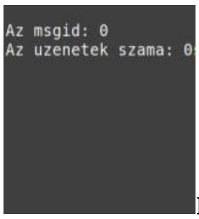
msgcreate.c



```c
struct msgbuf1
{
    long mtype;
    char mtext[512];

} rcvbuf, *msgp;

struct msqid_ds ds, *buf;
```

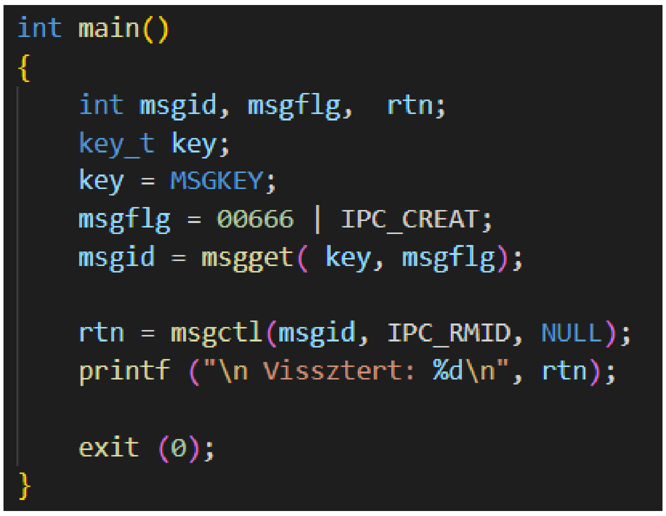


```c
int main()
{
    int msgid;
    key_t key;
    int mtype, msgflg;
    int rtn, msgsz;

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

    msgid = msgget( key, msgflg);
    if ( msgid == -1)
    {
        perror("\n The msgget system call failed!");
        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 uzenetek szama: %ld \n", buf->msg_qnum);

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

    exit(0);
}
```

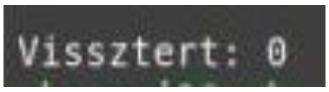


```
Az msgid: 0
Az uzenetek szama: 0
```

msgrcv.c



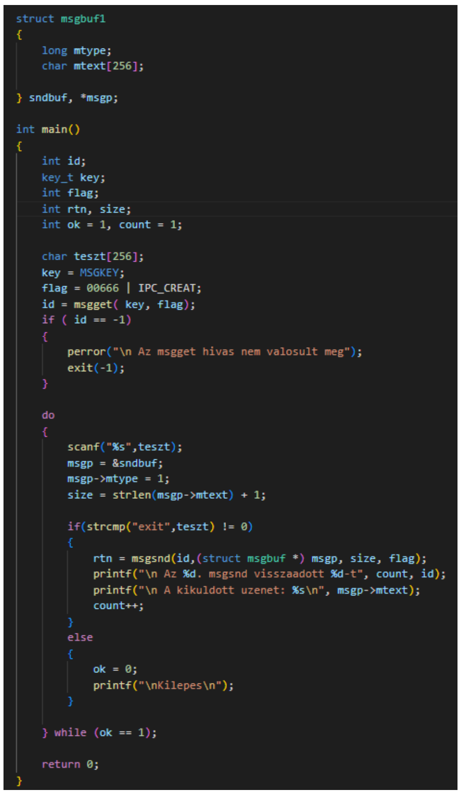
```c
int main()
{
    int msgid, msgflg,  rtn;
    key_t key;
    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;
    msgid = msgget( key, msgflg);

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

    exit (0);
}
```



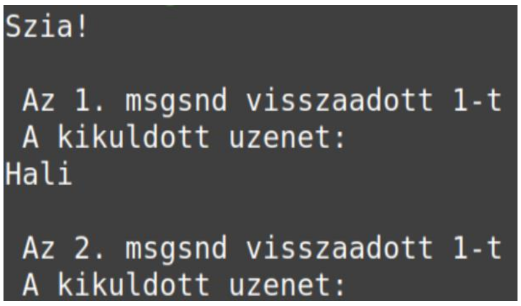
```
Visszter: 0
```

msgctl.c

4a.

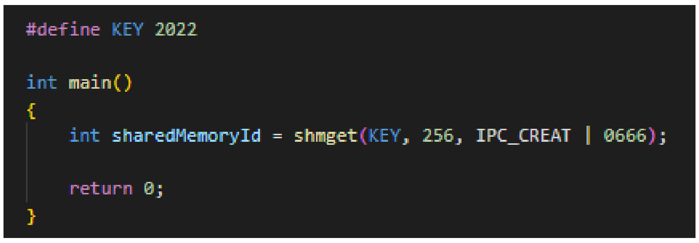


```c
struct msgbuf1
{
    long mtype;
    char mtext[256];

} 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;
        size = strlen(msgp->mtext) + 1;

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

    } while (ok == 1);

    return 0;
}
```



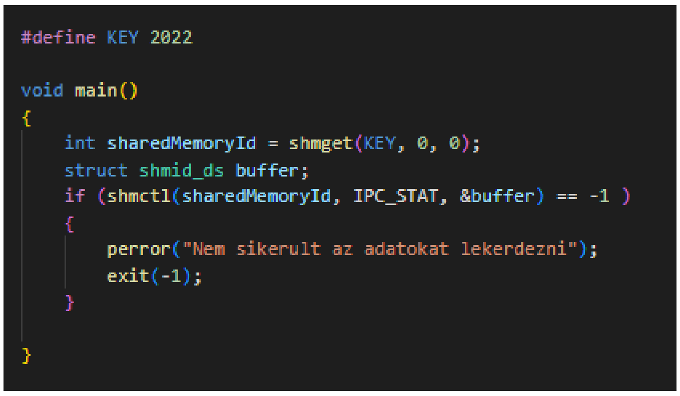
```
Szia!

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

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

5.



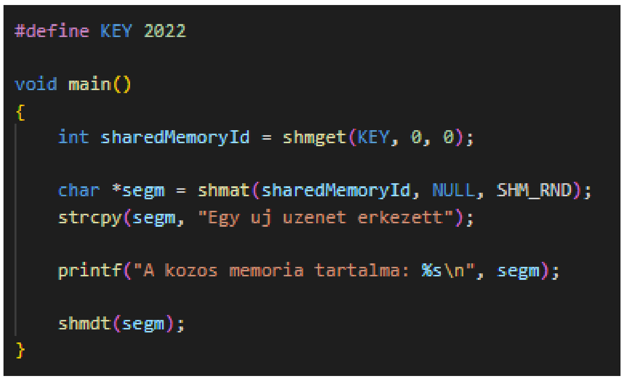
```c
#define KEY 2022

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

    return 0;
}
```

shmcreate.c

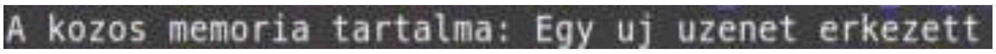


```c
#define KEY 2022

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



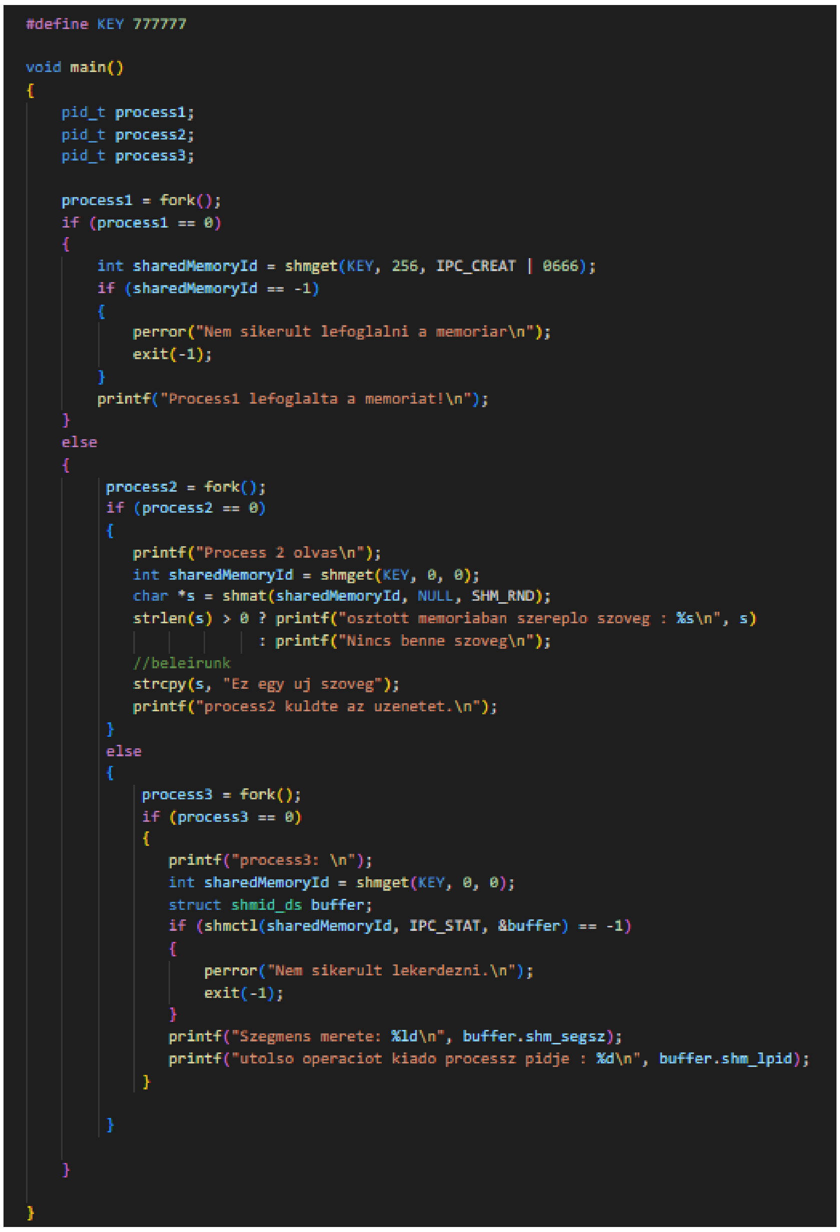
```c
#define KEY 2022

void main()
{
    int sharedMemoryId = shmget(KEY, 0, 0);

    char *segm = shmat(sharedMemoryId, NULL, SHM_RND);
    strcpy(segm, "Egy uj uzenet erkezett");

    printf("A kozos memoria tartalma: %s\n", segm);

    shmdt(segm);
}
```



```
A kozos memoria tartalma: Egy uj uzenet erkezett
```

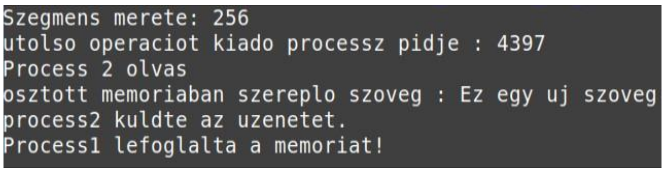
shmop.c

5a.



```c
#define KEY 777777

void main()
{
    pid_t process1;
    pid_t process2;
    pid_t process3;

    process1 = fork();
    if (process1 == 0)
    {
        int sharedMemoryId = shmget(KEY, 256, IPC_CREAT | 0666);
        if (sharedMemoryId == -1)
        {
            perror("Nem sikerult lefoglalni a memoriar\n");
            exit(-1);
        }
        printf("Process1 lefoglalta a memoriat!\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 kuldte 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:



```
Szegmens merete: 256
utolso operaciot kiado processz pidje : 4397
Process 2 olvas
osztott memoriaban szereplo szoveg : Ez egy uj szoveg
process2 kuldte az uzenetet.
Process1 lefoglalta a memoriat!
```