

Operációs rendszerek BSc

10. Gyak.

2022. 04. 13.

Készítette:

Berecz Antónia Bsc

Programtervező informatikus

LV8GJH

Miskolc, 2022

2. 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



```
File: LV86JH_unnamed.c
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main() {
    int pipefd[2];
    pid_t cpid;
    char buf;
    char szoveg[32];
    int ret;

    if (pipe(pipefd) == -1) {
        perror("pipe");
        exit(-1);
    }

    printf("fd1: %d, fd2: %d\n", getpid(), pipefd[0], pipefd[1]);

    cpid = fork();
    if (cpid == -1) {
        perror("fork");
        exit(-1);
    }

    if (cpid == 0) { // gyerek
        printf("fd: <-- gyerek PID %d\n", getpid());
        close(pipefd[0]);

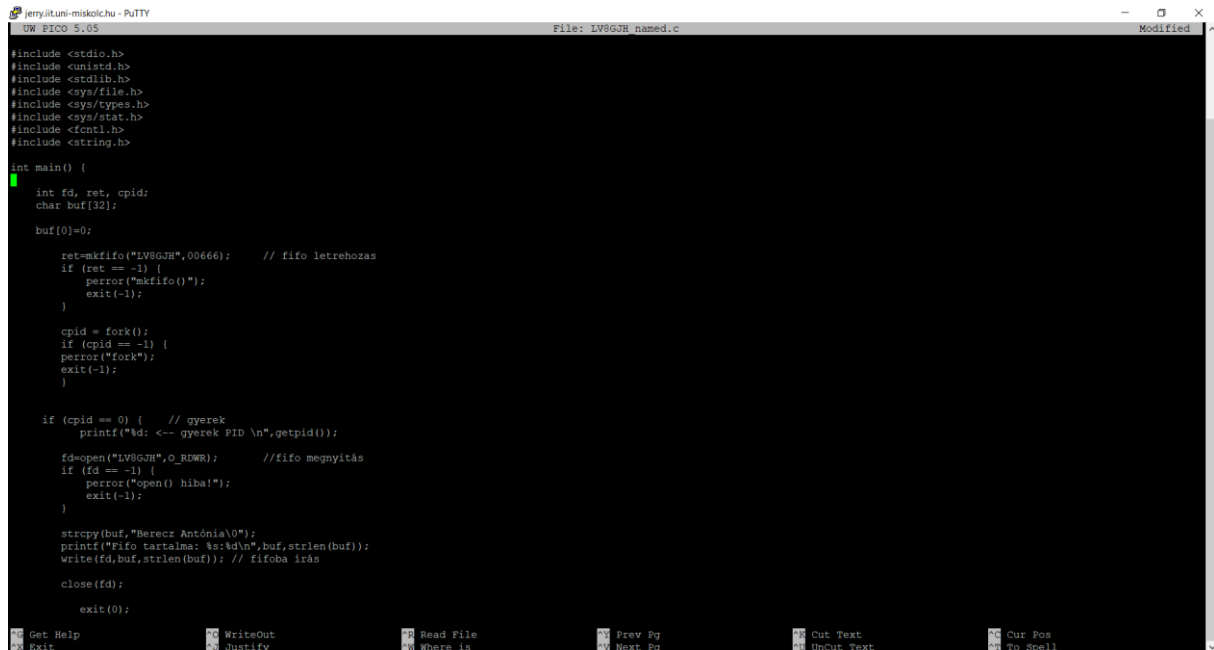
        strcpy(szoveg, "Berecz Antonia LV86JH\0");
        printf("fd: kuldom a szoveget: %s\n", getpid(), szoveg);

        write(pipefd[1], szoveg, strlen(szoveg));

        close(pipefd[1]);
        exit(0);
    } else { // szulo
        printf("fd: <-- szulo PID %d\n", getpid());
        close(pipefd[1]);

        Read 57 lines
        Get Help      WriteOut      Read File      Read 57 lines  Cut Text      Cur Pos
        Exit          Justify       Where is       Next Pg       UnCut Text    To Spell
```

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.: Keszérű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre. Mentés: neptunkod_named.c



```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main() {
    int fd, ret, cpid;
    char buf[32];

    buf[0]=0;

    ret=mknfif("LV8GJH",00666); // fifo létrehozás
    if (ret == -1) {
        perror("mknfif()");
        exit(-1);
    }

    cpid = fork();
    if (cpid == -1) {
        perror("fork");
        exit(-1);
    }

    if (cpid == 0) { // gyerek
        printf("Gyermek PID: %d\n",getpid());

        fd=open("LV8GJH",O_RDWR); //fifo megnyitás
        if (fd == -1) {
            perror("open() hiba!");
            exit(-1);
        }

        strcpy(buf,"Berez Antal");
        printf("Fifo tartalma: %s\n",buf,strlen(buf));
        write(fd,buf,strlen(buf)); // fífoba írás

        close(fd);

        exit(0);
    }

    fd=open("LV8GJH",O_RDONLY);
    if (fd == -1) {
        perror("open() hiba!");
        exit(-1);
    }

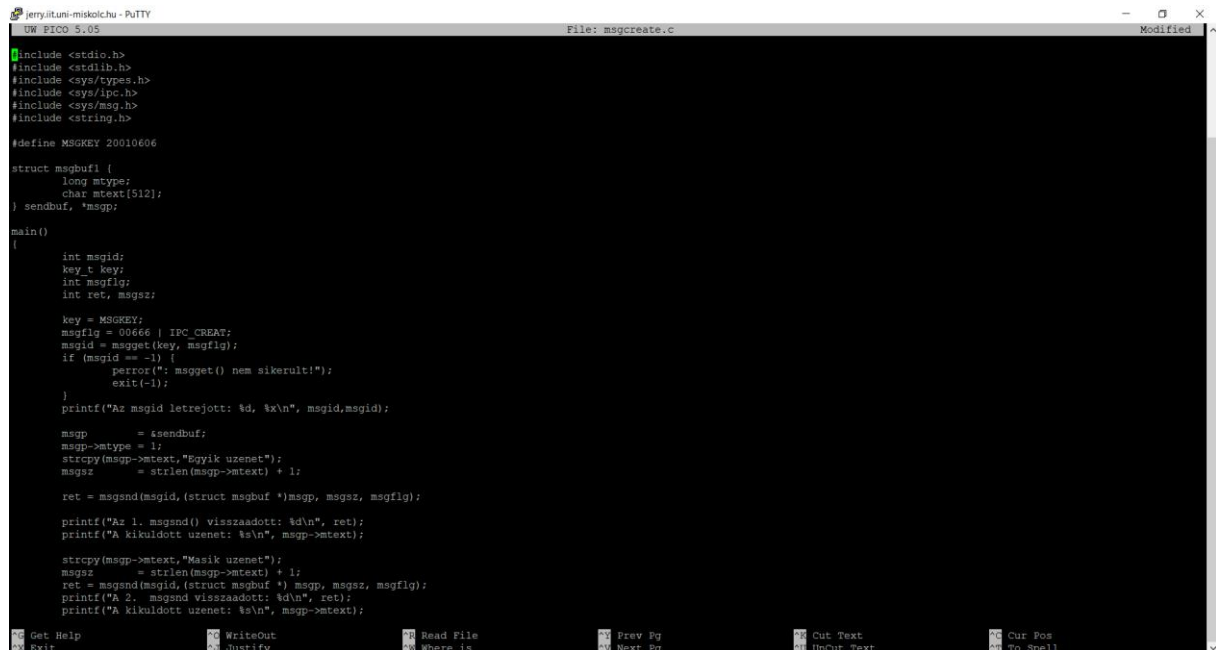
    ret=read(fd,buf,32);
    if (ret == -1) {
        perror("read()");
        exit(-1);
    }

    printf("Fifo tartalma: %s\n",buf,strlen(buf));
    close(fd);
    exit(0);
}
```

4. 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.



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

#define MSGKEY 20010606

struct msgbuf {
    long mtype;
    char mtext[512];
} sendbuf, *msgp;

main()
{
    int msgid;
    key_t key;
    int msgflg;
    int ret, msgsz;

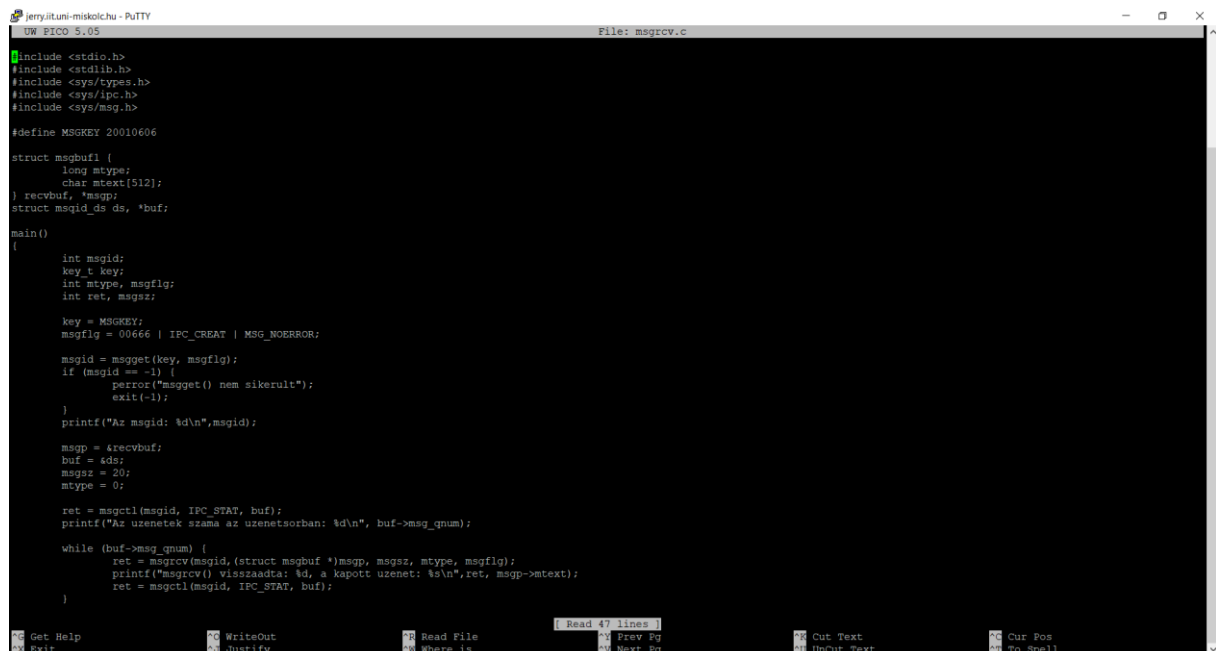
    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;
    msgid = msgget(key, msgflg);
    if (msgid == -1) {
        perror("msgget() nem sikerult!");
        exit(-1);
    }
    printf("Az msgid létrejött: %d, %x\n", msgid, msgid);

    msgp = &sendbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext, "Egyik üzenet");
    msgsz = strlen(msgp->mtext) + 1;

    ret = msgsnd(msgid, (struct msgbuf *)msgp, msgsz, msgflg);

    printf("Az 1. msgsnd() visszaadott: %d\n", ret);
    printf("A kiküldött üzenet: %s\n", msgp->mtext);

    strcpy(msgp->mtext, "Másik üzenet");
    msgsz = strlen(msgp->mtext) + 1;
    ret = msgsnd(msgid, (struct msgbuf *)msgp, msgsz, msgflg);
    printf("A 2. msgsnd visszaadott: %d\n", ret);
    printf("A kiküldött üzenet: %s\n", msgp->mtext);
}
```



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

#define MSGKEY 20010606

struct msgbuf {
    long mtype;
    char mtext[512];
} rcvbuf, *msgp;
struct msgid_ds ds, *buf;

main()
{
    int msgid;
    key_t key;
    int mtype, msgflg;
    int ret, msgsz;

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

    msgid = msgget(key, msgflg);
    if (msgid == -1) {
        perror("msgget() nem sikerult!");
        exit(-1);
    }
    printf("Az msgid: %d\n", msgid);

    msgp = &rcvbuf;
    buf = &ds;
    msgsz = 20;
    mtype = 0;

    ret = msgctl(msgid, IPC_STAT, buf);
    printf("Az üzenetek száma az üzenetsorban: %d\n", buf->msg_qnum);

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

```
UW PICO 5.05 File: msgctl.c

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

#define MSGKEY 20010606

main()
{
    int msgid, msgflg, ret;
    key_t key;

    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;

    msgid = msgget(key, msgflg);

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

    exit(0);
}
```

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

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

```
File: gyak10_4.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

#define MSGKEY 20010606

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

int main()
{
    int msgid;
    key_t key;
    int msgflg;
    int ret, msgsz;

    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT;
    msgid = msgget(key, msgflg);
    char msg[512];

    if (msgid == -1) {
        perror("msgget() nem sikerült!");
        exit(-1);
    }
    printf("Az msgid létrejött: %d, %d\n", msgid, msgid);

    do {
        printf("Adj meg a szöveget!\n");
        scanf("%[^\n]", msg);
        while (getchar() != '\n');

        if (strcmp("exit", msg) == 0) { exit(0); }
        else {
            msgp = &sendbuf;
            msgp->mtype = 1;
            strcpy(msgp->mtext, msg);

            ret = msgsnd(msgid, msgp, msgsz, 0);
            if (ret == -1) {
                perror("msgsnd() nem sikerült!");
                exit(-1);
            }
            printf("Üzenet elküldve!\n");
        }
    } while (1);
}
```

```
File: gyak10_4_2.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MSGKEY 20010606

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

struct msgid_ds ds, *buf;

int main()
{
    int msgid;
    key_t key;
    int msgflg;
    int ret, msgsz;
    int msg = 0;

    key = MSGKEY;
    msgflg = 00666 | IPC_CREAT | MSG_NOERROR;
    msgid = msgget(key, msgflg);
    if (msgid == -1) {
        perror("msgget() nem sikerült!");
        exit(-1);
    }
    printf("Az msgid: %d\n", msgid);

    do {
        printf("\n\n");
        printf("Mit szeretne választani?!\n");
        printf("1 - Üzenetek darabszámának kiírása!\n");
        printf("2 - 1 üzenet kiolvasása!\n");
        printf("3 - Összes üzenet kiolvasása!\n");
        printf("4 - Üzenetsor megszüntetése!\n");
        printf("5 - Kilépés!\n");
        scanf("%d", &msg);

        if (msg == 1) {
            printf("Üzenetek darabszámának kiírása!\n");
            printf("Összes üzenet darabszáma: %d\n", msgsnd(msgid, &ds, 0, 0));
        }
        else if (msg == 2) {
            printf("1 üzenet kiolvasása!\n");
            ret = msgrcv(msgid, msgp, 512, 1, 0);
            if (ret == -1) {
                perror("msgrcv() nem sikerült!");
                exit(-1);
            }
            printf("Kiolvasott üzenet: %s\n", msgp->mtext);
        }
        else if (msg == 3) {
            printf("Összes üzenet kiolvasása!\n");
            ret = msgrcv(msgid, msgp, 512, 0, 0);
            if (ret == -1) {
                perror("msgrcv() nem sikerült!");
                exit(-1);
            }
            printf("Kiolvasott üzenet: %s\n", msgp->mtext);
        }
        else if (msg == 4) {
            printf("Üzenetsor megszüntetése!\n");
            ret = msgctl(msgid, IPC_RMID, 0);
            if (ret == -1) {
                perror("msgctl() nem sikerült!");
                exit(-1);
            }
            printf("Üzenetsor megszüntetve!\n");
        }
        else if (msg == 5) {
            printf("Kilépés!\n");
            exit(0);
        }
    } while (1);
}
```

5. feladat

Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzetét - 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 shmctl-del azonosít osztott memória szegmenst. Ezután a segm nevű pointerváltozót használva a processz virtuális címtartományába kapcsolja (attach) a szegmenst (shmat() rendszerhívás). Olvassa, írja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

```
File: shmcreate.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

int 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! Készítsuk el!\n");
        shmflg = 00666 | IPC_CREAT;
        if ((shmid=shmget(key, size, shmflg)) < 0) {
            perror("Az shmget() system-call sikertelen!\n");
            exit(-1);
        }
        else printf("Van már ilyen shm szegmens!\n");
    }
    printf("Készült shm szegmens azonosítója %d: \n", shmid);
    exit(0);
}
```

```
File: shmctl.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

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

    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);
    }

    rtn = shmctl(shmid, IPC_STAT, buf);
    printf("Szegmens merete: %d\n", buf->shm_segsz);
    printf("Utolsó shmop()-os processz pid-je: %d\n", buf->shm_lpid);

    exit(0);
}
```



```
UW PICO 5.05 File: shmop.c Modified
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

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

    shmflg = 00666 | SHM_RND;
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);

    if (segm == (void *)-1) {
        perror("Sikertelen attach!\n");
        exit(-1);
    }

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

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

    shmdt(segm);
}
```

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

Mentés: gyak10_5.c

```
UW PICO 5.05 File: gyak10_5_1.c Modified
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

int 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! Készítsuk el!\n");
        shmflg = 00666 | IPC_CREAT;
        if ((shmid=shmget(key, size, shmflg)) < 0) {
            perror("Az shmget() system-call sikertelen!\n");
            exit(-1);
        }
    } else printf("Van már ilyen shm szegmens!\n");

    printf("\nAz shm szegmens azonosítója %d: \n", shmid);

    exit(0);
}
```

```
UW PICO 5.05 File: gyaki10_5_2.c Modified
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

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

    shmflg = 0666 | SHM_RND;
    segm = (struct vmi *)shmat(shmid, NULL, shmflg);

    if (segm == (void *)-1) {
        perror("Sikertelen attach!\n");
        exit(-1);
    }

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

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

    shmdt(segm);
}
```

```
UW PICO 5.05 File: gyaki10_5_3.c Modified
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHMKEY 20010606

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

    struct shm_ds {
        int shmid_ds;
        char *buf;
    };

    key = SHMKEY;
    int msg = 0;

    if ((shmid=shmget(key, size, shmflg)) < 0) {
        printf("Nincs meg szegmens! Reszitsuk el!\n");
        shmflg = 0666 | IPC_CREAT;
        if ((shmid=shmget(key, size, shmflg)) < 0) {
            perror("Az shmget() system-call sikertelen!\n");
            exit(-1);
        }
    } else printf("Van mar ilyen shm szegmens!\n");

    printf("\nAz shm szegmens azonositoja %d: \n", shmid);

    do {
        printf("\n\n");
        printf("Mit szeretne választani?!\n");
        printf("1 - Státusz lekérdezése\n");
        printf("2 - Osztott memória megszüntetése\n");
        printf("3 - Kilépés\n");

        scanf("%d", &msg);

        if(msg == 1){
            rtn = shmctl(shmid, IPC_STAT, buf);
        }
    } while(msg != 3);
}
```