# Operációs rendszerek BSc

## 10. Gyak.

2022. 04. 11.

**Készítette:**
Baráth Kristóf Bsc
Mérnökinformatikus
DQPDLY

**Miskolc, 2022**

### 1.feladat –

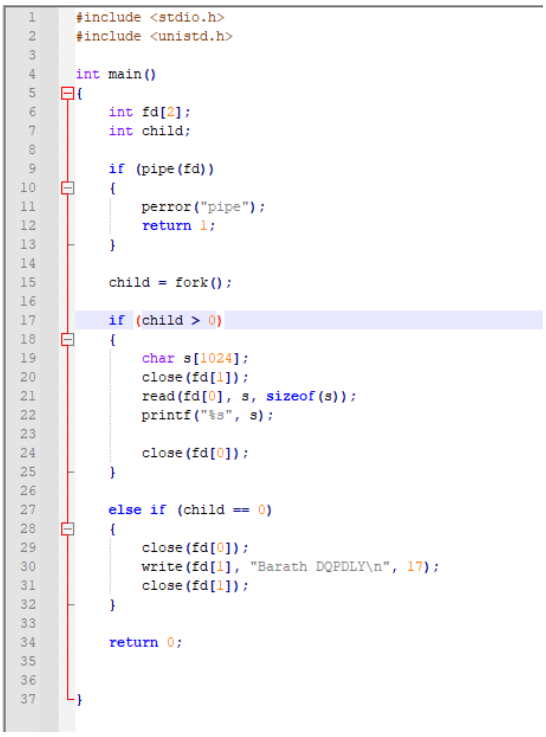
Az előadáson bemutatott mintaprogram alapján készítse el a következő feladatot.
Adott egy rendszerbe az alábbi erőforrások: R (R1: 10; R2: 5; R3: 7)
A rendszerbe 5 processz van: P0, P1, P2, P3, P4
Kérdés: Kielégíthető-e P1 (1,0,2), P4 (3,3,0) ill. P0 (0,2,0) kérése úgy, hogy biztonságos legyen, holtpontmentesség 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*?”

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | MAX. IGÉNY | | | | FOGLALÁS | | | | KIELÉGÍTETLEN IGÉNYEK | | | |
| 2 | | | | | | | | | | | | | |
| 3 | | **R1** | **R2** | **R3** | | **R1** | **R2** | **R3** | | **R1** | **R2** | **R3** | |
| 4 | p0 | 7 | 5 | 3 | | 0 | 1 | 0 | | 7 | 4 | 3 | |
| 5 | p1 | 3 | 2 | 2 | | 2 | 0 | 0 | | 1 | 2 | 2 | |
| 6 | p2 | 9 | 0 | 2 | | 3 | 0 | 2 | | 6 | 0 | 0 | |
| 7 | p3 | 2 | 2 | 2 | | 2 | 1 | 1 | | 0 | 1 | 1 | |
| 8 | p4 | 4 | 3 | 3 | | 0 | 0 | 2 | | 4 | 3 | 1 | |
| 9 | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | |
| 11 | | | | | | | | | | KÉSZLET-IGÉNY | | | |
| 12 | | | | | Foglaltak | 7 | 2 | 5 | | **R1** | **R2** | **R3** | |
| 13 | | | | | Összesen | 10 | 5 | 7 | | -4 | -1 | -1 | |
| 14 | | | | | Szabad erőforrás szám | 3 | 3 | 2 | | 2 | 1 | 0 | |
| 15 | | | | | | | | | | -3 | 3 | 2 | |
| 16 | | | | | | | | | | 3 | 2 | 1 | |
| 17 | | | | | | | | | | -1 | 0 | 1 | |
| 18 | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | |

**2. feladat** – Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetéket, 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`



```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], "Barath DQPDLY\n", 17);
        close(fd[1]);
    }

    return 0;


}
```

3. **feladat** – Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetéket (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`

```c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
    int child;

    mkfifo("Kovacs Denes", S_IRUSR | S_IWUSR);
    child = fork();

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

        fd = open("Kovacs Denes", O_RDONLY);
        read(fd, s, sizeof(s));
        printf("%s", s);
        close(fd);
        unlink("Kovacs Denes");
    }
    else if (child == 0)
    {
        int fd = open("Kovacs Denes", O_RDONLY);
        write(fd, "Barath DQPDLY\n", 17);
        close(fd);
    }

    return 0;
}
```

**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**.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSGKEY 654321L

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

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;

    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);
}
```

# msgcreate.c

```c
#include <stdio.h>
#include <stdlib.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 msqid_ds ds, *buf;



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);
}
```

**msgrcv.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSGKEY 654321L

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 Visszatert: %d\n", rtn);

    exit (0);
}
```

**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**

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
#define MSGKEY 654321L

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;
}
```

**5. Gyakorló feladat** – Először tanulmányozzák Vadász Dénes: Operációs rendszer jegyzetet - 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 címtartomanyába kapcsolja (attach) a szegmest (shmat() rendszerhívás). Olvassa, irja ezt a címtartományt, végül lekapcsolja (detach) a shmdt() rendszerhívással).

## shmcreate.c

```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

```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);
    }

}
```

```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 *segm = shmat(sharedMemoryId, NULL, SHM_RND);
    strcpy(segm, "New message");

    printf("Kozos: %s\n", segm);

    shmdt(segm);
}
```

**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)”

```c
#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;
    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");

            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);
            }

        }

    }

}
```