

Operációs rendszerek BSc

10. Gyak.

2022. 04. 11.

Készítette:

Martinák Mátyás Bsc

Programtervező informatikus

KLNSPG

Miskolc, 2022

1. feladat - 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?”

Az összes osztály -erőforrások száma: (10, 5, 7)											
Kiinduló állapot											
1. lépés				2. lépés							
MAX. igény				Foglal				MÉG			
	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
MAXr = [10, 5, 7]											
SZABAD = [10, 5, 7] - [7, 2, 5] = [3, 3, 2]											
Most megnézzük, hogy a MÉG[i] <= SZABAD feltétel igaz-e a P4 és a P0 processzekre											
P4 SZABAD = [3,3,2] + [3,3,0] = [6,6,2]											
P0 SZABAD = [6,6,2] + [0,2,0] = [6,8,2]											

Egyedül a P4-es processzre volt igaz a feltétel, hogy kevesebb erőforrást kér, mint amennyi szabad. A P0-ás processz azonban nem teljesül, így e feltételek alapján nem lesz biztonságos a rendszer.

2. feladat - 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.

```

#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], "MM KLNSPG\n", 12);
        close(fd[1]);
    }

    return 0;
}
```

A mainen belül a szülő processz létrehoz egy csővezetékét majd a gyerek processz beleír egy szöveget a vezetékekbe, és a szöveget kiírja a szülő processz a promptra.

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.

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

int main() {
    int child;

    mkfifo("Martinak Matyas", S_IRUSR | S_IWUSR);
    child = fork();

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

        fd = open("Martinak Matyas", O_RDONLY);
        read(fd, s, sizeof(s));
        printf("%s", s);
        close(fd);
        unlink("Martinak Matyas");
    } else if(child == 0) {
        int fd = open("Martinak Matyas", O_WRONLY);
        write(fd, "MM KLNSPG\n", 12);
        close(fd);
    }

    return 0;
}
```

A mainen belül létrejön egy nevesített csővezeték, majd a gyerek processz beleír egy szöveget a vezetékebe. Ugyanazok a lépések érvényesek itt is, mint a nem nevesített csővezetékénél.

4. feladat - Írjon három C nyelvű programot, ahol készít egy üzenetsort és ebbe két üzenetet tesz bele – msgcreate.c, majd olvassa ki az üzenetet - msgrev.c, majd szüntesse meg az üzenetsort (takarít) - msgctl.c.

- msgcreate.c:

```
int main()
{
    int id;
    key_t key;
    int flag;
    int rtn, size;

    key = MSGKEY;
    flag = 00666 | IPC_CREAT;
    msgid = msgget( key, msgflg);
    if ( id == -1) {
        perror("\n Az msgget hivas nem valosult meg");
        exit(-1);
    }

    printf("\n Az msgid %d, %x : ", msgid,msgid);

    msgp = &sndbuf;
    msgp->mtype = 1;
    strcpy(msgp->mtext,"Egyik uzenet");
    size = strlen(msgp->mtext) + 1;

    rtn = msgsnd(id,(struct msgbuf *) msgp, size, flag);
    printf("\n Az 1. msgsnd visszaadott %d-t", rtn);
    printf("\n A kikuldott uzenet: %s", msgp->mtext);

    strcpy(msgp->mtext,"Masik uzenet");
    size = strlen(msgp->mtext) + 1;
    rtn = msgsnd(id,(struct msgbuf *) msgp, size, flag);
    printf("\n A 2.msgsnd visszaadott %d-t", rtn);
    printf("\n Az uzenet: %s", msgp->mtext);
    printf("\n");

    exit (0);
}
```

- msgrcv.c

```
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 hivas nem valosult meg");
        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: %d",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);
}
```

- msgctl.c

```
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\n", rtn);

    exit (0);
}
```

4b. 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.

```
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;
        strcpy(msgp->mtext,teszt);
        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);
            printf("\n A kikuldott uzenet: %s\n", msgp->mtext);
            count++;
        }
        else
        {
            ok = 0;
            printf("\nKilepes\n");
        }
    } while(ok == 1);
    return 0;
}
```