Operációs rendszerek BSc

10. Gyak.

2022. 04. 13.

Készítette:
Kaló István Bsc
Mérnökinformatikus
T59MGG

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	

				KÉSZLET-IGÉNY			
Foglaltak	7	2	5	R1	R2	R3	
Összesen	10	5	7	-4	-1	-1	p0
Szabad erőforrás szám	3	3	2	2	1	0	p1
				-3	3	2	p2
				3	2	1	p3
				-1	0	1	p4

2.feladat:

```c
#include <stdio.h>
#include <unistd.h>

int main()
{

    int fd[2];
    int gyerek;


    if (pipe(fd)){
        perror("Pipe hiba!\n");
        return 1;
    }

    gyerek = fork();

    if (gyerek > 0) {
        char s[1024];
        close (fd[1]);
        read(fd[0], s, sizeof(s));
        printf ("%s", s);

        close(fd[0]);
    } else if (gyerek == 0){
            close(fd[0]);
            write(fd[1], "Kaló István T59MGG\n");
            close(fd[1]);
        }

        return 0;
}
```

3.feladat:

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

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, "KI T59MGG\n");
            close(fd);
        }

    return 0;
}
```

4.feladat:

```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
        {
```

5.feladat:

```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("Az 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);
```