JEGYZŐKÖNYV

Operációs rendszerek BSc

2022. tavasz féléves feladat

Készítette: Kriston Ádám Bsc

Program Tervező Informatikus

SYQ7E2

Miskolc, 2022

13.

Adott négy processz (A, B, C, D) a rendszerbe, induláskor a p_cpu értéke A=0, B=0, C=0, D=0. A rendszerben a P_USER = 60. Az óraütés 1 indul, a befejezés 301-ig. Induláskor a p_usrpri A=60, B=65, C=60 és D=60. Induláskor a p_nice értéke A=0, B=5, C=0 és D=0. a.) Határozza meg az ütemezést RR 301 óraütésig - táblázatba! b.) Minden óraütem esetén határozza meg a processzek sorrendjét óraütés előtt/után. c.) Igazolja a számítással a tanultak alapján

Elkészítésének lépései: Mindig a legkisebb prioritású processz fut. Minden őraütésnél a futó processz p_cpu ideje 1-gyel nő. Minden 60.dik óraütésnél a futó processz p_cpu idejét és p_uspri-jét újraszámoljuk

p_cpu = p_cpu/2

p_pri = p_user+p_cpu/2+p_nice/2

	Round Robin	A folyamat		B folyamat		C folyamat		D folyamat		Átütemezés			
Quantum	Óraütés	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	előtte	utána	P_USER:	60
	kiindulás	**60**	**0**	65	0	60	0	60	0	A	A	A nice:	0
	1		**1**		0		0		0	A	A	B nice:	5
	2		**2**		0		0		0	A	A	C nice:	0
	3		**3**		0		0		0	A	A	D nice:	0
	4		**4**		0		0		0	A	A	konst1:	2
	5		**5**		0		0		0	A	A	konst2:	2
	6		**6**		0		0		0	A	A		
1	.				0		0		0	A	A		
	10		**10**		0		0		0	A	A		
	20		**20**		0		0		0	A	A		
	30		**30**		0		0		0	A	A		
	40		**40**		0		0		0	A	A		
	50		**50**		0		0		0	A	A		
	60		**60**		0		0		0	A	C		
	61	75	30	62	0	**60**	**1**	60	0	C	C		
	62		30		0		**2**		0	C	C		
	63		30		0		**3**		0	C	C		
	64		30		0		**4**		0	C	C		
	65		30		0		**5**		0	C	C		
	66		30		0		**6**		0	C	C		
2	.		30		0				0	C	C		
	70		30		0		**10**		0	C	C		
	80		30		0		**20**		0	C	C		
	90		30		0		**30**		0	C	C		
	100		30		0		**40**		0	C	C		
	110		30		0		**50**		0	C	C		
	120		30		0		**60**		0	C	D		
	121	67	15	62	0	75	30	**60**	**1**	D	D		
	122		15		0		30		**2**	D	D		
	123		15		0		30		**3**	D	D		
	124		15		0		30		**4**	D	D		
	125		15		0		30		**5**	D	D		
	126		15		0		30		**6**	D	D		
3	.		15		0		30			D	D		
	130		15		0		30		**10**	D	D		
	140		15		0		30		**20**	D	D		
	150		15		0		30		**30**	D	D		

3	.		15		0		30			D	D
	130		15		0		30		**10**	D	D
	140		15		0		30		**20**	D	D
	150		15		0		30		**30**	D	D
	160		15		0		30		**40**	D	D
	170		15		0		30		**50**	D	D
	180		15		0		30		**60**	D	B
	181	63	7	**62**	**1**	67	15	75	30	B	B
	182		7		**2**		15		30	B	B
	183		7		**3**		15		30	B	B
	184		7		**4**		15		30	B	B
	185		7		**5**		15		30	B	B
	186		7		**6**		15		30	B	B
4	.		7				15		30	B	B
	190		7		**10**		15		30	B	B
	200		7		**20**		15		30	B	B
	210		7		**30**		15		30	B	B
	220		7		**40**		15		30	B	B
	230		7		**50**		15		30	B	B
	240		7		**60**		15		30	B	A
	241	62	4	77	30	63	7	67	15	A	A
	242		5		30		7		15	A	A
	243		6		30		7		15	A	A
	244		7		30		7		15	A	A
	245		8		30		7		15	A	A
	246		9		30		7		15	A	A
5	.				30		7		15	A	A
	250		14		30		7		15	A	A
	260		24		30		7		15	A	A
	270		34		30		7		15	A	A
	280		44		30		7		15	A	A
	290		54		30		7		15	A	A
	300		64		30		7		15	A	C
6	301	76	32	69	15	61	3	63	7	C	C

25. Irjon egy olyan C programot, amely
letrehozza, olvassa, irja es torli az osztott memoriat! A muveletet a parancssoron keresztul adja meg.
Amennyiben egy muvelet kiadasakor a kozos memoria nem letezik a program automatikusan hozza letre azt!

Az argc a beadott argumentumok száma és a argv az argumentumok. A key_t-vel létrehozok 1 kulcsot,shmget-el lekérek egy smid-et, megnyitok egy file-t,, shmat-al csatlakozunk a memóriaszegmensre. A szoveg stringhez hozzáírjuk az argumentumban megadott szöveget (az strcat() -el). A 2. argumentum (argv[1]) tartalmazza a parancsot hogy mi történjen (a parancsokat). A be parancs beolvas a memóriába, a ki kiolvassa a memóriát és a töröl törli az osztott memóriát.

```c
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{




    // Kulcs létrehozása
    key_t kulcs = ftok("shmf",65);

    // Az shmget egy azonosítót ad vissza az shmid-vel és az IPC_Creat létrehozza az osztott memóriát
    int shmid = shmget(kulcs, 1024, 0666|IPC_CREAT);

    // shmat-al csatlakozunk a memóriaszegmensre
    char* str = (char*) shmat(shmid,NULL,0);




    // beolvassuk a memóriába
    char szoveg[1024] = "";
    if(strcmp(argv[1],"be")==0)
    {

        for(int i = 2; i < argc; i++)
        {
            strcat(szoveg,argv[i]);
            strcat(szoveg," ");
        }
        strcat(str,szoveg);
        // szöveg beolvasása majd kiírása a memóriaszegmensbe, pointer bezárása

        printf("written data: %s\n", str);
    }


    //kiírjuk mi van a memóriában
    else if(strcmp(argv[1],"ki")==0)
    {
        printf("A szegmensbol olvasott adat: %s\n",str);
    }

    else if(strcmp(argv[1],"torol")==0)
    {
        // Lecsatlakozás a memóriaszegmensről
        shmdt(str);
```

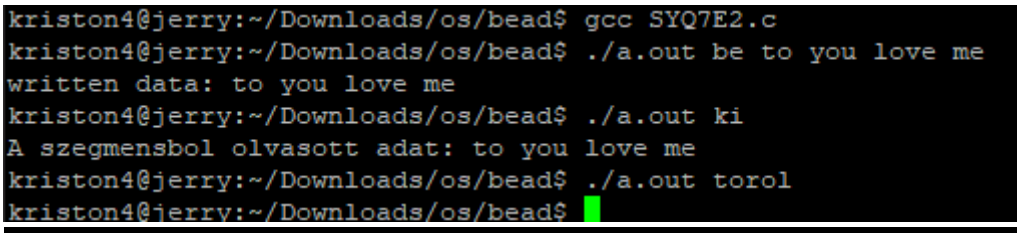
```c
        shmdt(str);

        // Memóriaszegmenst törlése
        shmctl(shmid, IPC_RMID, NULL);
    }

    else
    {
        shmdt(str);
        shmctl(shmid, IPC_RMID, NULL);
        printf("Wrong input");
        exit(1);

    }


    return 0;
}
```


kriston4@jerry:~/Downloads/os/bead$ gcc SYQ7E2.c
kriston4@jerry:~/Downloads/os/bead$ ./a.out be to you love me
written data: to you love me
kriston4@jerry:~/Downloads/os/bead$ ./a.out ki
A szegmensbol olvasott adat: to you love me
kriston4@jerry:~/Downloads/os/bead$ ./a.out torol
kriston4@jerry:~/Downloads/os/bead$