Operációs rendszerek Bsc

Féléves Beadandó

2022. 05. 09.

Készítette:

Flaskó Lilian Laura

Programtervező informatikus

GCNS8S

Miskolc, 2022.05.

Algoritmusos feladat:
17. -feladat:
Adott egy rendszerbe az összes osztály-erőforrások száma: R (R1: 10; R2: 9; R3: 12)
A rendszerbe 4 processz van: P1, P2, P3, P4. Biztonságos-e vagy nem biztonságos holtpontmentesség szempontjából a rendszer - a következő kiinduló állapot alapján?
a) Határozza meg a folyamatok által igényelt erőforrások mátrixát?
b) Határozza meg pillanatnyilag szabad erőforrások számát?
c) Igazolja az egyes processzek végrehajtásának lehetséges sorrendjét – számolással?

	r1	r2	r3
	10	9	12

Max. Igény			
	r1	r2	r3
p1	4	4	5
p2	1	4	3
p3	6	7	7
p4	3	7	10

Foglal			
	r1	r2	r3
p1	2	2	3
p2	1	2	2
p3	0	1	3
p4	2	1	2
ÖSSZESE	5	6	10
max.r	10	9	12
szabad	5	3	2

még	2	2	2	lefut	szabad:	7	5	5
	0	2	1	lefut	szabad:	8	7	7
	6	6	4	lefut	szabad:	8	8	10
	1	6	8	lefut	szabad:	10	9	12

A rendszer biztonságos

A rendszer biztonságos.

IPC feladat:
14. Feladat
Írjon C nyelvű programokat, ami hozzon létre egy osztott memória szegmenst a felhasználótól olvasson be szöveget, és ezt írja be az osztott memória területére és küldjön signalt a fogadó félnek, hogy kész az üzenet (SIGUSR1) (segítségképpen a másik program pid-je fixen beletehető a programba) a másik program pedig olvass ki az osztott memória szegmensből, de csak egy adott signal hatására (SIGUSR1) végül szüntesse meg az shm szegmenst.

Beolvasó

```c
int main(int argc, char *argv[])
{
    key_t key= 111; //shm kulcsa
    int shmid;
        char *guess_mem, *userinput;
        if(( shmid = shmget(key,1024,0666|IPC_CREAT)) < 0 ){ //osztott memória kreálása (kulcs, méret, jogosultság | creálásmód)
                perror("shmget"); // error kezelés
                exit(1);
        }

    int f,pid;


        f = open("myfifo", O_RDONLY); //fifo nyit

        read(f, &pid, sizeof(int)); //pid kiolvas
        close(f);
        unlink("myfifo"); //bezár minden
        printf("read pid: %d \n", pid);
        printf("Adja meg az üzenetet: ");

    size_t len = 0;
    ssize_t lineSize = 0;

    if ((guess_mem = shmat(shmid,NULL,0)) == (char *) -1){ //osztott mem hez csatlakozás
                perror("shmat"); //error kezelés
                exit(1);
        }
    lineSize = getline(&userinput, &len, stdin); //get user input
    strcpy(guess_mem, userinput); // inputot az osztott memóriába másoljuk
    shmdt(guess_mem); //lecsatlakozás az osztott mem
    kill(pid, SIGUSR1); //signal küld
    exit(0); //kilép
    return 0;
}
```

C ▾ Tabulátorszéle

Kiolvasó

```c
#include<stdio.h>
#include<unistd.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/wait.h>
#include<sys/shm.h>
#include<sys/ipc.h>
#include<sys/types.h>

void handle_sigusr1(int signum){ // signal lekezelő method

        key_t key= 111; //shm kulcs
        int shmid;
        char *guess_mem;
        if ((shmid = shmget(key,1024, 0666 | IPC_CREAT)) < 0){ //shm
                perror("shmget");
                exit(1);
        }
        if((guess_mem = shmat(shmid,NULL,0)) == (char *) -1){ // shm csatlakozás
                perror("shmat");
                exit(1);
        }

        printf("message: %s \n", guess_mem); // kiírja az üzenetet
        shmdt(guess_mem);
        shmctl(shmid, IPC_RMID, NULL);
        exit(0);
}

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

        int f, mypid = getpid(); // saját pid
        printf("mypid: %d\n" mypid);
```

```c
int main(int argc, char *argv[])
{

        int f, mypid = getpid(); // saját pid
        printf("mypid: %d\n",mypid);
        if(mkfifo("myfifo", 0666) < 0){ // kreál fifo
                perror("mkfifo"), exit(EXIT_FAILURE);
        }

        f=open("myfifo", O_WRONLY); // nyit fifo
        write(f, &mypid, sizeof(int)); // beír pid
        close(f);
        unlink("myfifo");

        signal(SIGUSR1, handle_sigusr1); // signal lekezelő beállítása
        while(1); //várunk a signálra
        return 0;
}
```