Operációs rendszerek BSc

11.gyak.

2021. 04.21.

Készítette:

Tóth József BProf
Üzemmérnök-informatikus alapszak
WI2GDP

Miskolc, 2021

1. **feladat -** *Adott egy rendszer (foglalási stratégiák), melyben a következő Szabad területek: 30k, 35k, 15k, 25k, 75k, 45k és Foglalási igények: 39k, 40k, 33k, 20k, 21k állnak rendelkezésre. Határozza meg változó partíció esetén a következő algoritmusok felhasználásával: first fit, next fit, best fit, worst fit a foglalási igényeknek megfelelő helyfoglalást!*

First Fit	Memória terület - szabad terület					
Foglalási igény	30	35	15	25	75	45
39	30	35	15	25	39, 36	45
40	30	35	15	25	75	40, 5
33	30	33, 2	15	25	75	45
20	20, 10	35	15	25	75	45
21	30	35	15	21, 4	75	45

Next Fit	Memória terület - szabad terület					
Foglalási igény	30	35	15	25	75	45
39	30	35	15	25	39, 36	45
40	30	35	15	25	75	40, 5
33	30	33, 2	15	25	75	45
20	30	35	15	20, 5	75	45
21	30	35	15	25	39, 21, 15	45

Best Fit	Memória terület - szabad terület					
Foglalási igény	30	35	15	25	75	45
39	30	35	15	25	75	39, 6
40	30	35	15	25	40, 35	45
33	30	33, 2	15	25	75	45
20	30	35	15	20, 5	75	45
21	21, 9	35	15	25	75	45

Worst Fit	Memória terület - szabad terület					
Foglalási igény	30	35	15	25	75	45
39	30	35	15	25	39, 36	45
40	30	35	15	25	75	40, 5
33	30	35	15	25	39, 33, 3	45
20	30	20, 15	15	25	75	45
21	21, 9	35	15	25	75	45

2. **feladat -** *A feladat megoldásához először tanulmányozza Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (6.4)., azaz Írjon C nyelvű programokat, ahol kreál/azonosít szemafor készletet, benne N szemafor-t.*
A kezdő értéket 0-ra állítja – ***semset.c,***
kérdezze le és írja ki a pillanatnyi szemafor értéket – ***semval.c***
szüntesse meg a példácskák szemafor készletét – ***semkill.c***
sembuf.sem_op=1 értékkel inkrementálja a szemafort – ***semup.c***

semset.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */

    int semid, nsems, semnum, rtn;
    int semflg;
    struct sembuf sembuf, *sop;
    union semun arg;
    int cmd;

int main()
{


    nsems = 1;  /* egyetlen semaphore a set-ben */
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror( s: " semget hiba"); exit(0);}
    else printf( format: "\n semid: %d ",semid);
    printf ( format: "\n kerem a semval erteket ");

    semnum = 0; /* 0-i semaphort azonositom */

    cmd = SETVAL;   /* allitsd be a semaphor erteket */
    scanf("%d",&arg.val);
    rtn = semctl(semid,semnum, cmd, arg); /* a semid-vel azonositott
                        set 0-ik semaphorat ! */

    printf("\n set  rtn: %d ,semval: %d ",rtn,arg.val);
    printf( format: "\n");
```

semval.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */


    int semid,nsems,rtn;
    int semflg;
    struct sembuf sembuf, *sop;
    union semun arg;
    int cmd;

int main()
{


    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror( s: " semget hiba"); exit(0);}
    else printf( format: "\n semid: %d ",semid);
    printf ( format: "\n");

    cmd = GETVAL;   /* E parancsra a semctl visszaadja a currens
                semaphor erteket. Itt az rtn-be.  */
    rtn = semctl(semid, semnum: 0, cmd, NULL);

    printf( format: "\n semval: %d ",rtn);
    printf( format: "\n");

}
```

semkill.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */


    int semid,nsems,rtn;
    int semflg;
    struct sembuf sembuf, *sop;
    union semun arg;
    int cmd;

int main()
{


    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror( s: " semget hiba"); exit(0);}
    else printf( format: "\n semid: %d ",semid);
    printf ( format: "\n");

    cmd = IPC_RMID;      /* Ez a megszuntetes parancsa */
    rtn = semctl(semid,0, cmd, arg);

    printf( format: "\n kill rtn: %d ",rtn);
    printf( format: "\n");

}
```

semup.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 123456L  /* kulcs a semget-nek; remelem, egyedi */

    int semid,nsems,rtn;
    unsigned nsops;
    int semflg;
    struct sembuf sembuf, *sop;


int main()
{



    nsems = 1;  /* Egy semaphore legyen */
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror( s: " semget hiba"); exit(0);}
    else printf( format: "\n semid: %d ",semid);
    printf ( format: "\n");

    nsops = 1;      /* Egy operacio van */
    sembuf.sem_num = 0; /* A 0-ik semaphor-ral foglakozunk */
    sembuf.sem_op = 1;  /* Inkrementaciot kerunk! */
    sembuf.sem_flg = 0666;  /* Flag beallitas */
    sop = &sembuf;      /* Igy ker1 a semop az argumentumot */
    rtn = semop(semid, sop, nsops);
        /* 0-val visszatero semop sikeres. */
    printf( format: "\n up rtn: %d ",rtn);
    printf( format: "\n");
```