JEGYZŐKÖNYV

Operációs rendszerek BSc

2021 tavasz féléves feladat

11.Gyakorlat

Készítette: **Érsek Norbert**

Neptunkód: **IIJU0Z**

A feladat leírása: 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!

A futtatás eredménye: Nem lehet lefuttatni, de egy táblázatban így néz ki a memóriafoglalás:

First fit		Next fit		Best fit		Worst fit	
Free		Free		Free		Free	
30k	20k	30k	21k	30k	21k	30k	20k
35k	33k	35k	33k	35k	33k	35k	33k
15k		15k		15k		15k	
25k	21k	25k	20k	25k	20k	25k	21k
75k	39k	75k	39k	75k	40k	75k	39k
45k	40k	45k	40k	45k	39k	45k	40k

- First fit: Az első megfelelő méretű memóriaterülethez fogja hozzárendelni a foglalási igényt.
- Next fit: Hasonló az előzőhez, viszont itt miután kiválasztotta az elsőt a First fit-nek megfelelően, mindig a következő megfelelő méretű helyre fogja foglalni.
- Best fit: A lehető legkevesebb fennmaradó memória maradjon szabadon az egyes mezőkben.
- Worst fit: A lehető legtöbb memória maradjon szabadon az egyes helyeken.

A feladat leírása: 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**.

A feladat elkészítésének lépései:

- Elsőként létrehozzuk a semset.c -t, aminek a központi parancsa a sem_init() lesz. Magának a parancsnak három argumentuma van: szemafor ID, egy egész, ami megmondja, hogy megoszthatjuk-e a szemafort más processzekkel vagy processzek szálaival, míg a harmadik az értékét állítja be.
- Elsőként szükségünk lesz egy sem_t típusú változóra. `sem_t mutex;`
- Ezután inicializáljuk a szemafort sem_init() segítségével. `sem_init(&mutex, 0, 1);`

- Következőnek elindítjuk a jelen esetben két szemafort. Két másodperccel egymás után indulnak, majd az első vár még két másodpercet, hogy induljon.

```
pthread_t t1,t2;
pthread_create(&t1,NULL,thread,NULL);
sleep(2);
pthread_create(&t2,NULL,thread,NULL);
pthread_join(t1,NULL);
pthread_join(t2,NULL);
```

- Ezt következően létrehozzuk a semval.c -t, aminek a középpontjában a sem_getvalue() lesz. Ez megmutatja a szemafor értékét.

```
sem_t sem;
sem_init(&sem, 0, 0);
int value;
sem_getvalue(&sem, &value);
printf("%d\n", value);
```

- Lényegében nagyon hasonló lesz a program az előzőhöz, viszont a legvégén létrehozunk egy value egész típusú változót, ami a következő sorban kerül felhasználásra.
- Utána jön a sem_getvalue(), ami nevéből adódóan visszaadja a szemaforunk értékét. Nullával indítottam el, ezért olyan értéket várunk. Magának a függvénynek a bemeneti argumentumai a következők:
    - Szemafor neve
    - Egész változó, amiben majd vissza fogja adni a szemafor értékét.
- Végül kiírjuk a konzolra.
- Harmadikként jön a semkill.c, ami az előzőtől annyiban fog eltérni, hogy a végén lesz egy sem_destroy() nevű függvényhívás, ami megöli a szemaforunkat.

```
sem_t sem;
sem_init(&sem, 0, 0);
int value;
sem_getvalue(&sem, &value);
printf("%d\n", value);
sem_destroy(&sem);
```

- Az új sor függvényének egyetlen bemeneti értéke van, mégpedig a szemafor neve.
- Utoljára a semup.c jön. Ez másabb az előzőknél, mégpedig mindenben.

```
int semid;
struct sembuf sb[1];
int nsops = 2;
int result;
sb[0].sem_num = 0;
sb[0].sem_op = -1;
sb[0].sem_flg = 0;
result = semop(semid, sb, nsops);
printf("%d\n", result);
```

- Létrehozunk egy tömböt, ami a semop() második argumentuma lesz, ezt pedig feltöltjük adatokkal.
- Az előbb említett függvény egész számmal tér vissza, így ezt az eredményt a result-ban tároljuk későbbi használatra.
- Ez a későbbi használat a mi esetünkben egy konzolra való kiírásban merül ki.

A futtatás eredménye:

1. **semset.c**

```
pi@raspberrypi:~/Documents/OsGyak11 $ ./semset

Entered..

Just Exiting...

Entered..

Just Exiting...
pi@raspberrypi:~/Documents/OsGyak11 $
```

2. **semval.c**

```
pi@raspberrypi:~/Documents/OsGyak11 $ ./semval
0
pi@raspberrypi:~/Documents/OsGyak11 $
```

3. **semkill.c**

```
pi@raspberrypi:~/Documents/OsGyak11 $ ./semkill
0
```

4. **semup.c**

```
pi@raspberrypi:~/Documents/OsGyak11 $ ./semup
-1
```