

Operációs rendszerek BSc

11. Gyak.

2022. 04. 27.

Készítette:

Honti Dániel BSc

Programtervező Informatikus

HR6121

Miskolc, 2022

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.

A rendszerben a memória 4 kbyte-os blokkokban kerül nyilvántartásra, ennél kisebb méretű töredék igény esetén a teljes blokk lefoglalásra kerül.

Határozza meg változó méretű partíció esetén a következő algoritmusok felhasználásával: first fit, next fit, best fit, worst fit a foglalási igényeknek megfelelő helyfoglalást – táblázatos formában (az ea. bemutatott mintafeladat alapján)!

Hasonlítsa össze, hogy a teljes szabad memóriaterület hány százaléka vész el átlagosan az egyes algoritmusok esetén! A kapott eredményeket ábrázolja oszlop diagrammal!

Magyarázza a kapott eredményeket és hogyan lehet az eredményeket javítani!

A 4 byte-os blokkok miatt a valódi foglalási igények:

Igény	Valódi foglalási igény
39	40
40	40
33	36
20	20
21	24

First fit:

		Szabad					
Igény	Foglalható	30	35	15	25	75	45
39	40	30	35	15	25	40+35	45
40	40	30	35	15	25	35	40+5
33	36	30	35	15	25	35	5
20	20	20+10	35	15	25	35	5
21	24	10	24+11	15	25	35	5

Szabad partíciók : 10, 11, 15, 25, 35, 5

Next fit:

		Szabad					
Igény	Foglalható	30	35	15	25	75	45
39	40	30	35	15	25	40+35	45
40	40	30	35	15	25	35	40+5
33	36	30	35	15	25	35	5
20	20	20+10	35	15	25	35	5
21	24	10	24+11	15	25	35	5

Szabad partíciók : 10, 11, 15, 25, 35, 5

Best fit:

		Szabad					
Igény	Foglalható	30	35	15	25	75	45
39	40	30	35	15	25	75	40+5
40	40	30	35	15	25	40+35	5
33	36	30	35	15	25	35	5
20	20	30	35	15	20+5	35	5
21	24	24+6	35	15	5	35	5

Szabad partíciók : 6, 35, 15, 5, 35, 5

Worst fit:

		Szabad					
Igény	Foglalható	30	35	15	25	75	45
39	40	30	35	15	25	40+35	45
40	40	30	35	15	25	35	40+5
33	36	30	35	15	25	35	5
20	20	30	20+15	15	25	35	5
21	24	30	15	15	25	24+11	5

Szabad partíciók : 30, 15, 15, 25, 11, 5

Tekintsük elvesző memóriának az olyan blokkokat, amelyek kisebbek, mint a legkisebb eddigi foglalásunk (20), valamint a fennmaradó blokkok azon részeit, ami a néggyel való osztási maradékaikat adja (mert 4 byte-onként tudunk foglalni).



Az eredmények javíthatók virtuális címezéssel, ki- belapozó megoldással, például LRU vagy második esélyes FIFO algoritmussal.

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 szemafor-t – semup.c A futtatás eredményét

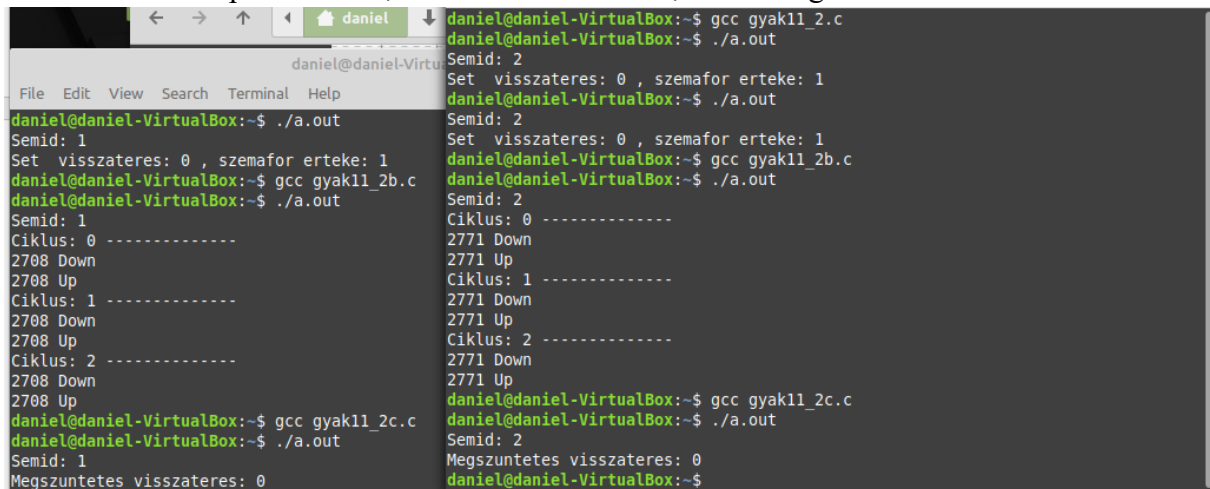
is tartalmazza a jegyzőkönyv.

```
szemafor készletet, benne N szemafor-t. A kezdő értéket 0-ra állítja – semset.c
daniel@daniel-VirtualBox:~$ gcc semset.c
daniel@daniel-VirtualBox:~$ ./a.out
Kerem a szemaforok szamat!
3
Semid: 0
Set  visszateres: 0 , 1. szemafor erteke: 0
Set  visszateres: 0 , 2. szemafor erteke: 0
Set  visszateres: 0 , 3. szemafor erteke: 0
daniel@daniel-VirtualBox:~$ gcc semval.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 0
1. szemafor erteke: 0
2. szemafor erteke: 0
3. szemafor erteke: 0
daniel@daniel-VirtualBox:~$ gcc semup.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 0
Semop visszateres: 0
Semop visszateres: 0
Semop visszateres: 0
daniel@daniel-VirtualBox:~$ gcc semval.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 0
1. szemafor erteke: 1
2. szemafor erteke: 1
3. szemafor erteke: 1
daniel@daniel-VirtualBox:~$ gcc semkill.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 0
Megszuntetes visszateres: 0
```

2a.

Írjon egy C nyelvű programot, melyben

- egyik processz létrehozza a szemafor (egyetlen elemi szemafor; inicializálja 1-re, vagy x-re, ha még nem létezik),
- másik processz használja a szemafor, belépési szakasz (down), a kritikus szakaszban alszik 2-3 sec-et, m pid-et kiír, kilépési szakasz (up), ezt ismételve 2x-3x (és a hallgató egyszerre indítson el 2-3 ilyen processzt),
- harmadik processzben, ha létezik a szemafor, akkor megszünteti”.



```
daniel@daniel-VirtualBox:~$ gcc gyak11_2.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 2
Set visszateres: 0 , szemafor erteke: 1
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 2
Set visszateres: 0 , szemafor erteke: 1
daniel@daniel-VirtualBox:~$ gcc gyak11_2b.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 2
Ciklus: 0 -----
2771 Down
2771 Up
Ciklus: 1 -----
2771 Down
2771 Up
Ciklus: 2 -----
2771 Down
2771 Up
daniel@daniel-VirtualBox:~$ gcc gyak11_2c.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 2
Megszuntetes visszateres: 0
daniel@daniel-VirtualBox:~$

daniel@daniel-VirtualBox:~$ ./a.out
Semid: 1
Set visszateres: 0 , szemafor erteke: 1
daniel@daniel-VirtualBox:~$ gcc gyak11_2b.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 1
Ciklus: 0 -----
2708 Down
2708 Up
Ciklus: 1 -----
2708 Down
2708 Up
Ciklus: 2 -----
2708 Down
2708 Up
daniel@daniel-VirtualBox:~$ gcc gyak11_2c.c
daniel@daniel-VirtualBox:~$ ./a.out
Semid: 1
Megszuntetes visszateres: 0
```