Model 05 2019-2020

1. Fie următoarea secvență

```
for (i=0; i<2; i++)
  for (j=i; j>0; j--)
    fork();
```

(a) (5p) Câte procese sunt create?

(b) (5p) Desenați arborescența proceselor create.

a. Este creat un singur proces.
b. (P0) -> (P1), unde P1 este procesul copil.

(b) (5p) Desenați arborescența proceselor create.

2. (10p) Considerați problema filosofilor și soluția propusă mai jos pentru $n \in \mathbb{N}$ filosofi așezați la masă.

```
do {
  wait(chopstick[i]);
  wait(chopstick[(i+1)%n]);
  /* ... */
  signal(chopstick[i]);
  signal(chopstick[(i+1)%n]);
} while (true);
```

(a) (5p) Demonstrați că fenomenul de *deadlock* apare pentru $n = 5$ filosofi.

(b) (5p) Dacă modificăm soluția astfel încât fiecare filosof poate ridica bețele doar dacă ambele sunt disponibile, fenomenul *deadlock* dispare dar apare fenomenul de *starvation*. Arătați de ce

apar la diferite momente de timp t

a. Dacă fiecare filozof ia întâi bățul din stânga, nimeni nu va mai putea lua bățul din dreapta deoarece toate bețele sunt luate. Cum nimeni nu poate avea 2 bețe, se ajunge la deadlock.
b. Inițial, doi filozofi(A și B) care nu sunt unul lângă altul vor începe să mănânce. Când unul din ei termină(să zicem A), atunci filozoful care este lângă B dar nu și A va mânca. Un filozof poate mânca numai dacă niciunul din vecinii lui nu mănâncă. Astfel, starvation apare atunci când în permanență cel puțin unul din vecinii lui mănâncă.

sunt disponibile, fenomenul *deadlock* dispare.

3. Fie următoarea secvență de procese care apar la diferite momente de timp t

Proces	t	CPU
P_0	0	5
P_1	0	3
P_2	2	8
P_3	4	1
P_4	6	7

(a) (5p) Cum arată diagrama Gantt rezultată în urma aplicării algoritmului Round Robin *nonpreemptive*?

(b) (5p) Dar pentru același algoritm în modul *preemptive* cu o cuantă de timp $q = 2$?

(b) (5p) Dar pentru același algoritm în modul *preemptive* cu o cuantă de timp $q = 2$? ... necesar pentru execuția corectă a proceselor pe un ... [0xbeef]).

a. Round Robin non-preemptive = FCFS

b. Round Robin preemptive cu cuanta q=2

(b) (5p) Dar pentru același algoritm în modul *preemptive* ...

4. (10p) Care este numărul minim de *frame*-uri necesar pentru execuția corectă a proceselor pe un procesor cu instrucțiuni de tipul: `instr reg, memop, memop` (ex. `add r8, [0xdead], [0xbeef]`). ... de cilindri și următoarea coadă de cereri I/O în așteptare 2000, 3000, 1200, 4, ... de citire al disk-ului se află la poziția 1000 și a

Avem 2 memops și o instrucțiune care se află în txt(registrul este în cpu). Astfel avem nevoie de 3 frame-uri, dar în cazul fericit în care ambele memop-uri sunt în același bloc, ne ajung și 2 frame-uri.

4. (10p) Care este ... procesor cu instrucțiuni de tipul: `instr reg, memop, memop` ...

5. Fie un disk cu 5000 de cilindri și următoarea coadă de cereri I/O în așteptare 2000, 3000, 1200, 4, 2018. Fiecare intrare reprezintă un cilindru, iar capul de citire al disk-ului se află la poziția 1000 și a fost înainte la poziția 314.

(a) (5p) Începând de la poziția curentă, care este ordinea și distanța totală parcursă de cap pentru a satisface toate cererile din coadă folosind algoritmul FCFS?

(b) (5p) Dar folosind algoritmul SCAN?

a. FCFS satisface cererile în ordinea din enunț, astfel parcurge (2000-1000) + (3000-2000) + (3000-1200) + (1200-4) + (2018-4) = 7010 cilindri.
b. SCAN o ia în direcția 0 -> 4999 deoarece era la 314 și acum este la 1000. Parcurge cererile 1200, 2000, 2018 și 3000, iar când ajunge la 4999 se întoarce, mergând până la 4. Ordinea finală este 1200 2000 2018 3000 4, iar distanța este de 3999 + 4995 = 8994.