**Examen dat in 2019 (Nr1+2)**

1. (a) (5p) Ce este un proces zombie/orfan?

(b) (5p) Scrieti o secventa scurta de cod si aratati cand un process devine zombie/orfan.

1. Fie urmatoarea secventa

for( i = 0; i < 3; i++){

pthread\_create();

fork();

fork();

}

for( i = 0; i < 3; i++){

fork();

pthread\_create();

fork();

}

1. (5p) Cate procese si fire de executie sunt create? La numarare, ce presupunere ati facut legata de fork()?
2. (5p) Desenati arborescenta proceselor si firelor de executie create. Etichetati cu P procesele si cu T firele de executie.
3. Considerati problema filosofilor si solutia propusa mai jos pentru n ∈N filosofi asezati la masa.

do {

wait(chopstick[i]);

wait(chopstick[(i+1)%n]);

/\* … \*/

signal(chopstick[i]);

signal(chopstick[(i+1)%n]);

} while (true);

Aceasta solutie permite aparitia fenomenului de *deadlock*.

1. (5p) Modificati solutia ridicand asimetric betisoarele; filosofii impari ridica intai betisorul din stanga, cei pari pe cel din dreapta. Aratati ca nu mai apare fenomenul.
2. (10 p) Aratati daca noua solutie satisfice cele trei proprietati: exclusivitate mutuala, progresul si timp finit de asteptare.
3. Fie o matrice A ∈ N10x10 tinuta contiguu in memorie pe linii si fie un sistem in care avem 3 frame‑uri disponibile. In acest sistem intr-o pagina incap 10 intregi, iar programele P1 si P2 de mai jos incap in totalitate intr-o pagina.

P1:

P1:

for(i = 0; i < 10; i++)

for(j = 0; j < 10; j++)

A[i][j] = 0;

P2:

for(j = 0; j < 10; j++)

for(i = 0; i < 10; i++)

A[i][j] = 0;

for(i = 0; i < 10; i++)

for(j = 0; j < 10; j++)

A[i][j] = 0;

P2:

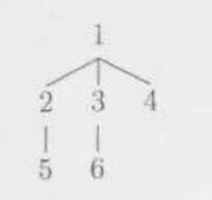
for(j = 0; j < 5; j++)

for(i = 0; i < 5; i++)

A[i][j] = 0;

1. (5p) Cum arata programul si datele repartizate pe pagini?
2. (5p) Folosind algoritmul LRU, care este programul efficient? De ce?
3. (5p) Cum arata diagramele Gannt pentru P1 si P2?

**Examen dat in 2019 (Nr4)**

1. (10p) Puteti simula o structura de directoare pe mai multe nivele cu o structura cu un singur nivel? Daca da, explicate cum ati implementat o astfel de simulare si cum se compara cu structura pe mai multe nivele. Daca nu, explicate ce va impiedica sa faceti acest lucru. Dati un exemplu.
2.  (10p) Fie urmatoarea arborescenta de procese

Scrieti o secventa de cod care reproduce aceasta structura.

1. Fie doua procese Pi si Pj care vor sa acceseze o zona critica si urmatoarea implementare pentru Pi:
2. do {
3. flag[i] = true;
4. while (flag[j]) {
5. if(turn == j) {
6. flag[i] = false;
7. while (turn == j)
8. ; /\* do nothing \*/
9. flag[i] = true;
10. }
11. }
12. /\* critical section \*/
13. turn = j;
14. flag[i] = false;
15. /\* remainder section \*/
16. } while (true);

unde procesele impart flag[2] (initializat cu false) si turn.

1. (5p) Ce se intampla daca eliminam linia 13?
2. (10p) Aratati daca solutia satisface cele trei proprietati: exclusivitatea mutuala, progres si timp finit de asteptare.
3. Fie doua matrice A ∈ N6x6 si B ∈ N4x6 tinute contiguu in memorie pe linii si fie un sistem in care avem 5 frame‑uri disponibile. In acest sistem intr-o pagina incap 6 intregi, iar programele P1 si P2 de mai jos incap fiecare separat intr-o pagina.

1 P2:

2 for(i = 0; i < 4; i++)

3 for(j = 0; j < 6; j++)

4 B[i][j] = i + j;

1 P1:

2 for(i = 0; i < 6; i++)

3 for(j = 0; j < 6; j++)

4 A[i][j] = i + 2 \* j;

Presupunem ca programele se executa concurent astfel: fiecare program sta pe procesor cat sa duca pana la capat instructiunea de la linia 4 o singura data dupa care cedeaza locul concurentului.

1. (5p) Cum arata programele si datele repartizate pe pagini?
2. (5p) Cum arata diagrama Gantt folosind algoritmul LRU cu o strategie de inlocuire a paginilor globala (in care ambele programe pot folosi toate frame-urile disponibile in sistem).
3. (5p) Cum ati aloca frame-urile pentru o strategie de inlocuire a paginilor locala (in care fiecare program primeste un numar fix de frame-uri pe care doar el le poate utiliza). De ce?

**Alt examen (Nr 2)**

1. Fie urmatoarea secventa

for( i = 0; i < 2; i++)

for( j = i; j > 0; j--)

fork();

1. (5p) Cate procese sunt create?
2. (5p) Desenati arborescenta proceselor create.
3. (10p) Considerati problema filosofilor si solutia propusa mai jos pentru n ∈N filosofi asezati la masa.

do {

wait(chopstick[i]);

wait(chopstick[(i+1)%n]);

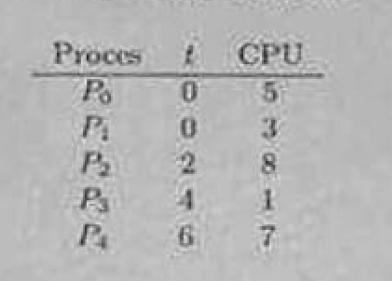
/\* … \*/

signal(chopstick[i]);

signal(chopstick[(i+1)%n]);

} while (true);

1. (5p) Demonstrati ca fenomenul de deadlock apare pentru n=5 filosofi.
2. (10p) Daca modificam solutia astfel incat fiecare filosof poate ridica betele doar daca abele sunt disponibile, fenomenul deadlock dispare dar apre fenomenul de starvation. Aratati de ce.
3. Fie urmatoarea secventa de procese care apar la diferite momente de timp t.



1. (5p) Cum arata diagrama Gantt rezultata in urma aplicarii argoritmului Round Robin nonpreemptive?
2. (5p) Dar pentru acelasi algoritm in modul preemptive cu o cuanta de timp q=2?
3. (10p) Care este numarul minim de frame-uri necesar pentru executia corecta a proceselor pe un procesor cu instructiuni de tipul: instr reg, memop, memop (ex. add r8, [0xdead], [0xbeef]).
4. Fie un disk cu 5000 de cilindri si urmatoarea coada de cereri I/O in asteptare 2000, 3000, 1200, 4, 2018. Fiecare intrare reprezinta un cilindru, iar capul de citire al disk-ului se afla la pozitia 1000 si a fost inainte la pozitia 314.
5. (5p) Incepand de la pozitia curenta, care este ordinea si distanta totala parcursa de cap pentru a satisface toate cererile din coada folosind algoritmul FCFS?
6. (5p) Dar folosind algoritmul SCAN?