SISTEME DE OPERARE
EXAMEN FINAL
5 FEBRUARIE 2024

INSTRUCTIUNI

- Va rog sa va scrieti numele pe fiecare foaie pe care o predati (inclusiv pe foile cu subiectele de examen pe care le veti returna impreuna cu celelalte foi)
- Examenul se tine fara documentatie pe masa si fara acces la echipamente electronice (telefon mobil, tableta, ceas inteligent, etc)
- Aveti 120 minute pentru a termina examenul. Abordati examenul cu inteligenta: daca nu stiti pe moment raspunsul la o intrebare, treceti la urmatoarea si reveniti mai tarziu; dati raspunsuri concise si evitati sa pierdeti vremea furninzand detalii irelevante sau care nu sunt solicitate.
- Primele 10 minute sunt destinate citirii subiectelor. In tot acest timp nu aveti voie sa va atingeti de ustensilele de scris. Nerespectarea acestei conditii se pedepseste cu iesirea din examen si pierderea punctajului aferent.

SUBIECTE

1. (6 pcte) Remote Procedure Call (RPC).

(a) (2 pcte) Ce este un apel de procedura la distanta (RPC)? Enumerati doua diferente fata de un apel de procedura obisnuit.

(b) (2 pcte) Ce este o operatie idempotenta? Dati un exemplu de operatie idempotenta si un exemplu de operatie care nu este idempotenta.

(d) (2 pcte) Ce este semantica RPC *exactly once* (exact o singura data)? Explicati folosind un exemplu de ce este dificil sa se implementeze aceasta semantica.

2. (20 pcte) Procese si thread-uri.

(a) (2 pcte) Enumerati principalele stari in care se poate afla un proces in executie si explicati in ce conditii se face tranzitia intre aceste stari? (O diagrama adnotata corespunzator e suficienta).

(b) (4 pcte) Care sunt cerintele necesare unei solutii corecte de partajare a resurselor?

(c) (2 pcte) Enumerati doua avantaje ale thread-urilor kernel fata de procese.

(d) (2 pcte) Enumerati doua avantaje ale thread-urilor user fata de cele kernel.

(e) (2 pcte) Enumerati doua dezavantaje ale thread-urilor user.

(f) (2 pcte) Ce este inversiunea de prioritati? Dati un exemplu.

(g) (2 pcte) Prezentati pe scurt o solutie pentru rezolvarea problemei inversiunii de prioritati folosind Lottery Scheduling (planificarea de tip loterie).

(h) (4 pcte) O functie multithreaded pe care trebuie sa o scrieti are doua sectiuni critice: *increment()* incrementeaza un intreg si *insert()* insereaza o structura de date intr-un buffer partajat de lungime fixa. Primitivele de sincronizare disponibile pentru implementarea sectiunilor critice sunt *spin-lock*-uri si *semafoare*. Pentru fiecare dintre cele doua sectiuni critice, specificati:

- ce primitive de sincronizare ati utiliza pentru implementare in fiecare caz

- justificati alegerea facuta anterior pentru fiecare caz de sectiune critica in parte

3. (12 pcte) Sincronizarea in sisteme multiprocesor.

(a) (3 pcte) Considerati un sistem multiprocesor simetric care foloseste un protocol *write-invalidate*. De ce este TATAS (Test-And-Test-And-Set) mai potrivita pentru implementarea sectiunilor critice decat TAS (Test-And-Set) ?

(b) (2 pcte) Ce se schimba daca se foloseste TATAS cu un protocol *write- update*?

(c) (2 pcte) Dati doua exemple de dezavantaje ale sincronizarii cu instructiuni hardware atomice pe care sincronizarea *wait-free* (*lock-free*) le rezolva?

(d) (2 pcte) Ce este sincronizarea *wait-free* (*lock-free*) ?

(e) (3 pcte) Dati un exemplu de instructiuni hardware care permit implementarea unei memorii tranzactionale restranse la dimensiunea unui cuvant si explicati pe scurt semantica lor.

4. (12 pcte) Sisteme de stocare a datelor si fisiere.

(a) (3 pcte) Un sistem de fisiere Unix System V (s5fs) foloseste blocuri de dimensiune 4 KB si pointeri catre blocuri pe 4 octeti. Cat de mare este cel mai mare fisier pe care il poate stoca acest sistem de fisiere? (Indicatie: folositi structura inode-ului).

(b) (3 pcte) Ce este un buffer cache? Dar un page cache? Ce problema apare cand ambele cache-uri co-exista in sistem si care ar fi o solutie posibila?

(c) (3 pcte) Care sunt componentele principale ale timpului de acces la date pentru discuri clasice organizate pe cilindri, piste si sectoare? Detaliati raspunsul pe scurt.

(d) (3 pcte) Care componenta dintre cele de la punctul anterior poate fi optimizata folosind algoritmi de planificare de disc? Dati doua exemple de asemenea algoritmi pe scurt.

5. (12 pcte) Gestiunea memoriei

Un PC are 512 MB RAM, un microprocesor pe 47 de biti, pagini de 16 KB si foloseste PTE-uri (intrari in tabela de pagini) de 8 octeti. Un program care ruleaza pe acest computer are 10 KB de text si date incarcate la baza spatiului de adrese si 5 KB de stiva la capatul superior al spatiului de adrese.

(a) (4 pcte) Cat de mare este o tabela liniara de pagini care mapeaza programul in RAM?

(b) (4 pcte) Trasati o diagrama (o schema) care mapeaza acest program in memoria RAM folosind o tabela de pagini pe 3 niveluri care foloseste un numar egal de biti de adresa pentru fiecare nivel. Cat spatiu ocupa aceasta tabela de pagini?

(c) (4 pcte) Descrieti o posibila implementare pentru tabela de pagini de la punctul (a) si calculati spatiul de memorie fizica necesar stocarii acestei tabele.

6. (8 pcte) Planificarea proceselor.

(a) (4 pcte) Data fiind o colectie de procese cu caracteristicile de mai jos, trasati o diagrama Gantt pentru o politica de planificare de tip Shortest Remaining Time First cu prioritati si calculati timpul mediu de asteptare pentru aceasta politica. Timpii de sosire/rulare sunt

exprimati in *ms*, valorile mici exprima prioritati mari. Politica de planificare Shortest Remaining Time First se aplica proceselor disponibile pentru rulare din aceeasi clasa de prioritate, in ordinea prioritatilor. Cand nu mai exista procese disponibile pentru rulare intr-o anumita clasa de prioritate, se aplica politica de planificare la urmatoarea clasa de prioritate in ordinea prioritatii, samd.

Proces	Timp de sosire	Timp de rulare	Prioritate
P1	0	6	2
P2	2	4	1
P3	2	2	3
P4	4	1	1
P5	0	5	2

(b) (4 pcte) Cum arata diagrama Gantt si care este timpul mediu de asteptare pentru o politica de tip Round-Robin cu prioritati care foloseste o cuanta de timp de 1ms? La fel ca mai sus politica de planificare se aplica la nivel de clase de prioritate.

7. (10 pcte) Paginarea la cerere (demand paging).

Se da urmatorul sir de referinte la pagini intr-un sistem cu 3 frame-uri de memorie:

3 4 1 2 3 1 4 2 5 3 6 4 1 5 2 6

(a) (2 pcte) Cate page-fault-uri inregistreaza rularea algoritmului FIFO de inlocuire a paginilor care trebuie scoase din memorie atunci cand o noua pagina trebuie incarcata si nu mai exista niciun frame liber in memorie?

(b) (4 pcte) Cum se schimba numarul de page-fault-uri daca se mareste numarul de frame-uri de memorie la 4? Ce observati? Explicati situatia.

(c) (4 pcte) Calculati numarul de page-fault-uri generat de algoritmul LRU de inlocuire a paginilor pentru dimensiuni ale memoriei de 3 respectiv 4 frame-uri. Explicati rezultatul.

8. (20 pcte) Sincronizarea proceselor/thread-urilor.

In simularea unui triaj, locomotive si vagoane sunt folosite pentru a forma trenuri de catre un controlor. Controlorul e modelat de un thread, fiecare locomotiva si respectiv fiecare vagon sunt modelate de cate un thread. Pana cand sunt necesare, locomotivele asteapta intr-o coada de locomotive, iar vagoanele asteapta intr-o coada de vagoane. Pentru a forma un tren, controlorul da voie unui vagon si unei locomotive sa intre in triaj. Dupa ce o locomotiva si un vagon se afla in triaj, trenului nou format i se permite sa plece. Cand trenul pleaca, locomotiva trebuie sa fie inaintea vagonului.

Pentru a modela operarea fiecarui tip de thread, adaugati la codul de mai jos:

- declaratiile de semafoare necesare si valorile lor initiale;
- cod semaforizat pentru rutina *controlor()* executata de catre thread-ul controlorului;
- cod semaforizat pentru rutina *locomotiva()* executata de catre thread-urile locomotiva;

- cod semaforizat pentru rutina *vagon()* executata de catre thread-urile vagon;

```
sem_t _____ = ___;  // semafoare atatea cate sunt necesare plus valorile lor initiale

controlor()
{
        while(true) {
                // lasa o locomotiva sa intre in triaj
                // lasa un vagon sa intre in triaj
                // asteapta pana cand o locomotiva si un vagon sunt in triaj
                formeaza_tren();
                // lasa locomotiva sa plece
                // asteapta pana cand a plecat locomotiva
                // lasa vagonul sa plece
                // asteapta pana cand a plecat vagonul
                trenul_pleaca();
        }
}
locomotiva()
{
        // asteapta cat e necesar
        // anunta intrarea in triaj
        locomotiva_in_triaj();
        // asteapta plecarea
        // anunta plecarea
        locomotiva_pleaca();
        exit();
}
vagon()
{
        // asteapta cat e necesar
        // anunta intarea in triaj
        vagon_in_triaj();
        // asteapta plecarea
        // anunta plecarea
        vagonul_pleaca();
        exit();
}
```