

SISTEME DE OPERARE  
EXAMEN RESTANTA  
20 FEBRUARIE 2024

INSTRUCTIUNI

- Va rog sa va scrieti numele pe fiecare foale 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 100 minute la dispozitie 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. (14 pcte) Gestiunea memoriei.

- (a) (2 pcte) Descrieti mecanismul generic de suport hardware pentru protectia memoriei (o diagrama corespunzator adnotata este suficienta).
- (b) (2 pcte) Ce este swapping-ul si la ce ajuta?
- (c) (3 pcte) Alocarea contigua a memoriei. Dati 3 exemple de algoritmi de alocare dinamica a memoriei cu partitii variabile (gestiunea "hole"-urilor care apar in asemenea sisteme). Explicati pe scurt cum functioneaza cei trei algoritmi alesi.
- (d) (2 pcte) Ce este fragmentarea memoriei si de cate tipuri este? Explicati pe scurt.
- (e) (5 pcte) Ce este segmentarea memoriei? Enumerati trei diferente intre paginare si segmentare.

2. (10 pcte) (14 pcte) Memoria virtuala.

- (a) (2 pcte) Cum functioneaza principial paginarea la cerere (*demang paging*) ?
- (b) (2 pcte) Enumerati doua avantaje ale paginarii la cerere (ale memoriei virtuale in general)?
- (c) (2 pcte) Ce este tehnica de *Copy-On-Write* (COW) si la ce foloseste?
- (d) (4 pcte) Cate tipuri de algoritmi de inlocuire a paginilor exista in functie de setul de frame-uri din care se alege victima (frame-ul care contine pagina ce va fi evacuata din memorie)? Explicati pe scurt consecintele alegerii fiecaruia dintre algoritmi.

3. (20 pcte) Sincronizarea proceselor.

- (a) (2 pcte) Care sunt limitările alternanței stricte ca metoda software de sincronizare între două procese?



- (b) (3 pte) Ce probleme ale alternanței stricte rezolvă soluția lui Peterson? Raman și probleme nereșolvare?
- (c) (3 pte) Ce este un monitor? Care sunt avantajele folosirii monitorilor?
- (d) (4 pte) Ce problema apare când un proces aflat într-un monitor apelează operația **signal** asupra unei variabile condiție? Care sunt soluțiile posibile?
- (e) (4 pte) Care sunt condițiile necesare producerii deadlock-ului?
- (f) (1 pct) Câte dintre condițiile de mai sus trebuie îndeplinite pentru a se putea produce un deadlock?
- (g) (3 pte) Ce sunt lock-urile adaptive? Explicați.

4. (12 pte) Sisteme de stocare a datelor și fișiere.

- (a) (3 pte) Ce este Virtual Filesystem Switch (VFS) și la ce folosește? Cum funcționează dpdv principal? (Indicație: explicați de pilda cum funcționează citirea unui fișier în VFS).
- (b) (3 pte) Enumerați două caracteristici ale sistemului de fișiere care influențează performanța algoritmilor de planificare de disc. Explicați ce se întâmplă.
- (c) (6 pte) Un disc are 40 de cilindri. Fie o secvență de citiri care implică cilindrii 11, 1, 36, 16, 34, 9 și 12, în această ordine. Comparați timpul de căutare (*seek time*) al următorilor algoritmi: FCFS, C-SCAN, LOOK. Puteti presupune ca la început capul de citire a discului se afla în dreptul primului cilindru din lista de cereri.

5. (12 pte) Planificarea proceselor.

O aplicație multimedia conține trei thread-uri de timp real:  $T_v$  afișează stream-ul video iar  $T_s$  și  $T_d$  reprezintă canalele stâng și respectiv drept ale sursei de sunet stereo. Caracteristicile de timp în milisecunde ale acestor thread-uri sunt:

Thread	Perioada	Timp de calcul
$T_v$	24	10
$T_s$	8	2
$T_d$	8	2

- (a) (4 pte) Se pot planifica aceste thread-uri conform algoritmului Rate Monotonic (considerați ca  $\ln 2 = 0,69$ )? Explicați răspunsul.
- (b) (2 pte) Se pot planifica aceste thread-uri folosind algoritmul Earliest Deadline First? Explicați răspunsul.
- (d) (6 pte) Fie două thread-uri de timp real cu parametrii de timp exprimați în milisecunde ca în tabela de mai jos.

Thread	Timp de sosire	Deadline	Timp de calcul
$T_1$	0	17	10
$T_2$	2	10	5

Presupunand o cuanta de timp de 1 milisecunda, trasati diagrame de timp pentru executia acestor thread-uri conform algoritmilor de planificare *Earliest Deadline First*, respectiv *Least Slack First*.

6. (20 pcte) Functia `int ticket(int *sequencer)` incrementeaza intregul referit de *sequencer* si intoarce valoarea incrementata.

```
int ticket(int *sequencer)    {  
    *sequencer += 1;  
    return *sequencer;  
}
```

Folosind primitivele de sincronizare de la punctele (a) si (b), scrieti versiuni ale functiei `C ticket` care incrementeaza *atomic \*sequencer* (valoarea referentiata de adresa de memorie care constituie operandul functiei) atunci cand functia ruleaza pe un sistem *multiprocesor*. In fiecare caz, definiti semantica primitivei (felul in care opereaza) si specificati ce valoare intoarce.

(a) (4 pcte) *Compare-and-Swap*: `int CAS(int *location, int compare, int update)`;

(b) (4 pcte) *Load-Linked*: `int LL(int *location)`

*Store-Conditional*: `int SC(int value, int *location)`;

(c) (8 pcte) Sistemul de operare FOS (*Funny OS*) ofera programatorului un singur tip de primitiva de sincronizare: *event counters* (contoare de evenimente). Un contor de evenimente este o variabila de tip intreg *E nedescrescatoare* cu valoare initiala zero asupra careia se pot executa urmatoarele operatii:

**READ**(E) – citeste valoarea lui E

**ADVANCE**(E) – incrementeaza *atomic* valoarea lui E

**AWAIT**(E, v) – blocheaza procesul apelant pana cand E ajunge la valoarea v

Folositi functia `ticket` cu semantica de mai sus si contoare de evenimente pentru a implementa semafoare, adica furnizati pseudocod pentru structura de date folosita de semafor si pentru operatiile **down** si **up**.

(d) (4 pcte) Explicati cum functioneaza semafoarele definite la punctul anterior atunci cand sunt folosite pentru a proteja accesul la o sectiune critica. Mai exact, simulati operatiile **down** si **up** efectuate de doua procese care vor acces in sectiunea critica si evidentiati valorile structurii de date care reprezinta semaforul pe durata executiei sincronizate a celor doua procese.

7. (12 pcte) Alocarea memoriei.

Kernelul unui sistem de operare foloseste un alocator de memorie de tip *Buddy System* pentru a gestiona o memorie de 4 KB. Kernelul efectueaza urmatoarea secventa de operatii de alocare/deallocare de blocuri de memorie, dimensiunea blocurilor alocate fiind furnizata ca parametru al apelului si reprezentata in **octeti**:



```
A = allocate(73);  
B = allocate(1536);  
C = allocate(512);  
D = allocate(1536);  
E = allocate(512);  
free(C);  
free(A);  
free(B);  
free(E);  
free(D);
```

La fiecare operatie de alocare/dealocare desenati harta memoriei, reprezentand hasurat blocurile de memorie alocata la momentul respectiv si nehasurat blocurile libere (evidentiati in acelasi timp, separat si listele cu blocuri libere disponibile si dimensiunea lor). Harta memoriei trebuie adnotata cu adresele de inceput si sfarsit ale **tuturor** blocurilor, deopotriva a celor alocate cat si a celor nealocate (libere). Ori de cate ori exista mai multe blocuri libere de aceeaasi dimensiune, se alege blocul cu adresa cea mai mica. Explicati pe scurt regulile de alocare si dealocare care se aplica precum si consecintele lor. Cum afecteaza operatiile de alocare/dealocare de mai sus fragmentarea memoriei sistemului? Comentati raspunsul folosind exemple concrete alese dintre operatiile de mai sus.