**SUBIECT RESTANTA SEPTEMBRIE 2023**

**EX1 Structura sistemelor de operare**

(a) (2 pcte) Dati doua exemple de organizare a structurii sistemelor de operare.

(b) (2 pcte) Explicati pe scurt diferentele dintre cele doua exemple de la punctul anterior.

### (a) Exemple de Organizare a Structurii Sistemelor de Operare:

1. \*\*Monolitice:\*\* (SEPARA KERNELUL DE SYSTEM PROGRAMS)

- Într-un sistem de operare monolitic, întregul nucleu (kernel) este implementat și rulează în spațiul de adresă al sistemului.

- Toate serviciile și driverele sunt implementate în cadrul aceluiași kernel.

- Exemplu: Linux.

2. \*\*Microkernel:\*\*

- Într-un sistem de operare bazat pe microkernel, funcționalitățile de bază ale sistemului, precum planificarea proceselor și gestionarea memoriei, sunt implementate în nucleul mic.

- Servicii suplimentare, cum ar fi driverele de dispozitiv și sistemele de fișiere, sunt implementate în afara nucleului.

- Exemplu: QNX.

### (b) Diferențele dintre Exemplele de mai Sus:

1. \*\*Monolitice:\*\*

- \*\*Pro:\*\* Performanță ridicată, deoarece toate apelurile la sistem sunt în interiorul aceluiași spațiu de adresă.

- \*\*Contra:\*\* O modificare sau un bug într-un serviciu poate afecta întregul sistem. Complexitatea și dimensiunea kernelului pot crește. Program mare, concurent, dificil de inteles si de intretinut

2. \*\*Microkernel:\*\* (MUTA CAT MAI MULTE ACTIUNI DIN KERNEL IN SPATIUL UTILIZATORULUI)

- \*\*Pro:\*\* Modularitate crescută și izolare a serviciilor. O schimbare într-un serviciu nu afectează direct restul sistemului.

- \*\*Contra:\*\* Performanța poate fi mai scăzută decât în sistemele monolitice din cauza comunicării interproces cu nucleul mic. Dezvoltarea și întreținerea pot fi mai complexe.

**EX2. Exceptii si intreruperi**

(a) (2 pcte) Ce sunt exceptiile? Dati doua exemple.

(b) (1 pct) Care e rolul intreruperilor in operarea sistemelor de calcul?

(c) (2 pcte) Ce face sistemul de operare (kernelul) ca raspuns la o intrerupere hardware? Descrieti mecanismul de tratare a intreruperilor.

### (a) Exceptii:

1. \*\*Definiție:\*\*

- Exceptiile sunt evenimente neașteptate sau situații anormale care apar în timpul execuției unui program și care pot afecta fluxul normal de execuție.

2. \*\*Exemple:\*\*

- \*\*Excepții aritmetice:\*\* Generarea unei excepții atunci când se încearcă împărțirea la zero.

- \*\*Excepții de acces la memorie:\*\* Tentativa de a accesa o zonă de memorie nepermisă poate duce la o excepție.

Sau mai sunt:

=>interne CPU

-> fault (se intample inaintea executiei instructiunii care o genereaza)

-> trap (se intampla dupa executia instructiunii care o genereaza)

-> abort (se intampla in timpul executiei instructiunii care o genereaza)

=>externe CPU (i.e. intreruperi hardware)

### (b) Rolul Intreruperilor:

- Intreruperile reprezintă mecanisme prin care dispozitivele hardware sau alte componente ale sistemului pot comunica cu CPU-ul și pot întrerupe execuția normală a unui program pentru a gestiona evenimente specifice.

### (c) Tratarea Intreruperilor de către Sistemul de Operare:

1. \*\*Generarea Intreruperii:\*\*

- O componentă hardware (cum ar fi un dispozitiv de intrare/ieșire sau un ceas) generează o intrerupere.

2. \*\*Întreruperea este Reținută:\*\* (KERNELUL SALVEAZA CONTEXTUL CPU-ULUI)

- Procesorul întrerupe execuția normală și salvează starea curentă a procesului, inclusiv conținutul registrelor, într-o structură de date cunoscută sub numele de "context de întrerupere".

3. \*\*Salt la Rutina de Tratare:\*\* (KERNELUL DA CONTROLUL UNUI HANDLER DE TRATARE A INTRERUPERII)

- Sistemul de operare (kernelul) determină tipul de intrerupere și sare la o rutină de tratare corespunzătoare.

- Această rutină de tratare este o secțiune de cod specializată pentru gestionarea evenimentului specific asociat intreruperii.

4. \*\*Execuția Rutinei de Tratare:\*\*

- Rutina de tratare a intreruperii este executată pentru a gestiona evenimentul asociat.

- Sistemul de operare poate efectua operații specifice, precum gestionarea evenimentelor de intrare/ieșire, actualizarea ceasului de sistem sau gestionarea excepțiilor hardware.

5. \*\*Restaurarea Stării și Continuarea Execuției:\*\*

- După ce rutina de tratare a fost executată, sistemul de operare restaurează starea anterioară a procesorului din contextul de întrerupere.

- Execuția normală a programului este reluată de la punctul în care a fost întreruptă.

**EX3. SINCRONIZAREA PROCESELOR**

(a) (3 pcte) Ce este un monitor? Care sunt avantajele folosirii monitoarelor?

(b) (3 pcte) Ce sunt variabilele conditie si cum se folosesc?

(c) (3pcte) Ce este un semafor si la ce se foloseste?

### (a) Monitoare:

1. \*\*Definiție:\*\*

- Un monitor este o construcție de programare care facilitează sincronizarea și gestionarea accesului la resurse partajate în cadrul unui program concurent.

2. \*\*Avantaje ale Monitoarelor:\*\*

- \*\*Simplicitate:\*\* Monitoarele oferă o abordare simplificată și structurată pentru gestionarea resurselor partajate în programe concurente.

- \*\*Abstractizare:\*\* Ascund detaliile implementării și furnizează o interfață clară pentru programatori.

- \*\*Sincronizare Automată:\*\* Monitoarele asigură automat sincronizarea și evită conflictele pentru resursele partajate.

### (b) Variabile Condiție:

1. \*\*Definiție:\*\*

- Variabilele condiție sunt utilizate în cadrul monitoarelor pentru a permite proceselor să aștepte condiții specifice sau evenimente să se întâmple.

2. \*\*Utilizare:\*\*

- O variabilă condiție este asociată cu o anumită condiție logică în cadrul programului.

- Procesele așteaptă pe o variabilă condiție pentru a intra într-o stare de așteptare până când condiția este îndeplinită.

- Procesele care schimbă starea sistemului pot să semnaleze sau să trezească alte procese care așteaptă pe aceeași variabilă condiție.

### (c) Semafor:

1. \*\*Definiție:\*\*

- Un semafor este o structură de date utilizată pentru a asigura sincronizarea între procese sau pentru a controla accesul la resurse partajate.

2. \*\*Utilizare:\*\*

- Semafoarele pot fi utilizate pentru a limita accesul simultan la resursele partajate, precum zonele critice ale codului.

- Un semafor are două operații principale: `P` (proberen, "a încerca") și `V` (verhogen, "a crește").

- Operația `P`(DOWN) decrementează valoarea semaforului și blochează procesul dacă valoarea devine negativă, controlând astfel accesul la resurse.

- Operația `V`(UP) incrementează valoarea semaforului și poate debloca un proces blocat în operația `P`.

### Observații:

- Monitoarele, variabilele condiție și semafoarele sunt instrumente importante pentru gestionarea concurenței în programele care rulează pe mai multe fire de execuție sau procese. Ele ajută la evitarea condițiilor de cursă și asigură accesul ordonat și sigur la resurse partajate.

**EX4. Deadlock.**

(a) (4 pcte) Care sunt conditiile necesare producerii deadlock-ului?

(b) (1 pct) Cate dintre conditiile de mai sus trebuie indeplinite pentru a se putea produce un deadlock?

### (a) Condițiile Necesare pentru Producerii Deadlock-ului:

1. \*\*Mutual Exclusion (Excluderea Mutuală):\*\*

- Cel puțin unul dintre resursele în cauză trebuie să fie folosit exclusiv de către un singur proces la un moment dat.

2. \*\*Hold and Wait (Așteptare și Deținere):\*\*

- Cel puțin un proces trebuie să dețină o resursă și să aștepte să obțină altele, ținând astfel resursele deja deținute.

3. \*\*No Preemption (Nicio Întrerupere):\*\*

- Resursele nu pot fi luate forțat de la un proces și eliberate pentru a fi utilizate de către altele; ele trebuie eliberate voluntar.

4. \*\*Circular Wait (Așteptare Circulară):\*\*

- Un lanț circular de procese, fiecare așteptând resursa detinută de procesul următor în lanț.

### (b) Condiții Necesare pentru a se Produc Deadlock:

- Toate cele patru condiții de mai sus trebuie să fie îndeplinite simultan pentru a avea un deadlock. Dacă oricare dintre aceste condiții nu este îndeplinită, atunci un deadlock nu poate apărea.

**EX5. Planificarea proceselor.**

(a) (2 pcte) Ce este inversiunea de prioritati?

(b) (2pcte) Dati un exemplu de inversiune de prioritati.

(c) (2pcte) Prezentati pe scurt o solutie pentru rezolvarea problemei inversiunii de prioritati.

### (a) Inversiunea de Priorități:

1. \*\*Definiție:\*\*

- Inversiunea de priorități apare atunci când un proces cu o prioritate mare este blocat sau așteaptă eliberarea unei resurse, iar în acest timp un alt proces cu o prioritate mai mică deține resursa și blochează astfel progresul procesului cu prioritate mai mare.

### (b) Exemplu de Inversiune de Priorități:

- Considerăm trei procese:

1. \*\*Procesul A (Prioritate Mare):\*\* Are nevoie de o resursă.

2. \*\*Procesul B (Prioritate Medie):\*\* Deține resursa și o folosește.

3. \*\*Procesul C (Prioritate Mică):\*\* Așteaptă să obțină resursa pe care o deține procesul B.

- Inversiunea de priorități apare atunci când procesul A cu prioritate mare este blocat deoarece procesul B cu prioritate medie deține resursa, iar procesul C cu prioritate mică așteaptă.

### (c) Soluție pentru Rezolvarea Inversiunii de Priorități:

- \*\*Protocolul Priorității Cu Învăluire (Priority Inheritance Protocol):\*\*

- Atunci când un proces cu prioritate mare așteaptă să obțină o resursă deținută de un proces cu prioritate mai mică, prioritatea procesului cu prioritate mai mică este temporar crescută la nivelul procesului cu prioritate mai mare.

- Acest protocol asigură că resursa este eliberată cât mai repede posibil pentru a permite progresul procesului cu prioritate mare.

- După eliberarea resursei, prioritatea procesului cu prioritate mai mică este revenită la nivelul inițial.

### Observații:

- Inversiunea de priorități poate afecta performanța sistemului, iar utilizarea protocolului priorității cu învăluire este o soluție comună pentru a gestiona această problemă.

**EX7. Gestiunea memoriei.**

(a) (2 pcte) Descrieti mecanismul generic de suport hardware pentru protectia memoriei

(b) (2 pcte) Ce este swapping ul si la ce ajuta?

(c) (3 pte) Dati 3 exemple de algoritmi de alocare dinamica a memoriei cu partitii fixe (gestiunea "hole-urilor care apar in asemenea sisteme). Explicati pe scur (1 fraza) cum functioneaza cei trei algoritmi alesi.

(d) (2 pcte) Ce este fragmentarea memoriei si de cate tipuri este? Explicati pe scurt.

(e) (3 pcte) Cum se face transformarea adresei logice in adresa fizica in sistemele de gestiune a memoriei cu paginare? (E suficient sa explicati mecanismul de suport hardware cu ajutorul unei scheme).

(f) (2 pcte) Rezolva paginarea vreuna dintre problemele fragmentarii memoriei? Explicati.

### (a) Mecanismul Generic de Suport Hardware pentru Protecția Memoriei:

- Mecanismul de suport hardware pentru protecția memoriei include:

- Registre de bază și limite asociate fiecărui proces sau modul.

- Adresa fizică a unei locații de memorie este verificată pentru a se asigura că este cuprinsă între regiunea de bază și regiunea de bază plus limita.

### (b) Swapping:

- \*\*Definiție:\*\*

- Swapping-ul este o tehnică prin care porțiuni ale memoriei unui proces sunt mutate între memoria principală și spațiul de swap pe disc, pentru a face loc altor porțiuni de memorie.

- \*\*Utilitate:\*\*

- Ajută la gestiunea eficientă a memoriei și permite executarea proceselor care depășesc capacitatea memoriei RAM disponibile prin transfer temporar între RAM și spațiul de swap.

### (c) Algoritmi de Alocare Dinamică a Memoriei cu Partiții Fixe:

1. \*\*First Fit (Primul Potrivit):\*\*

- Alloca primul spațiu liber care este suficient de mare pentru cererea de memorie.

2. \*\*Best Fit (Cel Mai Potrivit):\*\*

- Alloca spațiul liber cel mai mic care este suficient de mare pentru cererea de memorie.

3. \*\*Worst Fit (Cel Mai Rău Potrivit):\*\*

- Alloca spațiul liber cel mai mare care este suficient de mare pentru cererea de memorie.

### (d) Fragmentarea Memoriei:

- \*\*Definiție:\*\*

- Fragmentarea memoriei este fenomenul în care spațiul de memorie liber este fragmentat în bucăți mici, dificultând alocarea eficientă a resurselor.

- \*\*Tipuri:\*\*

1. \*\*Fragmentare Internă:\*\*

- Spațiu pierdut în interiorul unui bloc alocat, dar care nu este folosit de către proces.

2. \*\*Fragmentare Externă:\*\*

- Spațiu liber fragmentat în bucăți mici și răspândite în întreaga memorie.

### (e) Transformarea Adresei Logice în Adresa Fizică în Sistemele de Gestiune a Memoriei cu Paginare:

- \*\*Mecanism:\*\*

- Adresa logică este împărțită în două părți: numărul de pagină și offset-ul în cadrul paginii.

- Numărul de pagină este utilizat pentru a accesa tabela de pagini pentru a obține numărul de cadru de pagină corespunzător.

- Adresa fizică este compusă din numărul de cadru de pagină și offset-ul, furnizând adresa completă.

### (f) Paginarea și Fragmentarea Memoriei:

- \*\*Rezolvare:\*\*

- Paginarea poate contribui la reducerea fragmentării, deoarece spațiul de memorie este împărțit în pagini fixe, ușor de gestionat.

- Fragmentarea internă este redusă, deoarece o pagină este alocată doar dacă toată ea este necesară.

**EX8. Memoria virtuala.**

(a) (2 pcte) Cum functioneaza paginarea la cerere (demang paging)?

(b) (2 pcte) Care sunt avantajele folosirii paginarii la cerere?

(c) (4 pcte) Ce este un page-fault si cum este tratat de catre sistemul de operare (kernel)?

(d) (2 pcte) Care este formula timpului efectiv de acces la memorie (Effective Access Time, EAT) in sisteme cu paginare la cerere, dandu-se rata p (0 <= p <= 1) a page-fault-urilor?

(e) (2 pcte) Ce este tehnica de Copy-On-Write (COW) si la ce foloseste?

(f) (2 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.

### (a) Paginarea la Cerere (Demand Paging):

- \*\*Funcționare:\*\*

- La început, doar paginile necesare pentru inițializarea unui proces sunt încărcate în memorie.

- Restul paginilor sunt încărcate în memorie doar atunci când acestea sunt accesate pentru prima dată (la cerere).

- Aceasta minimizează timpul și spațiul necesar încărcării în memorie a întregului program, accelerând începerea execuției.

### (b) Avantajele Paginării la Cerere:

1. \*\*Economie de Timp:\*\*

- Se încarcă doar paginile necesare, reducând timpul inițial de încărcare a programului.

2. \*\*Economie de Spațiu:\*\*

- Se economisește spațiu în memorie, deoarece nu toate paginile programului trebuie să fie prezente simultan în memorie.

### (c) Page-Fault și Tratarea de către Kernel:

- \*\*Page-Fault:\*\*

- Un page-fault apare atunci când o pagină este accesată și nu este în prezent în memorie.

- \*\*Tratare:\*\*

- Kernelul preia controlul și identifică locația paginii lipsă.

- Dacă pagina este pe disc, este încărcată într-un cadru liber de memorie.

- Se reia instrucția care a generat page-fault-ul.

### (d) Timpul Efectiv de Acces la Memorii (EAT) cu Paginare la Cerere:

- \*\*Formula EAT:\*\*

EAT = (1 - p) x MemAccessTime + p x (PageFaultRate x PageFaultHandlingTime + SwapPageOutTime + SwapPageInTime)

### (e) Copy-On-Write (COW):

- \*\*Definiție:\*\*

- COW este o tehnică care permite proceselor să partajeze inițial aceeași memorie fizică, dar creează copii separate ale paginilor numai atunci când una dintre ele intenționează să scrie.

- \*\*Utilitate:\*\*

- Reducerea costului de creare a copiilor paginilor și economisirea spațiului de memorie.

### (f) Algoritmi de Înlocuire a Paginilor în Funcție de Setul de Frame-uri:

1. \*\*Global Replacement (Înlocuire Globală):\*\*

- Victima este selectată din întregul set de frame-uri.

- Consecință: Unele procese pot fi dezavantajate, deoarece o pagină folosită frecvent de un proces poate fi înlocuită de către alt proces.

2. \*\*Local Replacement (Înlocuire Locală):\*\*

- Victima este selectată doar din setul de frame-uri asociat unui proces.

- Consecință: Procesele nu pot afecta negativ altele prin înlocuirea paginilor.