PROBLEMES Conceptes Bàsics
CURS 2022-23

Concurrència

1. **Problema dels Productors-Consumidors**. Suposeu que tenim una estructura de dades consistent en un buffer circular de mida limitada i un conjunt indeterminat de processos o fluxos que es classifiquen en dos tipus (amb el comportament que es descriu):

Productor	Consumidor
Fer fins alguna condició donada Si el buffer té espai lliure aleshores Produir un element Inserir-lo al Buffer sinó Esperar	Fer fins alguna condició donada Si el buffer no és buit aleshores Extreure un element del Buffer sinó Esperar

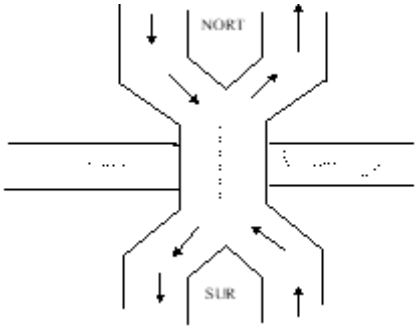
   Dissenyeu, fent servir semàfors i variables compartides, els algorismes del productor i el consumidor de tal forma que no es produeixin bloquejos i s'asseguri la integritat de les dades.

2. **El problema dels lectors-escriptors**. Suposeu que tenim una estructura de dades compartida arbitrària i un conjunt indeterminat de processos o fluxos que es classifiquen en dos tipus (amb el comportament que es descriu):

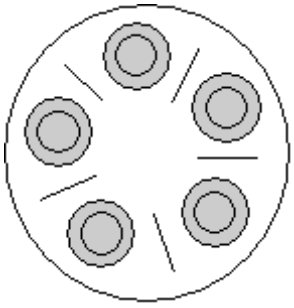
Lector	Escriptor
Fer fins alguna condició donada Si no hi ha cap escriptor escrivint Consulta els continguts de l'estructura de dades	Fer fins alguna condició donada Si no hi ha cap lector llegint, ni cap escriptor escrivint Modifica els continguts de l'estructura de dades

   Dissenyeu, fent servir semàfors i variables compartides, els algorismes del lector i l'escriptor de tal forma que no es produeixin bloquejos i s'asseguri la integritat de les dades.

3. L'ajuntament d'un poblet va decidir construir un pont per apropar les dues parts del poble que havia a cada costat del riu (nort/sur). Com que la obra era molt costosa, varen decidir un pont per vianants amb un únic sentit (tal i com s'observa en el dibuix). Com solucionaries l'accés al pont utilitzant semàfors i tenint en compte que: el vianants del sur sols poden creuar el pont si no existeix cap vianant del nort creuant-lo i el contrari. Ara bé, si existeix algun vianant del sur en el pont poden anar passant pel pont tants vianants com es vulgui i el contrari també.



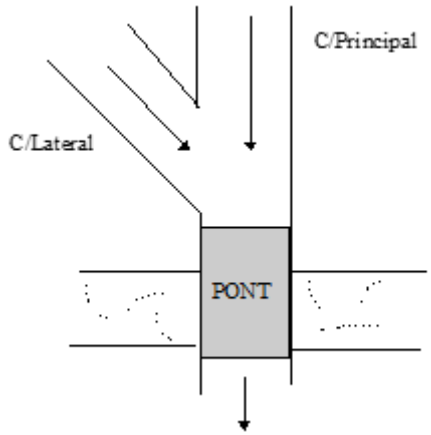
4. Cinc filòsofs passen la seva vida pensant i menjant. Els filòsofs comparteixen una taula circular on hi ha cinc cadires, una per a cadascun d'ells. Cada filòsof té un plat per menjar, però només pot menjar si disposa de dos coberts. Quan un filòsof pensa no interactua amb els seus companys. De tant en quan un filòsof té gana i intenta agafar dos coberts per menjar. Quan un filòsof té gana i disposa del cobert de la dreta i del de l'esquerra del seu plat, pot menjar i ho fa sense deixar els coberts. Quan el filòsof acaba de menjar diposita els coberts un altre cop sobre la taula i es posa a pensar una altra vegada. Implementa la solució a aquest problema utilitzant semàfors.



5. En l'entrada d'un pont unidireccional s'ajunten dos carrers. (c/Principal i c/Lateral). Els vehicles que vinguin dels dos carrers i vulguin creuar el pont hauran de tenir en compte les següents consideracions:

El pont sols pot suportar com a màxim el pes de 5 vehicles.
Quan pel carrer Principal hi hagi vehicles que vulguin creuar el pont, tindran sempre prioritat davant els vehicles del carrer Lateral.
Els vehicles que circulen pel c/Lateral poden començar a creuar el pont quan no existeix cap vehicle en el c/ Principal que vulgui creuar el pont.

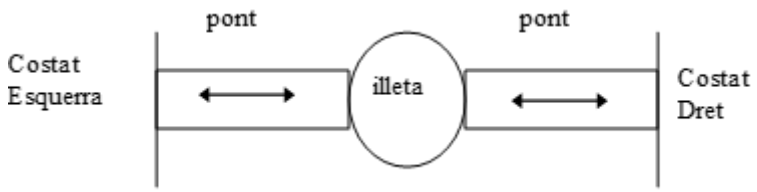
Implementa el codi que gestioni la circulació de vehicles pels dos carrers utilitzant semàfors.



6. Suposem una perruqueria on es disposen de 3 seients d'espera on els clients resten asseguts en tant no són atesos, i 2 seients de tall de cabell. Quan els 5 seients estan plens, els clients han d'esperar-se fora de la perruqueria. Un client quan arriba, primer s'asseu en els seients d'espera i després passa a tallar-se el cabell en un dels seients de tall que hi hagi lliure. Implementar una solució amb semàfors que controli l'accés als seients d'espera i de tall per part d'un client.

7. Suposem dos ponts separats per una illeta com es mostra en el dibuix. Escriure el codi amb semàfors per controlar l'accés als ponts per part dels vianants del costat esquerra i del costat dret, tenint en compte les següents restriccions:

- Només un vianant pot estar en un pont a la vegada
- A l'illa petita poden esperar set vianants com a màxim



8. El govern d'un estat fictici està avaluant la possibilitat de construir un nou aeroport. Inicialment volen veure com funcionaria si es dotés amb tres pistes, que podrien ser utilitzades tant per aterrar com per enlairar-se, però, evidentment, només per un avió alhora. En aquest projecte, ens encarreguen escriure un pseudo-codi basat en l'ús de semàfors que descrigui el protocol que hauria de fer servir un avió que vulgui enlairar-se per accedir a una de les tres pistes. Tenim la següent informació:

- Les pistes tenen noms (A, B i C),
- L'acció d'enlairar-se està modelada per una rutina (que no cal implementar) que anomenarem **Enlairar(n)**, la qual rep com a paràmetre el nom de la pista que es farà sevir.
- L'estat del sistema està descrit per tres variables compartides anomenades **estat_pista_A**, **estat_pista_B** i **estat_pista_C**. Si la variable val 1, vol dir que la pista corresponent és lliure (cap avió la fa servir per aterrar o enlairar-se en aquest moment), si val 0, vol dir que la pista està ocupada.
- La solució incomplerta al problema seria:

```
int estat_pista_A = 1, estat_pista_B = 1, estat_pista_C = 1;
```

```
<declaració i inicialització dels semàfors>

Enlairar_Avio(){

        char pista;

        <protocol per decidir la pista a utilitzar>

        Enlairar(pista);

        <protocol a seguir després d'enlairar-se>

}
```

Completeu aquesta solució indicant clarament quins semàfors fareu servir, el seu valor inicial, el codi que s'ha d'executar abans d'enlairar-se (per esperar una pista lliure i saber quina farà servir) i el codi que s'ha d'executar després d'enlairar-se per actualitzar l'estat del sistema.

Disseny d'algoritmes paral·lels (Metodologia de Foster)

9. El següent fragment de codi seqüencial implementa la multiplicació de dues matrius de dimensió NxN (C = AxB):

```
float A[N][N], B[N][N], C[N][N];

for(int  i = 0; i < N; i++)
        for(int j = 0; j < N; j++){
                C[i][j] = 0;
                for(int z = 0; z < N; z++) C[i][j]+= A[i][z] * B[z][j];
        }
```

Dissenyeu un algorisme paral·lel per implementar aquesta operació en un sistema de memòria distribuïda amb N unitats de còmput.

10. El següent fragment de codi seqüencial implementa la multiplicació d'una matriu de dimensió NxN per un vector de dimensió N (C = A x v):

```
float A[N][N], v[N], C[N];

for(int i = 0; i < N; i++ ){
        C[i] = 0;
        for(int z = 0; z < N; z++) C[i]+= A[i][z] * v[z];
}
```

Dissenyeu un algorisme paral·lel per implementar aquesta operació en un sistema de memòria compartida amb N unitats de còmput.

11. *Càlcul número Pi*. Pi és la relació entre la circumferència i el diàmetre d'un cercle o entre l'àrea i el radi al quadrat. Una manera d'aproximar el càlcul de Pi consisteix a aproximar l'àrea d'una circumferència generant punts de forma aleatòria dins el quadrat que la circumscriu. Dissenyi l'algoritme que realitza aquest còmput de forma paral·lela.

12. *N-Body*. Un bon nombre de problemes en física i química requereixen la simulació de la interacció d'un conjunt de partícules entre si, aquest problema rep el nom general de N-Body. Assumint que tenim un nombre N de partícules de les quals coneixem la seva posició inicial i la seva massa, que coneixem les lleis que permeten calcular la força que exerceixen unes partícules sobre unes altres, així com aquelles que ens permeten calcular l'efecte que produeixen aquestes forces en la posició de cada partícula, dissenyi un algoritme paral·lel que calculi la posició de cada partícula en un rang de temps T en passos de Δt.

13. *Relaxació de Jacobi*. Algoritme iteratiu que troba la solució discretitzada a una equació diferencial de la forma:

$$Ax+b=0$$

Per exemple, donada una superfície d'un cert material a la qual s'aplica una font de calor en un determinat punt, es pot calcular la temperatura a tota la superfície en diferents passos de temps discretitzant l'àrea de la superfície i aplicant l'expressió de transferència de calor adequada al material en cada punt per a cada pas de temps. El problema acaba quan la variació global de temperatura entre dos passos de temps és menor que una certa tolerància preestablerta.

Dissenyi un algoritme paral·lel per solucionar el problema anterior, raoni sobre l'escalabilitat de l'algoritme proposat.

14. *Quick Sort*. Proposi una versió paral·lela d'aquest algoritme d'ordenació i analitzi la qualitat de la mateixa. Aquest problema té un enunciat molt senzill, però la solució és força complexa. Analitzeu amb atenció quines són les tasques bàsiques, les dependències entre les mateixes i la potencialitat d'accel·leració de l'algoritme resultant.

15. *DFS*. Proposi una versió paral·lela de l'estratègia de recerca en profunditat i analitzi la qualitat de la mateixa. Novament ens trobem amb un enunciat curt d'un problema complex. Probablement, el plantejament de la solució paral·lela bàsica no sigui molt complicat, però l'anàlisi de les prestacions i, en particular, el plantejament de la fase d'assignació no sigui gens trivial.

Índexs de Rendiment

16. Es té un programa en un ordinador X que triga 35 segons a executar-se mentre que a l'ordinador Y empra 21 segons. El programa està format per 522 milions d'instruccions. Què tan més ràpid és l'ordinador Y que el X? Quina quantitat d'instruccions per segon (MIPS) executa cada ordinador?

17. S'ha executat una aplicació en un multiprocessador amb 32 processadors; un 10% del codi s'ha executat als 32 processadors, un 70% a 16 processadors, i la resta a 4 processadors. Calculeu el factor d'acceleració (speed-up) i l'eficiència que s'aconseguiran (sense considerar altres factors).

18. Es vol millorar el rendiment d'un ordinador introduint una targeta acceleradora de vídeo que faci les operacions en la meitat de temps.

a. Calcular el guany en velocitat del sistema per a l'execució d'un programa si el 87% es dedica a operacions gràfiques.

b. Si el programa triga 32 segons en executar-se sense la millora, quant trigarà amb la millora?

19. Suposeu que el 90% d'un programa pot ser paral·lelitzat. Quina és l'acceleració màxima possible segons la llei d'Amdahl? Quin serà el temps mínim d'execució del programa paral·lel, si el programa original triga 10 hores?

20. Suposeu que un ordinador té un recurs capaç d'augmentar la velocitat de processament de les aplicacions en un factor p, però que hi ha una probabilitat f de no utilitzar aquest recurs. A partir d'aquest supòsit, deduïu la llei d'Amdahl.

21. Sigui un programa que posseeix 5% del codi seqüencial i el 95% perfectament paral·lelitzable. Calculeu l'speed-up per a n = 8 i n = 16 processadors. Segons les lleis d'Amdahl i de Gustafson.

22. Suposeu que voleu paral·lelitzar un determinat programa seqüencial i que voleu obtenir un factor d'acceleració mínim de 20 amb 50 processadors. Determineu quina és la fracció sèrie màxima que pot tenir el programa sèrie (i) segons la llei d'Amdahl i (ii) segons la llei de Gustafson.

23. Suposeu que voleu paral·lelitzar un determinat programa seqüencial i que voleu obtenir un factor d'acceleració mínim de 20 amb 50 processadors. Determineu quina és la fracció sèrie màxima que pot tenir el programa sèrie en el cas que per 50 processadors la part sèrie hagi crescut un 15% i la part paral·lelitzable hagi crescut un 200%, respecte a la versió seqüencial.

24. Un programa es compon de dues parts, f1 i f2; aquestes parts (funcions) s'han d'executar una després de l'altra, i el temps d'execució de totes dues és similar. La primera part es pot executar en paral·lel utilitzant qualsevol nombre de processadors, i no hi ha sobrecàrrega associada per executar-la en paral·lel; amb la segona part, en canvi, no es poden utilitzar més de 8 processadors en la seva execució paral·lela. El sistema utilitzat per executar el programa té P processadors. Calculeu:

a. El factor d'acceleració en funció de P, i el factor d'acceleració màxim teòric.

b. El nombre de processadors que cal utilitzar per aconseguir una eficiència de 0,5 i el factor d'acceleració que s'aconseguirà en aquest cas.

25. Mostreu que, segons Amdahl, la porció paral·lelitzable d'un programa es pot estimar com:

$$X = \frac{(1/S\,(p)) - 1}{(1/p) - 1}$$

On, S(p) és l'acceleració que s'obté per a p processadors.
Deduïu una fórmula per calcular l'acceleració màxima, assumint que coneixem només l'acceleració per a p processadors (S(p)).

26. Suposeu que un programa executa 10^{12} instruccions per solucionar un problema determinat. Suposeu a més que el programa s'executa en un processador que és capaç solucionar el problema en 10^6 segons (gairebé 12 dies), és a dir, un processador capaç d'executar 10^6 instruccions/seg.
Suposeu ara que el programa ha estat paral·lelitzat per a la seva execució en un sistema de memòria distribuït. Suposeu també que si el programa paral·lel utilitza p processadors, cada processador executarà 10^{12}/p instruccions i enviarà 10^9(p-1) missatges.

Suposeu finalment, que no hi ha cap altra sobrecàrrega en executar el programa paral·lel, és a dir, els processos no han de per exemple esperar pels missatges d'altres processos.
Quant de temps necessitarà el programa per executar-se en 1000 processadors igual de ràpids que el processador original si cada missatge necessita 10^{-9} seg per ser enviat? i si es necessiten 10^{-3} seg per missatge?

27. S'executa una aplicació en paral·lel a una màquina de P processadors. Quan s'executa en paral·lel, per cada segon d'execució en sèrie s'ha de realitzar una operació de comunicació, el cost de la qual es pot modelar com a $40 + 16 \times P^{1/2}$ mil·lisegons.
Sense tenir en compte altres aspectes, calcula el factor d'acceleració màxim que es pot aconseguir i el nombre de processadors que cal utilitzar per aconseguir-ho.

28. Segons la llei d'Amdahl. Quin percentatge del codi d'un programa ha de ser perfectament paral·lelitzable si volem que sigui escalable fins a 16 processadors mantenint com a mínim una eficiència de 0.5? I fins a 32? I segons la llei de Gustafson?

29. Suposeu que per a una certa aplicació $T_{sèrie} = n$, mentre que $T_{paral·lel} = n/p + \log_2(p)$. Si incrementem p en un factor k, trobeu la fórmula que permeti calcular quan s'ha d'incrementar n per mantenir l'eficiència constant. Quan s'ha d'incrementar n per mantenir l'eficiència constant si doblem el nombre de processadors de 8 a 16? És el programa paral·lel escalable?

30. Assumiu que volem paral·lelitzar un programa seqüencial donat i obtenir almenys una acceleració de 10 amb 16 processadors. Quina és la fracció sèrie màxima del programa segons les lleis d'Amdahl i Gustafson?

31. Suposeu que tenim un programa paral·lel que:
   a. Té una part inherentment seqüencial amb un temps d'execució de (2500+n)ms, on n és la mida del problema.
   b. Té una part paral·lela amb un temps d'execució de n^2 ms.

Quina és la màxima acceleració possible d'aquest programa per a una mida de problema de n=10000?