

E

La concurrence confit écriture/lecture

Soit A une donnée partagée et 2 processus P_1, P_2 de même code :

$x \leftarrow \text{lire}(A)$

$x \leftarrow x + 1$

Ecrire (A, x)

Si $A = 3$ et on exécute P_1 puis P_2

$A = 5$

si on exécute P_2 puis P_1

$A = 5$

si : $P_1 \quad x \leftarrow \text{lire}(A)$

$P_1 \quad x \leftarrow x + 1$

$P_2 \quad x \leftarrow \text{lire}(A)$

$P_2 \quad \text{Ecriture}(x, A)$

$P_2 \quad x \leftarrow x + 1$

$P_2 \quad \text{Ecriture}(A, x)$

$\Rightarrow A = 4$

Définition :

On appelle section critique la portion de code dans laquelle le processus accède à la ressource partagée

Le problème de concurrence :

consiste à garantir :

- Sureté : un processus au plus exécute sa section critique à un instant donné
- Vivacité : un processus demandant à entrer en section critique finit par exécuter sa section critique

Solution:

- désactivation des interruptions (laisser la main de manière inconditionnelle à un processus \Rightarrow suspendre tout les autres processus)
 - \hookrightarrow inenvisageable aujourd'hui
- variable verrou
 - \hookrightarrow marche pas : variable verrou est aussi soumise aux accès concurrentes \Rightarrow dead lock
- Alternance stricte. (~~un autre système qui n'utilise pas la variable définit qui l'utilise~~ (marche sur des systèmes distribués mais pas SE))
 - \hookrightarrow fonctionne à petite échelle
 - (un coordinateur répartit les accès aux ressources partagées aux différents processus)

Algorithme de Peterson:

Tant que vrai

Actions avant la section critique

$D_i \leftarrow$ vrai

Tour $\leftarrow i$

Tant que $(D_{(i+1)/2})$ et $(\text{Tour} = i)$

rien

Actions section critique

$D_i \leftarrow$ faux

Actions après section Critique

fonctionne pour 2 processus seulement

bonne
mais

Test : dérouler l'algo avec :

$$D_0 = \text{faux} \quad D_1 = \text{faux} \quad \text{Tour} = 0$$

$$P_i, \quad i=1 \quad P_0 + i = 0$$

Scénario : P_0 demande la section critique

puis P_1

puis P_0 et P_1

→ les sémaphores : c'est un drapeau (levé ou baissé) qui indique l'état d'une ressource

Un sémaphore (info) est une variable dont les accès se fait à travers 2 actions :

$P()$ $V()$

$P()$: consiste à tester la valeur du sémaphore et si la valeur est strictement positive, elle est décrémentée. L'action est bloquante (?)

Ces opérations sont atomiques

L'action $V()$ incrémente la valeur d'un sémaphore

Exemple :

Soient la variable $A=3$ et P_1, P_2, P_3 3 processus de même code :

$x \leftarrow \text{lire}(A)$

$x \leftarrow x + 1$

Ecrire (A, x)

On considère un sémaphore s initialisé à 1 avec le code :

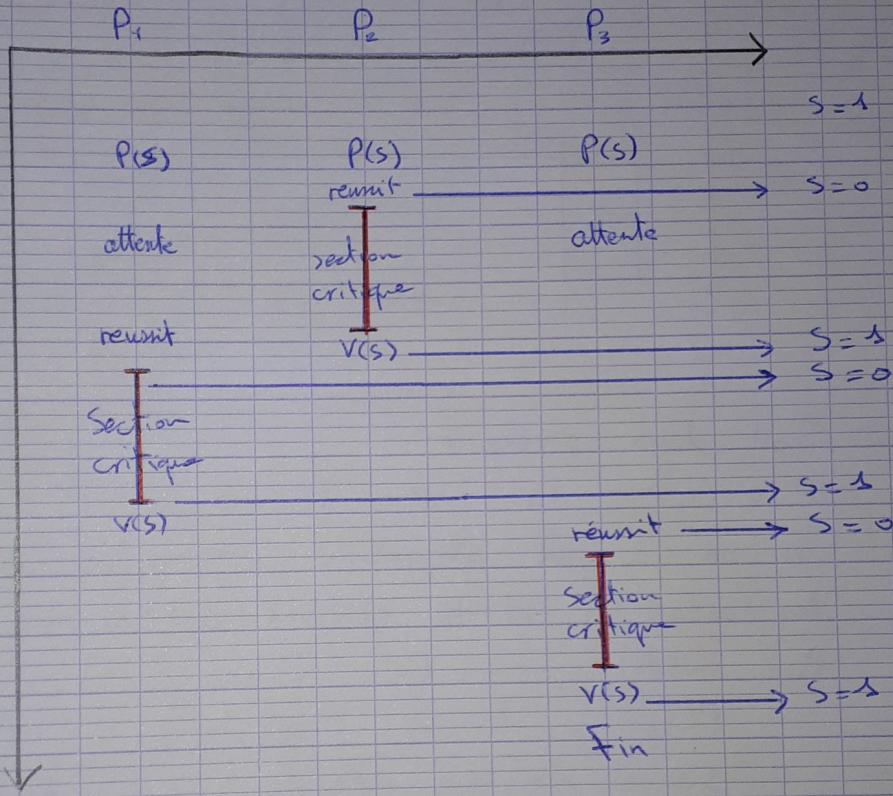
$P(s)$

$x \leftarrow \text{line}(A)$

$x \leftarrow x + 1$

Ecrire (A, x)

$V(s)$

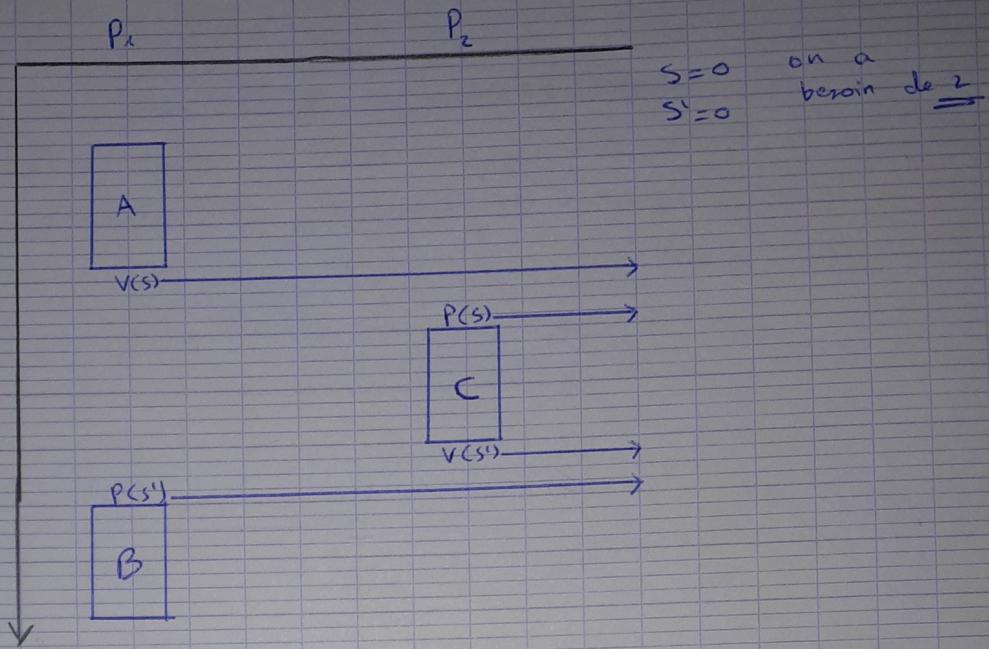


Exo:

Soient P_1 et P_2 2 processus. P_1 est décomposé en 2 parties, A et B. P_2 est composé d'une partie C. On souhaite que A précède C et que C précède B.

Ecrire un code à l'aide de sémaphores qui garantit cela.





Soit deux processus :

$P_1 : A, sc, B$ et $P_2 : C, sc, D$

Si on a un sémaphore S initialisé à 1 à l'exécution des codes P_1 et P_2 suivant de mémoire concurrenente :

$P_1 : A$

$P(S)$

sc

$V(S)$

B

$P_2 : C$

$P(S)$

sc

$V(S)$

D

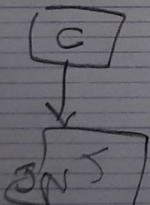
satisfait la prop.
de sûreté

JNJ

exécutions :

$P_2 C, P_2 P(S), P_1 A, P_2 sc, P_1 P(S)$, $P_2 D, P_1 V(S)$, $P_1 sc$
attente

$P_2 D, P_1 V(S), P_1 B$



[8]

* soient 2 processus P_1 et P_2 :

$P_1 : A, C$

$P_2 : B$

On souhaite ordonner les actions de manière que A précède B et B précède C

si s_1 et s_2 initialisés à 0 :

P_1

A

$V(s_1)$

$P(s_1)$

B

$V(s_2)$

$P(s_2)$

C

* Soient 4 processus P_1, P_2, P_3, P_4 .

P_1 et P_2 composés de 4 parties (ABCD)

P_2 et P_4 ————— 3 ————— ABC

et on a les EM suivants

P_1 B est en exclusion mutuelle avec P_3 C

P_2 A ————— P_1 A

P_4 C ————— P_3 D

en outre on a :

P_1 A précède P_2 A

P_2 A ————— P_2 A

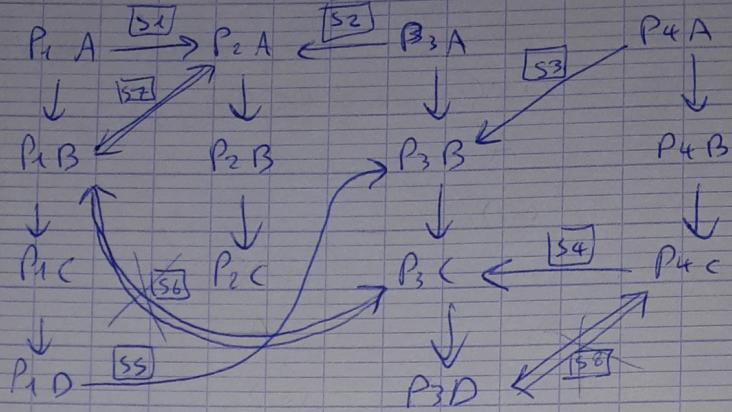
P_4 C ————— P_3 C

P_1 D ————— P_3 B

P_4 A ————— P_3 B

Table de préférence :

EM
double flèche



\Rightarrow 8 sémaphores si à ss initialisés à 0
 S6 à S8 initialisés à 1

A

P1: $V(S1)$ $P(S7)$ $P(S6)$ ~~B~~ $V(S7)$ ~~D~~ $V(ss)$

P2: $P(S1)$ $(P(S2))$ $P(S7)$ A $V(S7)$ B C

P3: A $V(S2)$ $P(S3)$ $P(ss)$ B $P(S4)$ $P(S6)$ C $V(S6)$
~~P(S8)~~ D $V(S8)$

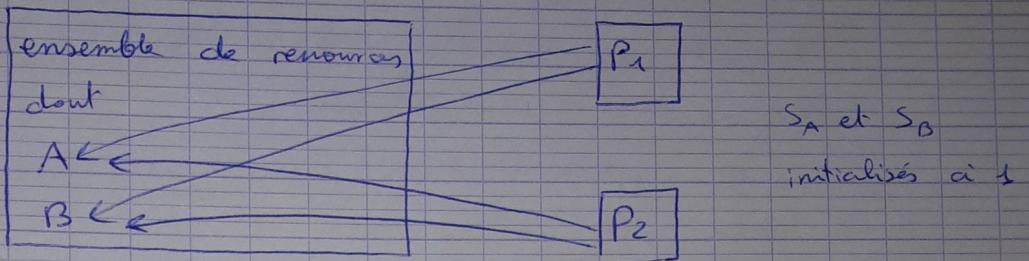
P4: R $V(S4)$ B $P(S8)$ C $V(S8)$ $V(S4)$

S8 on en a pas besoin car

$P4C \rightarrow P3C \rightarrow P3D \Rightarrow P4C \rightarrow P3D$

S6: $P_1B^* \rightarrow P_3C$

Soient 2 ressources A et B et 2 processus P₁ et P₂ qui doivent avoir accès à ces 2 ressources par exécuter leur code de SC l'accès à A s'obtient avec le sémaphore S_A et respectivement avec B



P₁: P(S_A) P(S_B) SC V(S_A) V(S_B)

P₂:

⇒ ok

P₁: P(S_A) | A(S_B) SC V(S_A) V(S_B)

P₂: P(S_B) | P(S_A) _____

Σ blocage : interblocage = conflit entre processus

Résoudre un interblocage:

La situat° d'interblocage se détecte et s'analyse grâce au graphe des ressources.

exemple: banque:

ressources : - imprimante

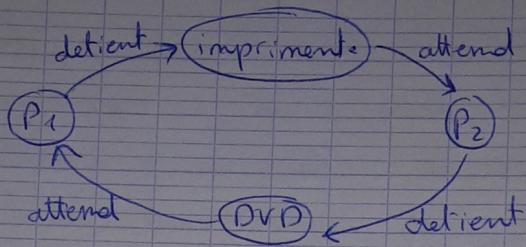
- lecteur DVD

① P₁ demande imprimante

② P₁ demande DVD

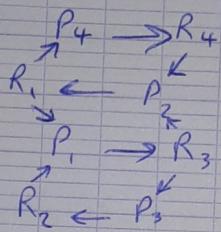
② P₂ demande DVD

① P₂ demande imprimante



Un graph des ressources détenues montre un circuit indique une situation d'interblocage
 \Rightarrow suppression d'un processus

situation plus complexe :

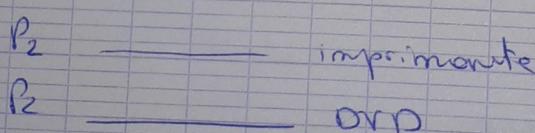
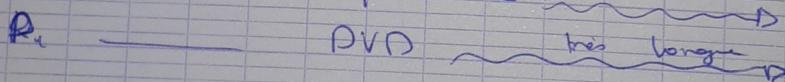


choix du processus à supprimer :

- temps d'exécution déjà effectué
- nb d'interblocages dans lesquels le processus est impliqué

Cas d'un interblocage inexistant: du aux délais de notification

P1 accède à imprimante



\Rightarrow filete mise trop tard

