

Examen semestriel

Module de Systèmes d'exploitation II

Nom et Prénom :

**Exercice : (20 points)**

On considère le problème du Producteur/Consommateur. Proposez une solution pour chacun des cas suivants :

**Cas 1 : Le buffer est circulaire et de capacité limitée. Il existe plusieurs producteurs et plusieurs consommateurs. La solution doit utiliser les sémaphores. (6 points)**

Réponse :

Déclarations :

Buffer : Tableau [0..N-1] de Message ; N étant la taille du buffer.

In : Entier (init 0), Indice de la case prête à recevoir un message déposé.

Out : Entier (init 0), indice de la case contenant un message prêt à être consommé.

Full : sémaphore (init 0), représente le nombre de cases pleines.

Empty : sémaphore (init N), représente le nombre de case vides.

Mutex1 : sémaphore (init 1), permet l'utilisation en exclusion mutuelle de la variable In (partagée par plusieurs producteurs)

Mutex2 : sémaphore (init 1), permet l'utilisation en exclusion mutuelle de la variable Out (partagée par plusieurs consommateurs)

<p>Processus Producteur i Début</p> <p style="padding-left: 40px;">Cycle</p> <p style="padding-left: 80px;">Produire un Message ; Wait (Empty) ; Wait (mutex1) ; Buffer[In] :=Message ; In :=In+1 mod N ; Signal(Mutex1) ; Signal(Full) ;</p> <p style="padding-left: 40px;">Fin cycle</p> <p>Fin.</p>	<p>Processus Consommateur i Début</p> <p style="padding-left: 40px;">Cycle</p> <p style="padding-left: 80px;">Wait (Full); Wait (mutex2); Message:=Buffer[Out]; Out :=Out+1 mod N; Signal(Mutex2) Signal(Empty); Consommer Message</p> <p style="padding-left: 40px;">Fin cycle</p> <p>Fin.</p>
--	---

**Cas 2 : Le buffer est circulaire et de capacité limitée. Il existe plusieurs producteurs et plusieurs consommateurs. La solution doit utiliser les moniteurs de Hoare. (6 points)**

Réponse : L'accès en exclusion mutuelle des variables In, Out et Nb est assuré par le moniteur lui-même.

<p>Type M = Monitor Var Buffer : Array[0..N] of Char; In , Out, Nb: integer ; x, y : condition ;</p> <p>Procédure Deposer(C : Char) Begin     if Nb=N then x.wait ;     Buffer[In]:=C; In:=In+1 mod N;     y.signal End Procédure Prelever(var C : Char) : Char Begin     if Nb=0 then y.wait ;     c:=Buffer[Out]; Out:=Out+1 mod N;     x.signal End Begin /*Initialisation */     In:=0; Out:=0; Nb:=0 End</p>	<p>Processus Producteur i Début Cycle     Produire(Message)     M.Deposer(Message) Fin cycle Fin.</p>	<p>Processus Producteur i Début Cycle     M.Prelever(Message)     Consommer(Message) Fin cycle Fin.</p>
---	---	---

**Cas 3 : Le buffer est circulaire et de capacité limitée. Il existe un producteur et un consommateur. Mais on impose que le consommateur ne doit pas consommer plus de x messages toutes les s secondes. On utilise un processus « Horloge » qui se déclenche toutes les s secondes. La solution doit utiliser les sémaphores. (07 points)**

Réponse :

Déclarations :

Buffer : Tableau [0..N-1] de Message ; N étant la taille du buffer.

In : Entier (init 0), Indice de la case prête à recevoir un message déposé.

Out : Entier (init 0), indice de la case contenant un message prêt à être consommé.

Nc : Entier (init 0), compte le nombre de messages consommés pendant une période.

Full : sémaphore (init 0), représente le nombre de cases pleines.

Empty : sémaphore (init N), représente le nombre de case vides.

Mutex : sémaphore (init 1), permet l'utilisation en exclusion mutuelle de la variable Nc .

Prochain : sémaphore (init 0), permet de bloquer le consommateur lorsque le nombre de consommationS est égal à x, jusqu'à la prochaine période.

<p>Processus Producteur</p> <p>Début</p> <p>Cycle</p> <p>    Produire un Message ;</p> <p>    Wait (Empty) ;</p> <p>    Buffer[In] :=Message ;</p> <p>    In :=In+1 mod N ;</p> <p>    Signal(Full) ;</p> <p>Fin cycle</p> <p>Fin.</p>	<p>Processus Consommateur</p> <p>Début</p> <p>Cycle</p> <p>    Wait (Full);</p> <p>    Wait(mutex) ;</p> <p>    Si Nc = x</p> <p>    Alors</p> <p>        Signal(mutex) ;</p> <p>        Wait(Prochain)</p> <p>    Sinon</p> <p>        Signal(mutex)</p> <p>    finsi</p> <p>    Message:=Buffer[Out];</p> <p>    Out :=Out+1 mod N;</p> <p>    Signal(Empty);</p> <p>    Wait(mutex)</p> <p>    Nc :=Nc+1 ;</p> <p>    Signal(mutex) ;</p> <p>    Consommer Message</p> <p>Fin cycle</p> <p>Fin.</p>	<p>Processus Horloge</p> <p>Début</p> <p>    Wait(mutex)</p> <p>    Si NC = x</p> <p>    Alors</p> <p>        Nc :=0 ;</p> <p>        Signal(mutex) ;</p> <p>        Signal(prochain)</p> <p>    Sinon</p> <p>        Nc := 0 ;</p> <p>        Signal(mutex)</p> <p>    finsi</p> <p>Fin.</p>
--	--	---