IPC, SYNCHRONISATION ET COORDINATION

Il est recommandé d'utiliser un tableur pour réaliser les différents calculs. On rappelle que les supports sont disponibles à http://mathieu.delalandre.free.fr/teachings/operating1.html

1. Problème du producteur/consommateur

1.1.Méthode de type « sleep/wakeup »

On se propose de mettre en œuvre une synchronisation de type « sleep/wakeup » au travers d'un problème de type producteur consommateur. Soit deux processus P (le producteur) et C (le consommateur), exécutant les algorithmes suivants. Dans ces algorithmes, les instructions (1-5) sont dites atomiques (non interruptibles par l'ordonnanceur). Les instructions « sleep » et « wakeup » sont implémentées selon le modèle wwb « wakeup waiting bit », spécifié dans les tableaux ci-dessous.

consumer, **producer** are processes

```
consumer                                          producer
    loop                                              loop
(1)     if buffer is empty                        (1)     if buffer is full
(2)         sleep                                 (2)         sleep
(3)     pop item from buffer                      (3)     push a new item in buffer
(4)     if buffer was full (i.e. actual size = n-1)(4)    if buffer was empty (i.e. actual size =1)
(5)         wakeup producer                       (5)         wakeup consumer
```

wakeup waiting bit

command	0	1
sleep	sleep	put to 0

process state

command	ready	waiting
wakeup	put to 1	wakeup

Définir la synchronisation des processus P et C au travers du schéma d'ordonnancement suivant, ou l'ordonnanceur passe la main tour à tour aux processus pour un nombre d'instruction donné. A t=0, les états des processus P et C sont respectivement « sleepy » et « awake ». On considéra ici l'état d'exécution du processus C comme en début de boucle (i.e. exécution de l'instruction (1)). Le buffer a une taille maximum de 5, et est initialisé à n=5 à t=0. Les valeurs des bits wwb des deux processus P et C sont à « false ». Indiquez pour chacun des changement de contexte les informations suivantes: instructions exécutées, état du buffer, états des wwb pour P et C, états des processus P et C. Sur l'ensemble de la synchronisation, indiquez les moments ou les wwb interagissent dans la coordination producteur-consommateur.

Nb d'instruction	4	1	1	2	3	3	1	2
Processus	C	P	C	P	C	P	C	P

1.2. Synchronisation par sémaphore

On se propose de mettre en œuvre une synchronisation par sémaphore au travers d'un problème de type producteur consommateur. Soit deux processus P (le producteur) et C (le consommateur), exécutant les algorithmes suivants. Dans ces algorithmes, les instructions (1-3) sont dites atomiques (non interruptibles par l'ordonnanceur).

fill = 0, **empty** = n are semaphores
buffer is the data structure

consumer	producer
loop	loop
(1) down **fill**	(1) down **empty**
(2) pop **item** from **buffer**	(2) push a new **item** in **buffer**
(3) up **empty**	(3) up **fill**

Définir la synchronisation des processus P et C au travers du schéma d'ordonnancement suivant, ou l'ordonnanceur passe la main tour à tour aux processus pour un nombre d'instruction donné. A t=0, les états des processus C et P sont respectivement « blocked » et « ready » (i.e. C est dans la queue du sémaphore « fill »). Les valeurs du buffer et des sémaphores sont respectivement à 0 (« buffer » et « fill ») et 2 « empty », on considéra une taille maximum de buffer n égale à 2. Indiquez pour chacun des changements de contexte les informations suivantes: instructions exécutées, état du buffer, états des sémaphores (valeur et état de la queue d'exploitation), états des processus P et C.

Nb d'instruction	7	2	3	7	2
Processus	P	C	P	C	P

2. Problème des producteur(s)/consommateur(s) multiples

2.1. Synchronisation par sémaphore-mutex

On se propose de mettre en œuvre une synchronisation, au travers d'un problème de type producteur consommateur impliquant plusieurs acteurs, par sémaphore-mutex. Soit trois processus P (le producteur) et C1, C2 (les consommateurs) exécutant les algorithmes suivants. Dans ces algorithmes, les instructions (1-5) sont dites atomiques (non interruptibles par l'ordonnanceur).

fill = 0, **empty** = n are semaphores
mutex is a mutex
buffer is the data structure

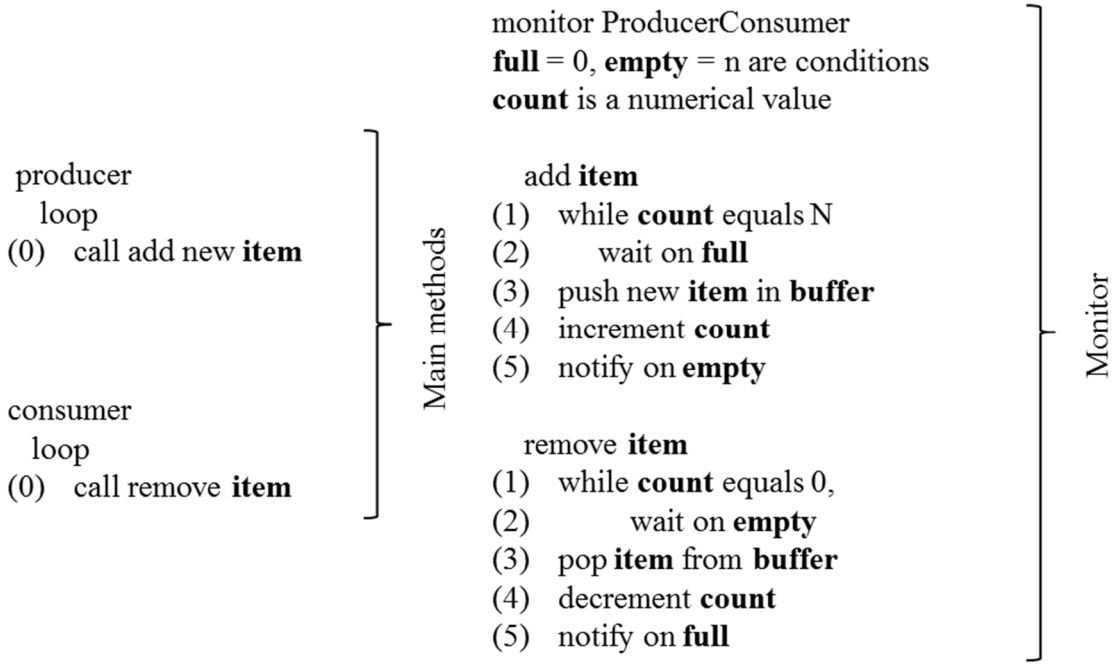
consumer	producer
loop	loop
(1) down **fill**	(1) down **empty**
(2) down **mutex**	(2) down **mutex**
(3) pop **item** from **buffer**	(3) push a new **item** in **buffer**
(4) up **mutex**	(4) up **mutex**
(5) up **empty**	(5) up **fill**

Définir la synchronisation entres les processus P, C1 et C2 au travers du schéma d'ordonnancement suivant, ou l'ordonnanceur passe la main tour à tour aux processus pour un nombre d'instruction donné. A t=0, les états des processus P, C1 et C2 sont « ready ». Les valeurs du buffer et des sémaphores sont respectivement à 4 (« buffer » et « fill ») et 1 « empty ». On considéra une taille maximum de buffer n égale à 5. Indiquez pour chacun des changements de contexte les informations suivantes: instructions exécutées, état du buffer, états des sémaphores et du mutex (valeur et état de la queue d'exploitation), états des processus P, C1 et C2.

Nb d'instruction	6	2	2	3	1	2	2	2	1	1	1	2
Processus	P	C2	C1	C2	P	C2	C1	P	C2	P	C1	C2

2.2. Synchronisation par moniteur

On se propose de mettre en œuvre une synchronisation, au travers d'un problème de type producteur consommateur, impliquant plusieurs acteurs, par moniteur. Soit cinq processus P1, P2, P3 (les producteurs) et C1, C2 (les Consommateurs), exécutant les algorithmes suivants via un moniteur de type « Mesa style ». Dans ces algorithmes, les instructions (0-5) sont dites atomiques (non interruptibles par l'ordonnanceur). Ces instructions se répartissent en deux parties, les appels communs (0) et la section moniteur (1-5).



Définir la synchronisation entres les processus P1, P2, P3 et C1, C2 considérant les paramètres suivants:

- N est égal à 2.
- à t=0, la taille du buffer et de la valeur de « count » égalent N.
- à t=0, P2 est dans la queue associée à la variable conditionnelle « full ».
- considérant la séquence d'ordonnancement (partielle) donnée ci-dessous, ou l'ordonnanceur passe la main tour à tour aux processus pour un nombre d'instruction donné, indiquez les informations manquantes concernant les id des processus et les durées d'exécution en nombre d'instruction.

- indiquez pour chaque changement de contexte les informations suivantes: états des variables « buffer » et « count », états des piles des variables conditionnelles « full » et « empty », détention du moniteur, état de la queue d'entrée.

Nb d'instruction	4	?	?	?	2	5	5	5	≥ 2	4	5	1
Processus	C2	C1	P1	P3	C2	?	?	?	?	?	C1	C2

Reprenez le problème de synchronisation précédent en utilisant une implémentation de type « Hoare style » pour le moniteur, considérant le code ci-dessous et les paramètres suivants:

- la gestion des queues « enter » et « signal » se fait via un algorithme d'ordonnancement type Round Robin avec un « time-slice » sur une répartition 4/5 « (E)nter », 1/5 « (S)ignal ». Au démarrage de l'ordonnancement, le tour est à S(5) (i.e. l'ordonnanceur vient de faire un tour complet, suite au passage de C2 le Round Robin basculera à E(1)).
- considérant la séquence d'ordonnancement (partielle) donnée ci-dessous, ou l'ordonnanceur passe la main tour à tour aux processus pour un nombre d'instruction donné, indiquez les informations manquantes concernant les id des processus et les durées d'exécution en nombre d'instruction.
- indiquez pour chaque changement de contexte les informations suivantes : états des variables « buffer » et « count », états des piles des variables conditionnelles « full » et « empty », détention du moniteur, états des queues « enter » et « signal », ordonnancement « i.e. tour » observé par le Round Robin.

Nb d'instruction	4	?	?	?	?	4	5	5	≥ 2	≥ 4	4	≥ 2
Processus	C2	C1	P1	P3	C2	?	?	?	?	?	?	?