

RICM4 projet système

Programmation concurrente

Application de type producteur - consommateur

Producteur

Tant qu'il reste des messages à produire faire

- On attend pendant un temps aléatoire

- On produit un message

- On insère ce message dans le tampon

- On passe au message suivant (messages--)

Fin faire

A la fin → décrémentation du nombre de producteurs



Consommateur

Tant que vrai faire

On récupère le prochain message du tampon

S'il s'agit du message de fin alors |

On sort de la boucle |-- classe MessageEnd

Fin si |

On attend pendant un temps aléatoire

On traite le message

On passe au message suivant (messages++)

Fin faire

A la fin → décrémentation du nombre de consommateurs



Objectifs n°1 et n°2

- Implémentations directes de ce qu'on a vu en cours et TD
 - Utilisation de wait() et de notify()
 - Spécification des sémaphores du cours

```
public synchronized void acquire() throws InterruptedException
{
    c--;

    if(c < 0)
    {
        wait();
    }
}
```

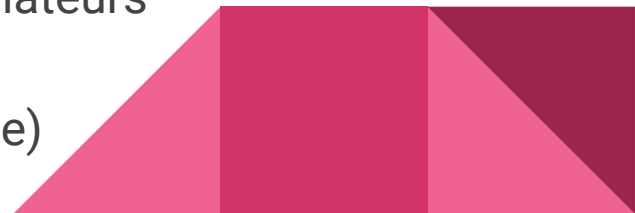
```
public synchronized void release()
{
    c++;

    if(c <= 0)
    {
        notify();
    }
}
```

Objectif n°2

Gestion des sémaphores à prendre en compte

- Retrait message de consommateur, si dernier exemplaire
→ réveil d'un producteur
Sinon
- → réveil d'un consommateur directement
- Lecture message, si plus de producteurs
→ réveil d'un consommateur
→ réaction en chaîne pour arrêter tous les consommateurs
- Arrêt du dernier producteur
→ réveil d'un consommateur (puis réaction en chaîne)



Objectif n°4

- Consommateur

- Après avoir publié un message → attente
- classe MessageTTL (Time To Live) contenant le nombre d'exemplaires restants du message
- Une fois tous les exemplaires consommés → réveil

- Tampon

- Lors du retrait d'un message par un consommateur
- Si dernier exemplaire → réveil du producteur du message
- Méthode wakeup()



Quelques mots

- Architecture de l'application
 - Chaque version (1 à 6) est indépendante
 - Plus grande clarté du code
 - En cas de bug → version précédente
 - MAIS redondance forte, notamment les acteurs
 - Paquetage message
 - Contient les différents types de messages
 - Message MessageEnd → 'arrêt du consommateur
 - Chaque message connaît son créateur
- Impressions
 - Désactivables via la configuration



RICM4 projet système

Programmation concurrente

Application de type producteur - consommateur