ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020

TP 11 – L'autre synchronisation : la barrière

Julien Sopena

novembre 2021

Cette onzième semaine revient sur les mécanismes de synchronisation étudiés la semaine dernière. Ainsi, nous avons déjà vu comment ces derniers permettent à plusieurs processus de travailler sur une même ressource partagée. Dans cette séance, nous verrons comment ces mêmes mécanismes permettent de mettre en place des barrières, *i.e.*, d'attendre la fin d'une action ou d'un calcul fait par un autre processus.

Exercice 1 : Jeudi 16h je ne peux pas, j'ai ...

Nous considérons un ensemble de processus représentant les usagers d'une piscine. La création d'un processus correspond à l'arrivée de l'usager à la piscine.

L'usager doit alors :

1. trouver une cabine disponible,
2. prendre un panier pour y déposer ses vêtements,
3. se changer dans la cabine et la libérer.

L'usager peut alors nager. Une fois sa baignade terminée, le nageur doit :

1. trouver une cabine vide,
2. sortir ses affaires du panier et rendre celui-ci,
3. se changer puis libérer la cabine.

Pour suivre la disponibilité des ressources, nous disposons d'un fichier par type de ressource (`/tmp/numéro_étudiant/nb_cabines`pour les cabines et `/tmp/numéro_étudiant/nb_paniers`pour les paniers où vous devez remplacer `numéro_étudiant` par votre propre numéro). Chacun des fichiers contient un entier qui correspond au nombre de ressources disponibles. Vous devrez créer le répertoire `/tmp/numéro_étudiant` et les fichiers qu'il contient.

Pour prendre une ressource, il faudra suivre l'algorithme suivant :

```
lire le contenu du fichier (pour savoir combien de ressources sont disponibles)
tant que la valeur lue n'est pas au moins égale à 1
    forcer une commutation (1 seconde d'interruption en espérant la libération d'une ressource)
    lire à nouveau le contenu du fichier
fin tant que
mettre à jour le contenu du fichier (pour signaler qu'une ressource de moins est disponible)
```

Pour libérer une ressource, il faut suivre l'algorithme suivant :

```
mettre à jour le contenu du fichier (pour signaler qu'une ressource de plus est disponible)
```

Question 1

Écrivez le script `prendre_ressource.sh` qui prend en paramètre le nom d'un fichier (celui qui contient le nombre de ressources disponibles) et qui prend une ressource. Écrivez le script `rendre_ressource.sh` qui prend en paramètre le nom d'un fichier (celui qui contient le nombre de ressources disponibles) et qui rend une ressource.

Question 2

Soit l'algorithme suivant réalisé par le script `usager.sh` qui représente les usagers de la piscine :

```
afficher "Arrivee de l'usager PID"
instructions de prise d'une cabine
instructions de prise d'un panier
instructions de libération d'une cabine
afficher "Usager PID se baigne"
forcer une commutation et attendre 3 secondes (pour représenter le temps de la baignade)
instructions de prise d'une cabine
instructions de libération d'un panier
instructions de libération d'une cabine
afficher "Fin de PID"
```

Écrivez le script `usager.sh` correspondant à l'algorithme précédent (la prise et la libération d'une ressource ne doivent pas créer de nouveaux processus) et un script `lancement.sh` qui :

1. rend disponibles 5 paniers (le contenu du fichier `/tmp/numéro_étudiant/nb_paniers`doit être 5),
2. rend disponibles 3 cabines (le contenu du fichier `/tmp/numéro_étudiant/nb_cabines`doit être 3),
3. lance l'exécution en parallèle de 7 usagers,
4. attend la fin des usagers avant de se terminer (indice : tous les processus usagers sont des fils du processus de lancement).

Attention, votre solution doit permettre de facilement modifier le nombre de ressources disponibles et le nombre d'usagers créés.

Exécutez le script `lancement.sh` plusieurs fois, jusqu'à ce que le contenu des fichiers `/tmp/numéro_étudiant/nb_paniers`et `/tmp/numéro_étudiant/nb_cabines`ne soit pas cohérent avec le nombre initial des ressources. Comment expliquez-vous les valeurs contenues dans les fichiers `/tmp/numéro_étudiant/nb_paniers`et `/tmp/numéro_étudiant/nb_cabines`obtenues à la fin de l'exécution ?

Question 3

Pour éviter les problèmes identifiés en question 2, nous protégeons les instructions de prise d'une ressource (et de libération) par un verrou (un verrou par type de ressource) :

— Lors de la prise d'une ressource, nous choisissons de prendre le verrou avant la première lecture dans le fichier associé et de le libérer après la mise à jour du contenu du fichier.

— Lors de la libération d'une ressource, le verrou est pris avant la première instruction et rendu après la dernière.

Écrivez une nouvelle version des scripts `prendre_ressource.sh` et `rendre_ressource.sh`.

Pour prendre un verrou, vous devez utiliser la commande `iss_synchro` que nous avons utilisée la semaine dernière.

ATTENTION, avant chaque nouvelle exécution du script `lancement.sh` vous devez vous assurer que (pourquoi ne pas faire un script `clean.sh` :

— tous les processus `usager.sh` et `lancement.sh` sont terminés (pour le vérifier, utilisez la commande `ps` ou la commande `pgrep`),
— les verrous ont tous été détruits (`iss_synchro destroy <verrou>`)

Exécutez plusieurs fois le script `lancement.sh` jusqu'à ce que vous constatiez un blocage. Pour faciliter l'apparition de ce blocage, ajoutez, dans le script `usager.sh`, l'instruction `sleep 5` juste après la première prise d'une cabine. Expliquez la cause de ce blocage (indice : regardez le contenu des fichiers `nb_cabines` et `nb_paniers`) ?

Attention, une explication n'est pas "L'exécution se bloque à cause de la valeur du contenu des fichiers". Elle doit expliquer l'enchaînement des actions qui a mené à la situation constatée.

Question 4

Écrivez une nouvelle version des scripts qui ne présente pas les problèmes identifiés dans les questions précédentes.

Question 5

Modifiez le script `usager.sh` en supprimant l'instruction `sleep 5` que vous avez ajoutée dans la question 3 et faites passer le temps de baignade à 15 secondes. Testez votre nouveau script pour 10 usagers, 5 paniers et 3 cabines (modifiez le script `lancement.sh`). Expliquez pourquoi l'exécution se bloque à nouveau (regardez le contenu des fichiers `nb_cabines` et `nb_paniers`). **Attention**, là encore, une explication n'est pas "L'exécution se bloque à cause de la valeur du contenu des fichiers". Elle doit expliquer l'enchaînement des actions qui a mené à la situation constatée.

Question 6

Que se passe-t-il, si l'on inverse l'ordre de prise du panier et de la cabine ? Un usager commence par prendre un panier avant de prendre une cabine pour se changer. Il libère alors la cabine, se baigne, prend à nouveau une cabine, se change puis libère le panier et la cabine.

