

# Les interblocages

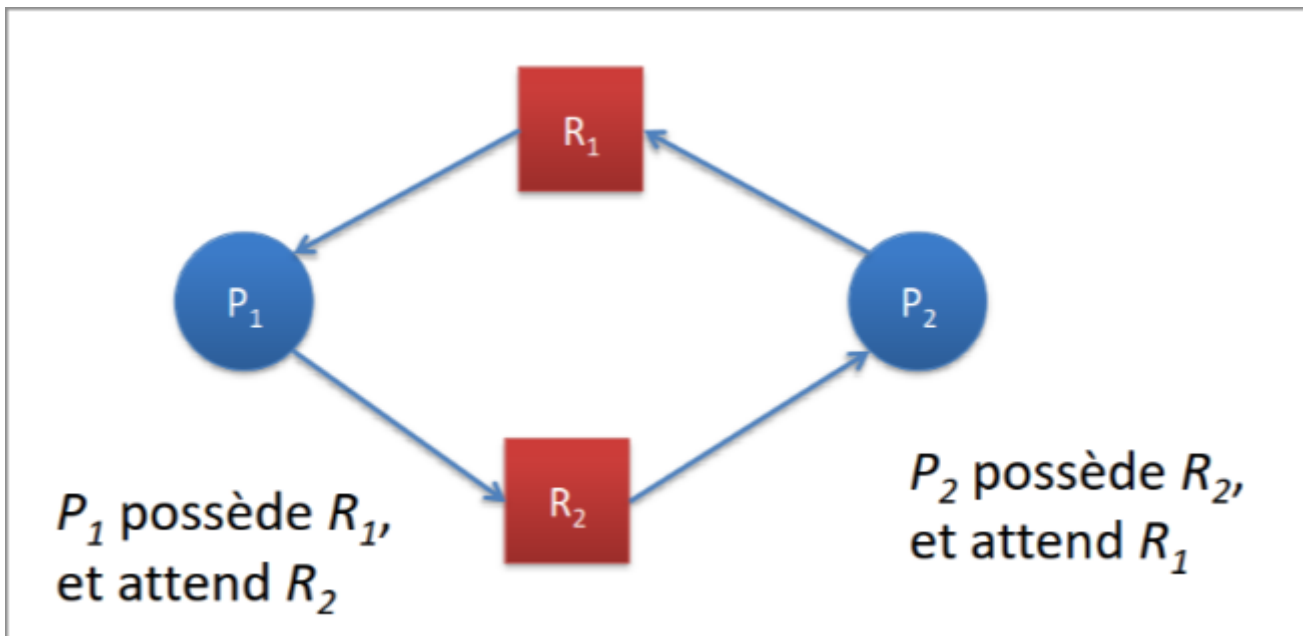
Les ressources (la mémoire, CPU, périphériques, etc) sont limitées, il faut donc gérer les ressources de manière efficace pour permettre au plus grand nombre de processus de s'exécuter.

Un interblocage peut survenir si un processus détient une ressource A qui est demandée par un autre processus détenant une ressource B qui est elle-même demandée par le premier processus.

## Conditions d'un interblocage

Un interblocage survient lorsque ces 4 conditions sont réunies simultanément :

- L'exclusion mutuelle, c'est lorsque les processus utilisent des ressources qui ne peuvent pas être partagées.
- La détention et l'attente, les processus doivent à la fois détenir une ressource et attendre une autre ressource.
- L'impossibilité de réquisitionner une ressource, car c'est dégueulasse et que l'on ne peut pas savoir l'état de la ressource.
- L'attente circulaire, voir plus bas



## Empêcher un interblocage

Pour empêcher un interblocage il faut empêcher l'une des conditions d'arriver.

- L'exclusion mutuelle ? on ne peut pas empêcher un processus de détenir des ressources non partageable
- La détention et l'attente ? Il y a deux solutions pour faire en sorte que la détention et l'attente n'arrive pas en même temps :
  - Un processus pourrait demander toutes les ressources dont il pourrait avoir besoin dès le départ de son exécution
  - Lorsqu'un processus demande une nouvelle ressource, il doit libérer toutes les autres puis récupérer toutes ces ressources, plus la ressource demandée. Ce qui signifie que le processus accumule toujours plus de ressources ce qui peut créer une famine parmi les autres.
- Impossibilité de réquisitionner une ressource ? Il n'est pas possible de s'assurer que les ressources seront dans un bon état lorsqu'elle sont réquisitionnées
- L'attente circulaire ? On peut essayer de détecter un cycle et si un cycle arrive, on peut par exemple numéroté chaque ressource et imposer aux ressources de demander les ressources dans l'ordre croissant de leur numéro.

## Eviter l'attente circulaire

Pour éviter l'attente circulaire il faut donc savoir la quantité de ressources disponibles et occupées ainsi que les besoins de chaque processus.

Le système est dit en **état sûr** s'il est capable de satisfaire tous les processus. Et tant que le système évolue d'état sûr en état sûr, aucun interblocage ne peut survenir. Ce pendant un état non sûr ne conduit pas nécessairement à un interblocage.

## Algorithme du banquier

Vidéo d'explication de l'algorithme du banquier

Compléter les informations que l'on a

Au total pour pouvoir appliquer l'algorithme du banquier il nous faut :

- La matrice des ressources existantes (E)
- La matrice des besoins des processus (B)
- La matrice des allocations courantes (C)
- La matrice des ressources disponibles (A), qui correspond aux ressources existantes - les allocations courantes (E-C)
- La matrice des demandes des processus (R), qui correspond aux besoins des processus - les allocations courantes (B-C)

Les matrices que l'on va vraiment utiliser pour l'algorithme sont celles des allocations courantes (C), des demandes (R) et des ressources disponibles (A).

Vérifier si le système est dans un état sûr

- Pour chaque processus on va regarder si on peut remplir sa demande (R) à partir des ressources disponibles (A).
  - Si c'est possible, alors on marque le processus comme terminé et on ajoute aux ressources disponibles (A) toutes les allocations du processus terminé (C).
- On fait cela en boucle jusqu'à arriver à un résultat où tous les processus (C) sont terminés. Si à la fin tous les processus ne sont pas terminés, alors l'état n'est pas sûr.

Pour allouer depuis un état sûr

- Pour un processus qui demande une ressource, on va *hypothétiquement* diminuer les ressources disponibles de la demande, on va augmenter ses ressources allouées et diminuer ses besoins. C'est à dire que l'on va faire :
  - ressources disponibles -= demande
  - besoins -= demande
  - ressources allouées += demande
- On va ensuite effectuer l'algorithme précédent pour vérifier si ce système hypothétique est en état sûr, si c'est le cas, alors on peut allouer, sinon il faut attendre

## Détecter un interblocage

Le problème avec la première solution est que l'on ne sait pas en avance ce dont les processus ont besoins. Et il est plus efficace de simplement détecter et corriger un interblocage que d'empêcher un interblocage car les interblocages restent peu fréquents.

Cet algorithme de détection et de correction va se lancer lorsque le CPU n'est plus utilisé, ce qui signifie que les processus sont en état "waiting".

## Détection d'un cycle d'attente dans l'allocation

Pour détecter un interblocage il suffit de simplement connaître les ressources disponibles, les allocations courantes et les demandes actuelles.

Pour chaque processus en cours on va vérifier si ses demandes actuelles peuvent être satisfaites avec les ressources disponibles. Pour chaque processus trouvé, on va incrémenter les ressources disponibles des allocations courantes et on va définir le processus comme terminé.

Si à la fin il reste des demandes non satisfaites, il y a un interblocage.

## Correction d'un interblocage

Pour corriger un interblocage on va tuer un processus qui pose problème et tenter de maintenir les ressources dans un état cohérent.

On peut donc faire un rollback vers le contexte où le système était avant pour s'assurer que les ressources ne sont pas dans un état dégueulasse, du moins si on sauvegarde le contexte du processus régulièrement.

# La politique de l'autruche

Sur certains systèmes (tel que les systèmes UNIX), c'est à l'administrateur·ice de s'occuper de gérer un interblocage et le système d'exploitation s'en fout.

---

Revision #3

Created 24 October 2023 13:48:19 by SnowCode

Updated 2 January 2024 16:24:47 by SnowCode