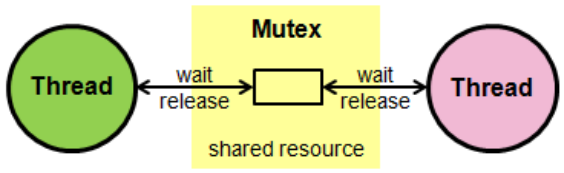
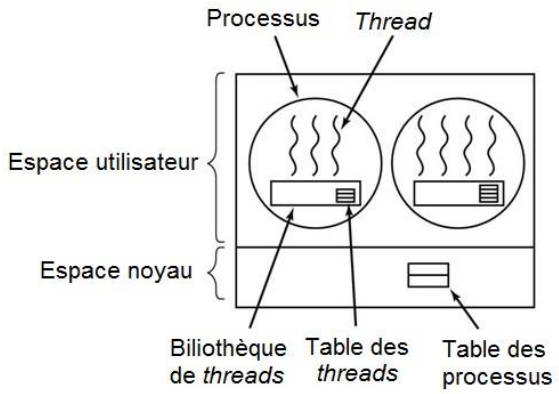




Extrait du référentiel : BTS Systèmes Numériques option A (Informatique et Réseaux)		Niveau(x)
S4. Développement logiciel S4.9. Programmation événementielle	Gestion des événements, signaux et interruptions	3
	Environnement multitâche : traitements parallèles (*thread*, sémaphores, tubes, …)	3
	Environnement temps réel : espace utilisateur, espace noyau, etc.	2

Objectifs du cours :

Ce cours traitera essentiellement les points suivants :

- Processus vs processus légers
- Synchronisation de tâches :
  - synchronisation de processus
  - synchronisation de données
  - définitions (section critique, exclusion mutuelle et chien de garde)
  - le mutex
  - le sémaphore
  - la variable condition et la barrière
  - les problèmes : inter blocage, famine et endormissement

PROCESSUS LOURS vs PROCESSUS LÉGERS

Le multitâche est moins coûteux avec les threads (processus légers) : puisqu'il n'y a pas de changement de mémoire virtuelle, la commutation de contexte (context switch) entre deux threads est moins coûteuse que la commutation de contexte obligatoire pour des processus lourds.

La communication entre threads est plus rapide et plus efficace : grâce au partage de certaines ressources entre threads, les IPC (Inter Processus Communication) sont inutiles pour les threads.

La programmation utilisant des threads est toutefois plus difficile : obligation de mettre en place des mécanismes de synchronisation, risques élevés d'inter blocage, de famine, d'endormissement.

SYNCHRONISATION DE TÂCHES

INTRODUCTION

Dans la programmation concurrente, le terme de synchronisation se réfère à deux concepts distincts (mais liés) :

La **synchronisation de processus** ou tâche : mécanisme qui vise à bloquer l'exécution des différents processus à des points précis de leur programme de manière à ce que tous les processus passent les étapes bloquantes au moment prévu par le programmeur.

La **synchronisation de données** : mécanisme qui vise à conserver la cohérence entre différentes données dans un environnement multitâche.

Les problèmes liés à la synchronisation rendent toujours la programmation plus difficile.

DÉFINITIONS

Section critique : c'est une partie de code telle que deux threads ne peuvent s'y trouver au même instant.

Exclusion mutuelle : une ressource est en exclusion mutuelle si seul un thread peut utiliser la ressource à un instant donné.

Chien de garde (watchdog) : un chien de garde est une technique logicielle (compteur, timeout, signal, ...) utilisée pour s'assurer qu'un programme ne reste pas bloqué à une étape particulière du traitement qu'il effectue. C'est une protection destinée généralement à redémarrer le système, si une action définie n'est pas exécutée dans un délai imparti.

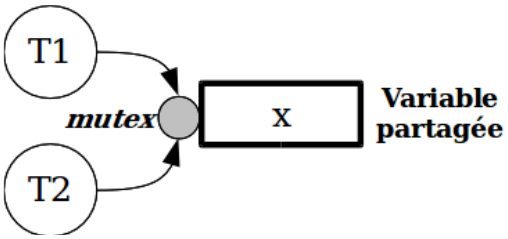
MÉCANISME DE SYNCHRONISATION : LE MUTEX

Rappel : la cohérence des données ou des ressources partagées entre les processus légers est maintenue par des mécanismes de synchronisation.

Il existe deux principaux mécanismes de synchronisation de données pour les threads : le **mutex** et le **sémaphore**.

Un mutex (verrou d'exclusion mutuelle) possède deux états : verrouillé ou non verrouillé.

Trois opérations sont associées à un mutex : lock pour verrouiller le mutex, unlock pour le déverrouiller et trylock (équivalent à lock, mais qui en cas d'échec ne bloque pas le thread).



MÉCANISME DE SYNCHRONISATION : LE SÉMAPHORE

Un sémaphore est un mécanisme empêchant deux processus ou plus d'accéder simultanément à une ressource partagée.

Un sémaphore général peut avoir un très grand nombre d'états car il s'agit d'un compteur dont la valeur initiale peut être assimilée au nombre de ressources disponibles.

Un sémaphore **ne peut jamais devenir négatif**.

Un sémaphore binaire, comme un mutex, n'a que deux états :
0=verrouillé (ou occupé) ou 1=déverrouillé (ou libre).

Un sémaphore bloquant est un sémaphore de synchronisation qui est initialisé à 0=verrouillé (ou occupé).

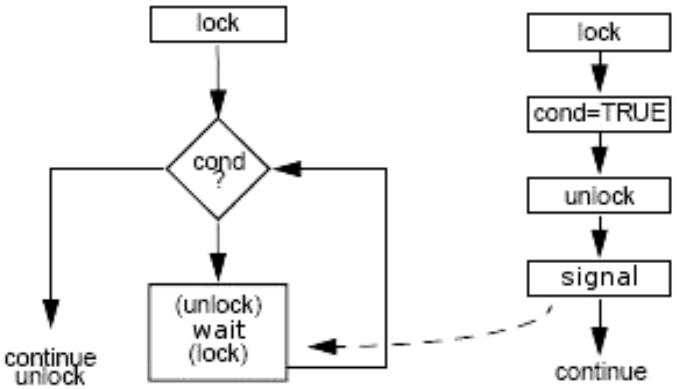
Trois opérations sont associées à un sémaphore : Init pour initialiser la valeur du sémaphore, P pour l'acquisition (Proberen, tester ou P(uis-je)) et V pour la libération (Verhogen, incrémenter ou V(as-y)).

MÉCANISME DE SYNCHRONISATION : LA VARIABLE CONDITION

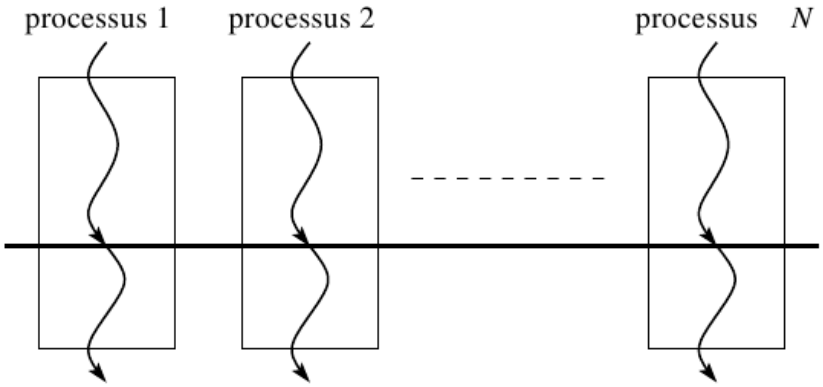
Il existe deux principaux mécanismes de synchronisation de processus pour les threads : la **variable condition** et la **barrière**.

Les variables conditions permettent de suspendre un fil d'exécution (thread) tant que des données partagées n'ont pas atteint un certain état.

Deux opérations sont disponibles : wait, qui bloque le processus léger tant que la condition est fausse et signal qui prévient les threads bloqués que la condition est vraie.



Une barrière de synchronisation (ou mécanisme de rendez-vous) permet de garantir qu'un certain nombre de tâches ait passé un point spécifique. Ainsi, chaque tâche qui arrivera sur cette barrière devra attendre jusqu'à ce que le nombre spécifié de tâches soient arrivées à cette barrière.



PROBLÈMES CONNUS

Les mécanismes de synchronisation peuvent conduire aux problèmes suivants :

Inter blocage (deadlocks) : le phénomène d'inter blocage est le problème le plus courant. L'inter blocage se produit lorsque deux threads concurrents s'attendent mutuellement. Les threads bloqués dans cet état le sont définitivement.

```
TâcheA :
  Obtenir M1
  Obtenir M2
  Action nécessitant les deux verrous
  Rendre M2
  Rendre M1
```

```
TâcheB :
  Obtenir M2
  Obtenir M1
  Action nécessitant les deux verrous
  Rendre M1
  Rendre M2
```

Famine (starvation) : un processus léger ne pouvant jamais accéder à un verrou se trouve dans une situation de famine. Cette situation se produit, lorsqu'un processus léger, prêt à être exécuté, est toujours devancé par un autre processus léger plus prioritaire.

Endormissement (dormancy) : cas d'un processus léger suspendu qui n'est jamais réveillé.

SYNTHÈSE

Section critique : section de code où jamais plus d'une tâche ne peut être active

Exclusion mutuelle : éviter que des ressources partagées ne soient utilisées en même temps par plusieurs tâches

Chien de garde (watchdog) : tâche de fond assurant la protection contre le blocage de tâches

Mutex : technique permettant de gérer un accès exclusif à des ressources partagées
Sémaphore : variable compteur permettant de restreindre l'accès à des ressources partagées