

# UE NSY103

## Communication & Synchronisation



# Sommaire

I – Introduction

II- Les Signaux

III- Les Mutex

IV- Les Sémaphores

V- L'Interblocage



# I- Introduction

**Sur un système multiprocessus Linux par exemple, de nombreux problèmes de synchronisation entre les processus peuvent se présenter.**

Pour réussir à se synchroniser les processus à mémoire privée peuvent utiliser les outils de communication interprocessus (Inter Processus Communication ou IPC) : tubes de communication et allocation de mémoire partagée.

- Les threads peuvent utiliser leur zone de mémoire commune pour communiquer.

Nous allons présenter dans ce document de nouveaux outils de synchronisation :

- Les signaux.
- les Mutex.
- les Sémaphores.

Nous allons également présenter dans ce document le concept d'interblocage



# II- Les Signaux

**Les Signaux vous ont été présentés dans le support de cours  
« Introduction au Noyau linux »**



# III- Les Mutex

## Le Problème des Réservations

Supposons que nous ayons besoin de développer une application de réservation de billets d'avion, pour cela nous allons devoir créer un compteur des places disponibles et décrémenter ce compteur à chaque réservation, cela tant qu'il reste des places :

```
Réservation {  
  Lire compteur  
  Si compteur > 0 alors  
    compteur--  
    écrire compteur  
    retour OK  
  Sinon  
    retour KO  
}
```



# III- Les Mutex

----- Le compteur est à 1 -----

```
P1 Réserve {  
Lire compteur          // 1
```

----- P1 préempté, P2 élu -----

```
P2 Réserve {  
Lire compteur          // 1  
décrémenter compteur  // 0  
réserver place
```

...

----- P2 préemptée, retour à P1 -----

```
Compteur > 0 alors  
    décrémenter compteur          // FAUTE Le compteur passe à -1  
    réserver place                // FAUTE déjà réservée  
}
```

- Dysfonctionnement du logiciel, on réserve 2 fois la dernière place et le compteur passe à -1.
- Le compteur (le mot mémoire qui le supporte) est une ressource exclusive, une seule unité d'exécution doit le manipuler (lecture + écriture) à la fois.

# III- Les Mutex

## Le Masquage des Interruptions

Nous pouvons interdire la préemption durant l'exécution de la réservation, ceci en masquant les interruptions :

```
Réservation {  
    CLI                // Masquage des interruptions  
    Lire compteur  
    Si compteur > 0 alors  
        compteur--  
        écrire compteur  
        STI // Démasquage des interruptions  
        retour OK  
Sinon  
    STI // Démasquage des interruptions  
    retour KO  
}
```

- Cependant le masquage des interruptions ne peut se faire qu'en mode noyau sur une machine Linux et donc les programmes en mode utilisateurs ne peuvent pas utiliser cette solution.

# III- Les Mutex

## Le Concept de Mutex

**Un Mutex (Mutual Exclusion) est un appel système qui va permettre de confier au noyau la mise en œuvre du masquage des interruptions pour gérer les accès aux ressources exclusives.**

- En fait on associe un Mutex à la ressource exclusive et les processus doivent respecter la règle du jeu : on n'accède à la ressource que si on est le propriétaire du Mutex :

```
/* ACCES A LA RESSOURCE */  
Prise de Mutex // Si le Mutex n'est pas libre le processus  
                // est bloqué en attente de la libération du Mutex.  
Manipulation de la Ressource //Lecture / Ecriture ...  
Libération du Mutex    // Le premier des éventuels  
                        // processus en attente  
                        // se voit affecter le Mutex
```

- Lors de la demande de prise du Mutex, le noyau masque les interruptions pour examiner l'état du Mutex et l'attribuer ou non au processus demandeur.
- Lors de la libération du Mutex, le noyau débloquent le premier processus en attente et lui attribue le Mutex.



# III- Les Mutex

## Applications Multiprocessus

- Les applications multiprocessus devraient utiliser les Sémaphores nommés à un jeton (équivalent du Mutex, cf. chapitre suivant) et de la mémoire partagée.

## Applications Multithreads

- Les applications multithread peuvent utiliser pour se synchroniser les Mutex `pthread_mutex_t` ou la mémoire commune de l'espace processus.

## La solution Mutex au Problème des Réservations

Cf. support d'Exercices

## IV- Les Sémaphores

Nous venons d'étudier les ressources à accès exclusif, cependant il existe aussi des ressources à accès multiples, telles que "n" processus - au plus - puissent y accéder simultanément.

# IV- Les Sémaphores

## Le Concept de Sémaphore

**Le concept de "Sémaphore" permet de gérer les ressources à accès multiples. Un sémaphore contient un certain nombre de jetons, chacun d'entre eux pouvant symboliser un accès à la ressource en question.**

Un sémaphore est géré par les 3 fonctions principales suivantes :

- La fonction d'initialisation qui fixe le nombre initial de jetons
- La fonction "P" (Probeer = Essayer en allemand) qui est utilisée pour retirer un jeton :
  - si le nombre de jetons disponibles est positif alors le jeton est attribué et le nbr. de jetons est décrémenté de 1
  - sinon le processus est bloqué et mis dans la file d'attente du sémaphore
- La fonction "V" (Verhoog = Incrémenter en allemand) est utilisée pour ajouter un jeton :
  - s'il y a des processus en file d'attente, on réveille le premier et on lui affecte le jeton qui vient d'être ajouté
  - sinon le nombre de jetons est incrémenté de 1

Un sémaphore à 1 jeton possède un fonctionnement analogue à celui du Mutex en considérant que :

- le nombre initial de jetons est de 1
- la prise de Mutex correspond à la fonction "P"
- la libération de Mutex correspond à la fonction "V"
- Il est conseillé d'utiliser des sémaphores à 1 jeton lorsque la prise et la libération du jeton s'effectuent dans des processus différents (certaines implémentations des Mutex se comportant de façon imprévisible dans ce cas).

# IV- Les Sémaphores

## Le Problème des Producteurs / Consommateurs

Il s'agit d'un mécanisme général de communication entre processus, où l'un est l'émetteur de messages (on lui donne le nom de *producteur*), et l'autre est le récepteur de ces messages (on lui donne le nom de *consommateur*).

- Pour ne pas ralentir le producteur en le faisant attendre jusqu'à ce que le consommateur ait fini de récupérer le message émis, on met en place un buffer tampon entre les deux processus, le buffer tampon peut contenir "**n**" messages.

Les deux processus doivent se synchroniser entre eux de façon à respecter certaines contraintes de bon fonctionnement :

- 1) Le producteur ne peut déposer un message dans le tampon s'il n'y a plus de place libre.
- 2) Le consommateur ne peut retirer un message depuis le tampon s'il est vide.
- 3) Le consommateur ne doit pas retirer un message que le producteur est en train de déposer (message incomplet).

Cf. « Le Problème des Producteurs / Consommateurs » dans le support d'exercices.



# IV- Les Sémaphores

## Applications Multiprocessus

- Les applications multiprocessus peuvent utiliser pour se synchroniser les sémaphores nommés Posix et de la mémoire partagée.

## Applications Multithread

- Les applications multithread peuvent utiliser pour se synchroniser les sémaphores Posix et les données communes de l'espace processus.

# V- L'Interblocage

L'Interblocage entre 2 processus peut survenir lorsque chacun de ces processus est en attente d'une ressource détenue par l'autre.

Un interblocage plus complexe peut survenir entre plusieurs processus chacun en attente de ressources détenues par un (ou plusieurs) des autres.



# V- L'Interblocage

Par exemple soit P1 et P2, deux processus qui utilisent les ressources RA, RB et RC :

- P1 a besoin de RA et RB simultanément
- P2 a besoin de RA, RB et RC simultanément

**P1 prends RA**

----- P1 préempté, P2 élu -----

**P2 prends RC**

**P2 prends RB**

**P2 essaie de prendre RA qui n'est pas disponible, il est alors mis en attente**

----- P2 mis en attente, P1 élu -----

**P1 essaie de prendre RB qui n'est pas disponible, il est alors mis en attente**

- Les 2 processus P1 et P2 se bloquent mutuellement, chacun d'eux dispose d'une ressource indispensable au déblocage de l'autre, ressource qu'il ne peut pas libérer étant lui-même en situation de blocage.

# V- L'Interblocage

Les interblocages se produisent inévitablement lorsque l'on n'établit pas de règles d'accès aux ressources critiques, par exemple dans le cas précédent on aurait pu établir une numérotation des ressources (RA = 1; RB = 2; RC = 3) et les processus auraient dû prendre ces ressources dans l'ordre de la numérotation.

P1 prend RA (n° 1)

----- P1 préempté, P2 élu -----

P2 essaye de prendre RA (n° 1) il est mis en attente

----- P2 mis en attente, P1 élu -----

P1 prend RB (n° 2)

P1 effectue ses traitements et libère RA et RB

----- P1 préempté, P2 élu -----

P2 prend RA (n° 1)

P2 prend RB (n° 2)

P2 prend RC (n° 3)

...



# V- L'Interblocage

## Exemple du Diner des Philosophes

Cf. le support d'exercices

## Exemple d'Interblocage Oracle

Cf. le support d'exercices

# V- L'Interblocage

## Cas du Noyau Linux

**Certains progiciels possèdent simplement des stratégies de détection des interblocages, d'autres possèdent des stratégies d'évitement des interblocages en mettant en attente certains processus lors de leurs demandes d'allocation de ressources en anticipation des interblocages.**

- Les stratégies d'évitement sont très couteuses en temps machine car elles nécessitent l'établissement et le suivi en temps réel de graphes de comportement des processus au niveau des allocations de ressources.
- Linux ne prend pas l'initiative de bloquer ou de supprimer les processus utilisateur en condition d'interblocage (ne pas intervenir au niveau applicatif, c'est la politique de la majorité des systèmes d'exploitation), par contre le noyau de Linux, qui est réentrant doit se protéger lui-même contre des possibilités d'interblocage internes.
- Pour cela le noyau Linux établit une numérotation ordinale des ressources partagées et s'impose d'y accéder selon cet ordre. Cette numérotation ordinale est basée sur les adresses en mémoire centrale des sémaphores qui gèrent ces ressources.



# EXERCICES

Vous pensez avoir bien assimilé les concepts présentés dans ce cours.  
Vous devez alors passer aux exercices, ce sont eux qui  
vous permettront de valider vos connaissances .