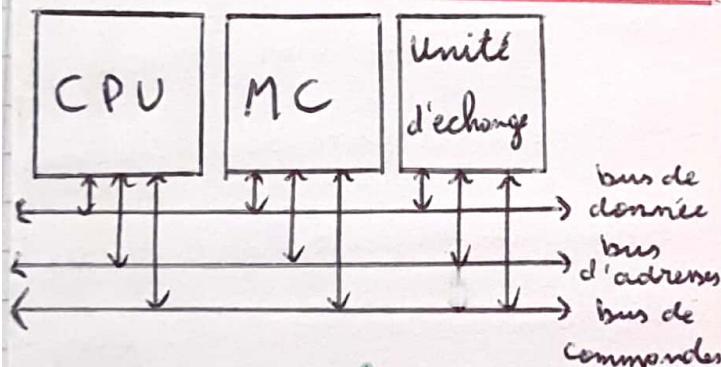


Chapitre 01 : Les processus

Système informatique :

est l'ensemble des composants matériels et logiciel dans une architecture à processeur

Architecture de Von Neuman :



Système d'exploitation :

Un ensemble de programmes agissant comme interface entre l'utilisateur et le matériel (machine virtuelle). Son but est de rendre le système informatique plus "pratique" à utiliser d'une manière "efficace" et "optimal".

Un programme :

Une suite ordonnée d'instructions.

"L'instruction" est l'élément

indivisible du programme, constituant la plus petite unité atomique du langage.

Processeur :

est un dispositif physique qui doit comprendre et exécuter un programme.

Etat de processeur :

L'état de processeur est caractérisé par :

- Le contenu stable de PSW
- Le contenu stable des registres programmable et les registres interne

Point observable (Interruptible) :

L'état du processeur ne peut être observable qu'au début ou à la fin de l'exécution d'une instruction.

Ressources :

On appelle une ressource tout élément contribuant à la progression de l'exécution d'un programme.

On distingue des ressources:

- Banalisées: existent en plusieurs copies "identiques" (espace mémoire, espace disque... etc)
- Non banalisées: des fichiers nommés, terminal... etc

Processus:

est un programme en cours d'exécution auquel on associe:

- Un Contexte du processeur (CPU): l'ensemble des registres.
- Un Contexte de MC: segments de code, segments de données...
- Un processus est une entité dynamique qui peut modifier l'état de son environnement par contre un programme est une entité statique.
- Un processus est dit séquentiel si l'exécution de ses instructions s'établit dans un ordre strict et bien déterminé autrement dit ne génère pas d'autres processus (Threads)

Operation sur les processus:

- Création d'un processus:
 - Un proces peut créer un autre proces.
 - La création d'un process implique la création de son PCB (Process Control Block)
 - Le processus père peut continuer l'exécution en parallèle de ses fils comme il peut être bloqué jusqu'à l'exécution du son fil.
- Destruction d'un processus:
 - Il y a 3 façon différent:
 - manière normal
 - instruction autodestructive (exit())
 - Un autre processus (souvent le père) peut terminer un proces (fil) exemple:
l'instruction Kill() sous Unix

Notion de tâche

Lorsqu'un processus présente un code un petit peu long ou complexe, il peut être décomposé en un ensemble d'unités de traitements élémentaires ayant une cohérence logique dont la fonction est bien déterminée. Cette unité est appelée une tâche.

Notion of Thread

Un Thread est une unité d'exécution légère d'un processus partageant avec lui des ressources comme le segment de code, les données, les fichiers ouverts.

Il permet d'exécuter plusieurs tâches simultanément pour une gestion plus efficace.

Chapitre 02: Processus Accès

concurrent et exclusion mutuelle

Interaction de processus

Processus dans le multi-tâches



indépendants
ils n'affectent pas l'un de l'autre. c-à-d l'ensemble de données et de ressources ne sont pas partagés

concurrents
des processus partagent les mêmes ressources

coopérants
l'un des processus peut être affecté pour l'exécution de l'autre. (des données partagées)
C-à-d : l'un des processus ne peut pas être exécuté jusqu'à l'exécution de l'autre

Section critique

est la partie d'un programme où la ressource partagée (critique) est manipulée

Exclusion Mutuelle :

C'est quand des processus demandent l'accès à une ressource critique (accès exclusif)

Problème de la Privation

(starvation) :

La situation où quelque processus progressent normalement en bloquant indéfiniment d'autres processus

Problème de l'interblocage

(deadlock) :

La situation où un ensemble de processus sont bloqués indéfiniment. Chacun en attente de ressources pour sa progression détenue par un autre proces dans l'ensemble

Propriétés de Dijkstra :

Toute solution apportée au problème EM doit respect les 4 propriétés suivantes :

1. Exclusion Mutuelle :

$$nb_process_dans_SC \leq 1$$

2. Absence d'interblocage :

L'existence d'interblocage peut être formulée comme suit

$$\exists t_0, \forall t > t_0 :$$

$$\begin{cases} nb_process_dans_SC = 0 \\ nb_attente > 0 \end{cases}$$

3. Progression :

Le blocage d'un proces hors de sa SC ne doit pas empêcher les autres proces d'entrer en SC

4. Absence de processus privilégiés :

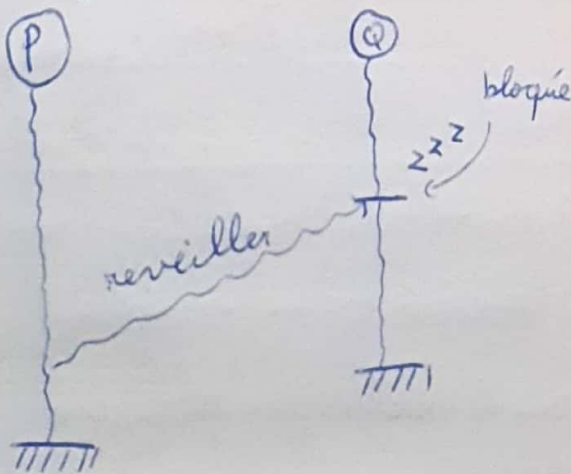
La solution doit être la même pour tous les proces.

Chapitre 3 : Outils de synchronisation de Processus.

Problème de synchronisation:

consiste à construire un mécanisme, indépendant de la vitesse d'exécution des processus, permettant à un processus actif P de :

- d'en bloquer un autre processus Q ou de se bloquer lui-même en attendant un signal d'un autre processus.
- d'activer ou d'éveiller le processus Q en lui transmettant un signal d'activation



Points de synchronisation:

Les contraintes de coopération entre deux processus peuvent être formulées selon deux formes:

1- Imposer un ordre de précedence logique dans le temps sur la trace d'exécution de certains points du code

2- Imposer aux processus une condition de franchissement de certains points de leurs traces temporelles

- ses points sont les points de synchronisation.

- Donc, les processus voulant se synchroniser doivent.

- * définir les points de synchronisation.

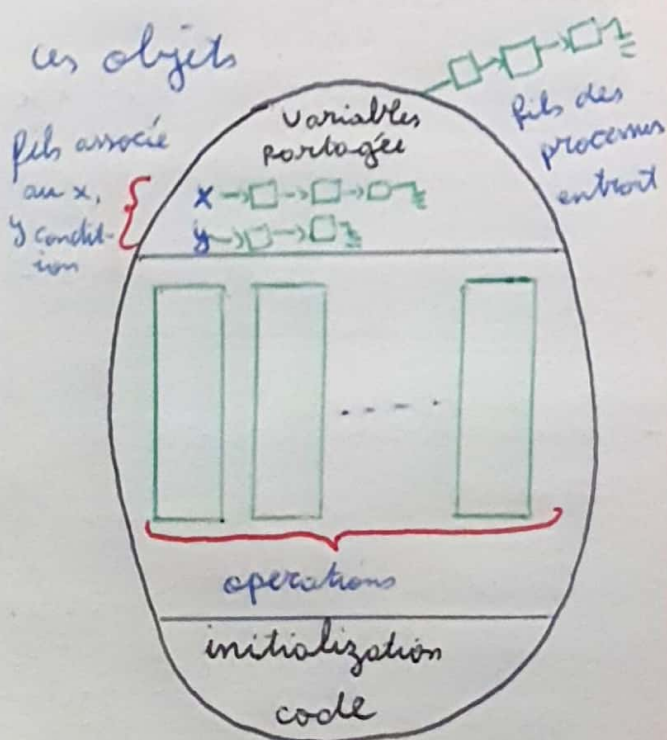
- * associer à chaque point les conditions de franchissement exprimées au moyen de variables d'état de synchronisation

Region Critique

est une structure linguistique de haut niveau comprenant une séquence d'instruction formant une section critique manipulant les variables partagées par plusieurs processus

Moniteur

est une structure linguistique de haut niveau semblable à une classe en POO encapsulant un ensemble d'objets partagés et des procédures définissant les traitements possibles sur ces objets



Chapitre 04: Interblocage

Définition:

Un sous ensemble S de P est dit en situation d'interblocage si tout processus de S est en attente d'un événement qui ne peut se produire que d'un autre processus de S et que

cette situation dure infiniment

Dans notre cas, l'événement attendu est la libération d'une ressource demandée.

Conditions nécessaires de l'interblocage:

une situation d'interblocage peut avoir lieu si les quatre conditions suivantes auront lieu "en même temps":

1. L'exclusion mutuelle

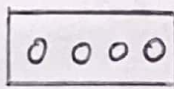
2. Allocation partielle: il existe au moins un processus qui détient au moins une


ressource et il est en attente d'autres ressources détenues par d'autres processus.

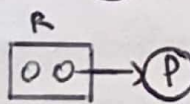
3. Non préemption: une ressource une fois allouée à un processus ne peut être lui retirée que s'il la libère volontairement

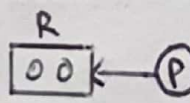
4. Attente circulaire

Graphe d'allocation de Ressources:

 Ressource contient n unités

 Process

 arc d'affectation: P détient une unité de R

 arc de requête: P est en attente de quelques unités de R

Remarque:

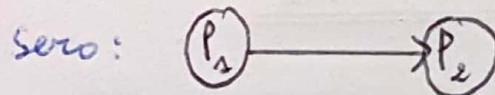
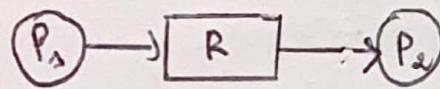
si un cycle existe et toutes les ressources impliquées dans le cycle sont critiques alors

il y a sûrement un interblocage.

Cycle + ressources critique \Rightarrow interblocage

Graphe des attentes:

si toutes les ressources existantes dans le système sont des ressources critiques, on peut éliminer les sommets des ressources et connecter directement les processus liés à lui



Représentation matricielle:

• R_{max} : array $[1..m]$: vecteur de toutes les ressources disponibles à l'état initial du système

$$R_{max} = [R_{1max}, \dots, R_{mmax}]$$

• Available: array $[1..m]$: vecteur de ressources disponible à un instant donné.

Available $[i]$: nbr unités disponibles de la ressource R_i

• **Allocate**: array $[n \times m]$:
Matrice des unités de ressource
allouées

$Allocate[i, j]$: le nbr d'unités
de ressource R_j allouées au
process P_i

• **Request**: array $[n \times m]$:
Matrice des demandes en
attente.

$Request[i, j]$: le nbr d'unités
de ressource R_j demandées
par le process P_i

Etat réalisable:

L'état de système est dit
réalisable si l'ensemble des
ressources disponibles et
celui des ressources allouées
sont dans un état cohérent

Contraintes de cohérence:

- 1) $Available \geq [0]$
- 2) $Request[i, *] \leq R_{max}$
- 3) $Allocate[i, *] \leq R_{max}$
- 4) $Allocate[i, *]_{t_1} \leq Allocate[i, *]_{t_0} + Request[i, *]_{t_0}^{t_1}$

$$5) \sum_{i=1, n} Allocate[i, *] \leq R_{max}$$

Traitement de l'interblocage:

1) Approche de l'Autriche:

Ignore de l'interblocage en
supposant qu'il aura pas lieu.
Adaptée aux système avec un
risque faible ou des application
non critiques

Avantages	Inconvénients
- Simplifie la gestion du système	- Les utilisateurs doivent gérer les interblocage
- Gain en temps d'exécution et en mémoire	- Peut entraîner des défaillances ou reinitialisation du système
	- Performances réduites

2) Approche Optimiste:

suppose que l'allocation
courante ne causera pas
d'interblocage. Détecte et
résout les interblocages si
nécessaire

bloccage

Avantages	Inconvénients
<ul style="list-style-type: none">- Permet une allocation flexible- Traite les inter-blocages après leur apparition	<ul style="list-style-type: none">- Les interblocages peuvent survenir- Coût supplémentaire pour la détection et la résolution

3) Approche Pessimiste :

Suppose qu'une allocation peut causer un interblocage et applique des mesures préventives pour l'éviter complètement

Avantages	Inconvénients
<ul style="list-style-type: none">- Evite totalement les interblocages- Garantit la stabilité du système	<ul style="list-style-type: none">- Surcoût dû à une surveillance constante- Performance utilisateur plus lente

Etat sain (fiable) :

Un état est dit sain si l'on peut allouer les ressources pour chaque processus dans un certain ordre tout en évitant l'inter-