Chapitre 04 : Inter-bocage, Synchronisation et Communication de Processus

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme Universitaire de Technique (DUT, 2^e année)
Diplôme Supérieure de Technologie (DST, 2^e année)
Option : **Informatique**

ECOLE SUPERIEURE POLYTECHNIQUE

www.esp.sn

Plan du Chapitre

Sommaire

Définitions

- ☛ Un ensemble de processus est en inter-blocage si chaque processus attend un événement que seul un autre processus de l'ensemble peut engendrer
- ☛ 2 (ou +) processus sont en attente de la libération d'une ressource attribuée à l'autre
- ☛ Un ensemble de processus est bloqué !

Vers un situation d'inter-blocage

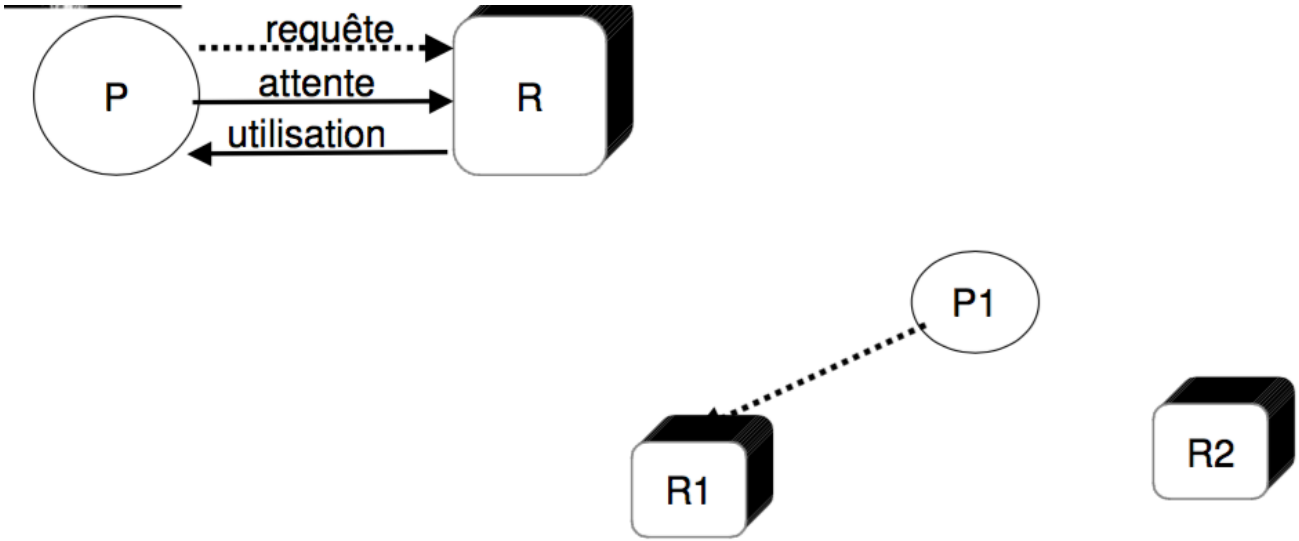


Figure: Vers un situation d'inter-blocage : 1

Vers un situation d'inter-blocage

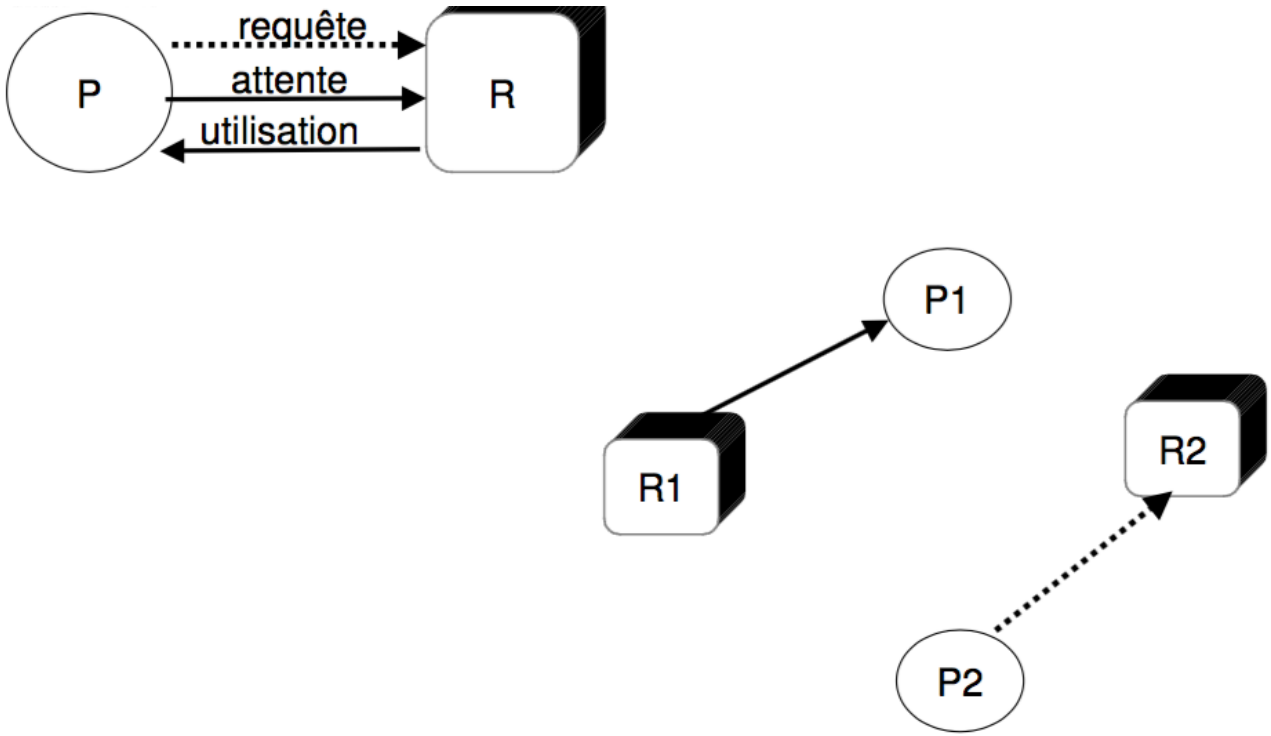


Figure: Vers un situation d'inter-blocage : 2

Vers un situation d'inter-blocage

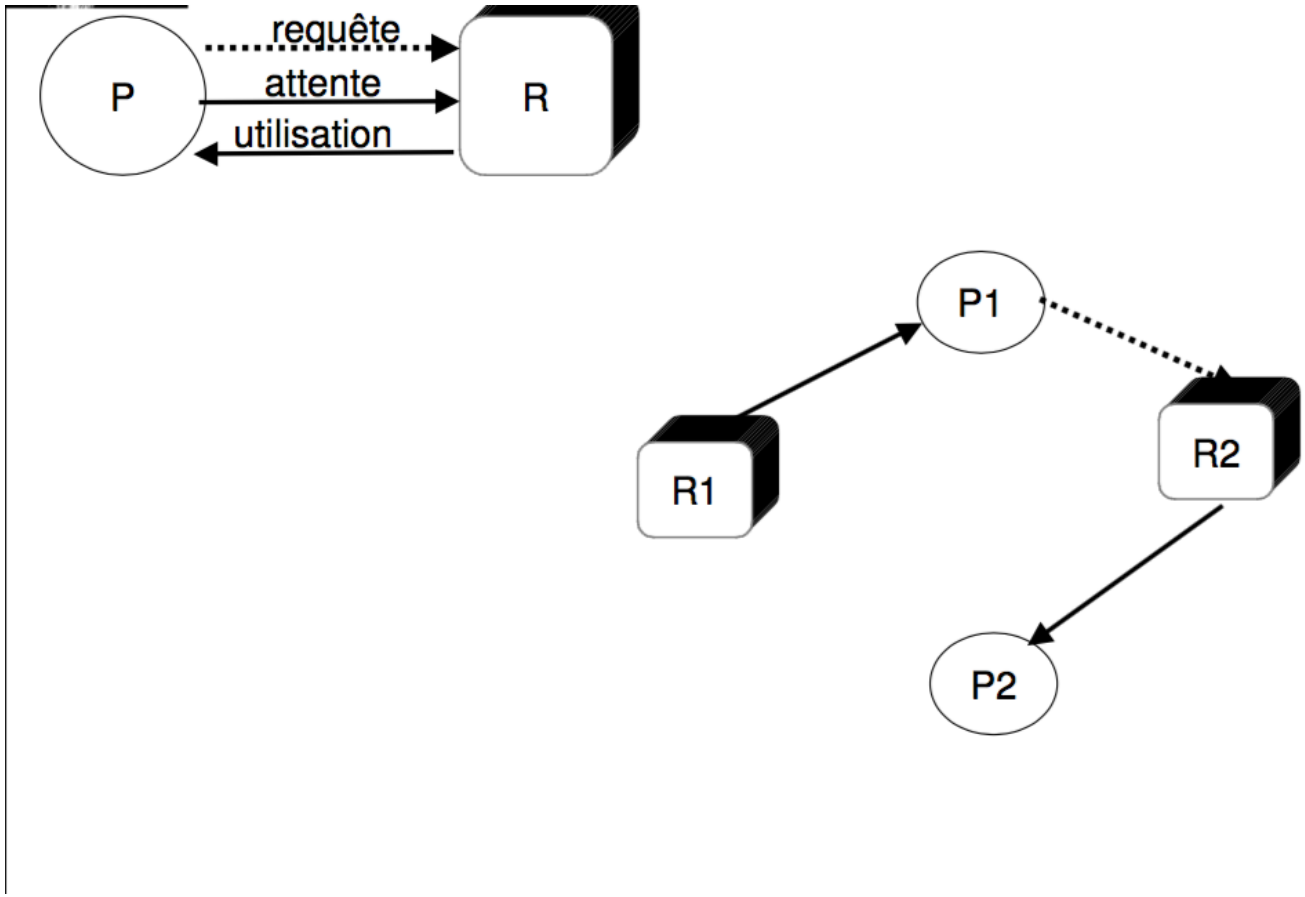


Figure: Vers un situation d'inter-blocage : 3

Vers un situation d'inter-blocage

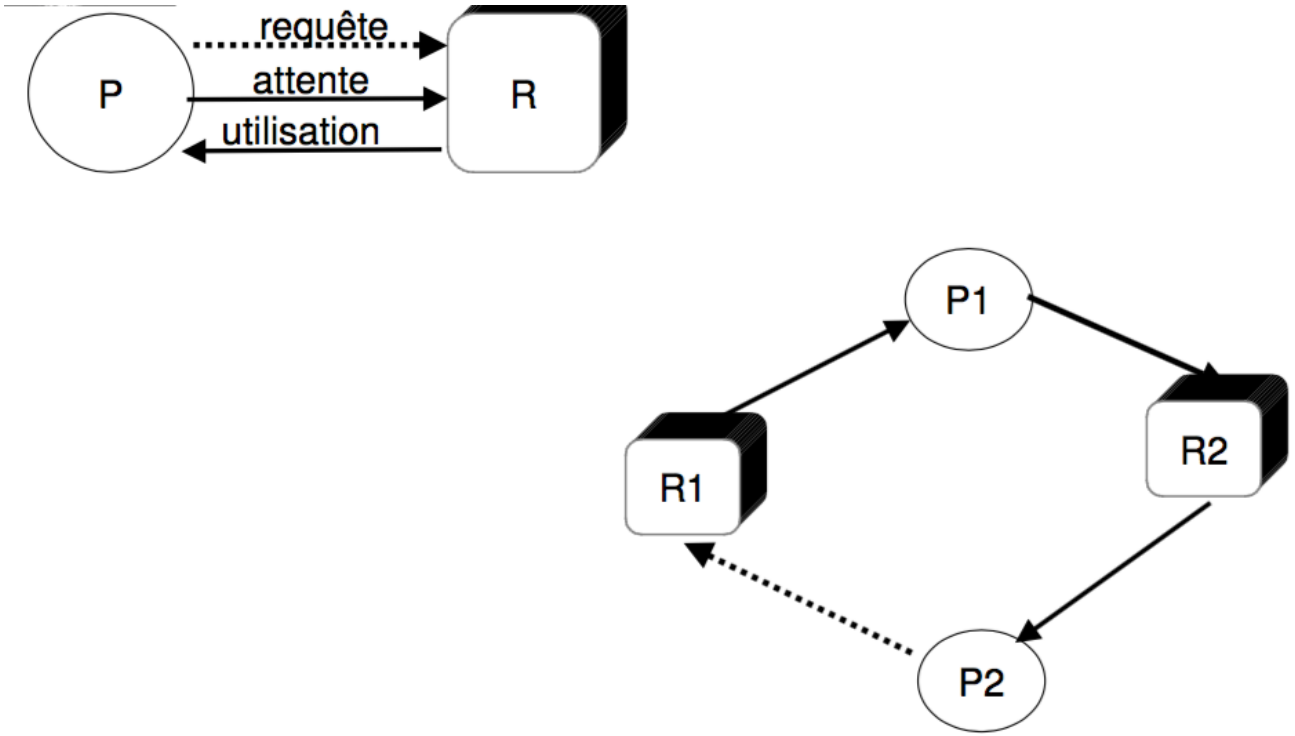


Figure: Vers un situation d'inter-blocage : 4

Vers un situation d'inter-blocage

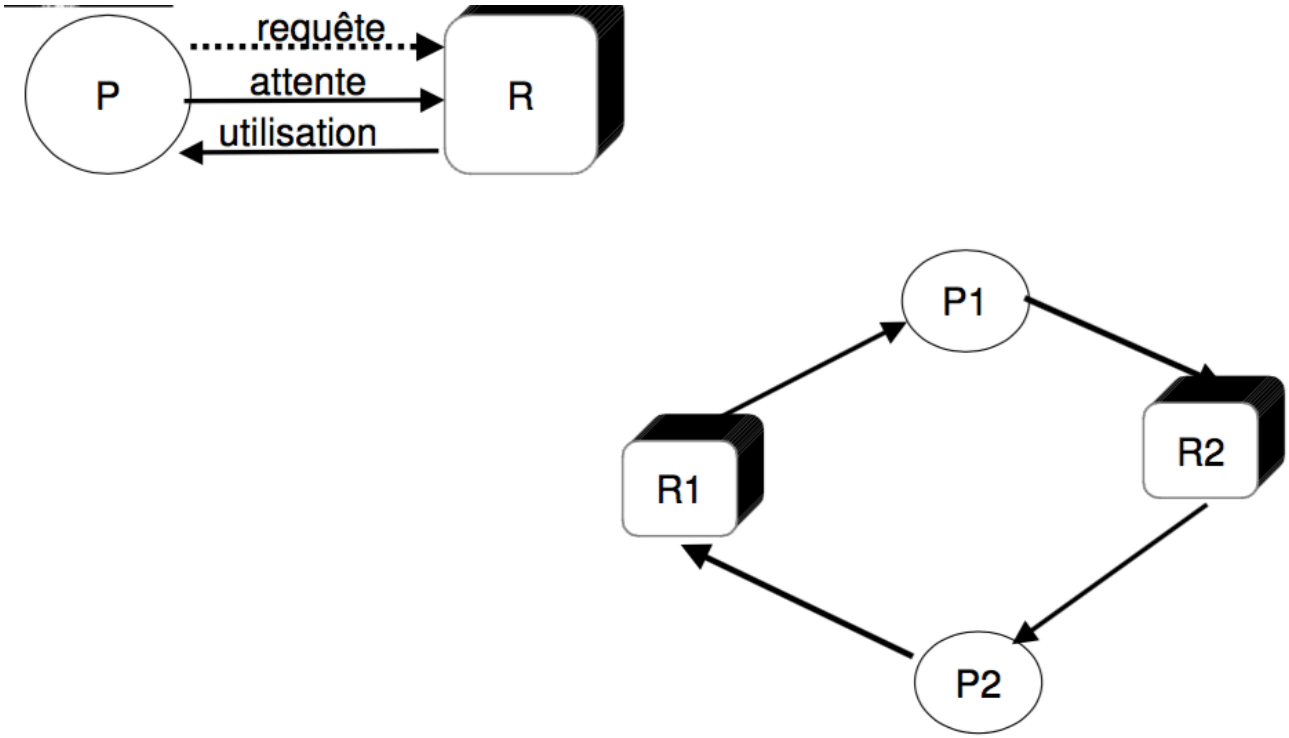


Figure: Vers un situation d'inter-blocage : 5

4 Conditions conduisant à un inter-blocage

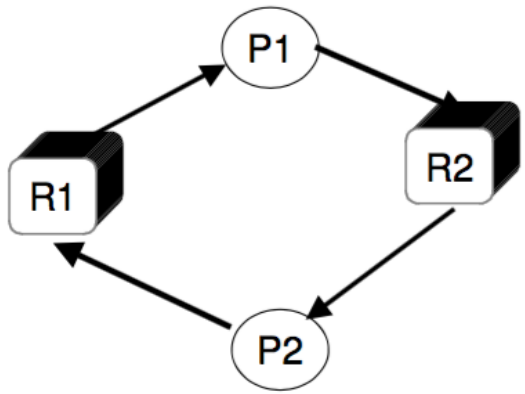
1. Ressource limitée (exclusion mutuelle)
2. Attente circulaire : il doit y avoir au moins deux processus chacun attendant une ressource détenu par un autre
3. Occupation et attente : les processus qui détiennent des ressources peuvent en demander de nouvelles
4. Pas de réquisition (non préemption) : les ressources sont libérées par le processus

Stratégies de gestion

1. Détecter et résoudre
2. **Prévenir la formation d'inter-blocages**
   - en empêchant l'apparition d'une des 4 conditions
3. Éviter les inter-blocages
   - en allouant de manière « intelligente » les ressources

Détecter et résoudre

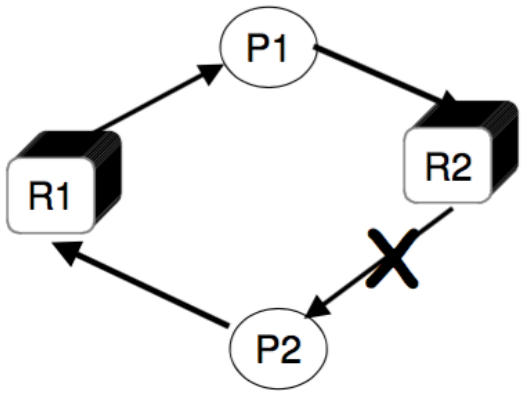
- Graphe de dépendance + détection des cycles



- Comment continuer? :

Détecter et résoudre

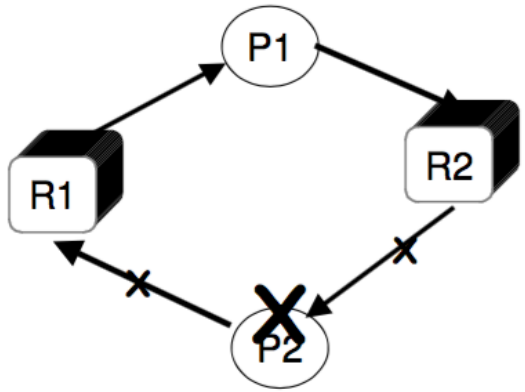
- Graphe de dépendance + Détection des cycles



- Comment continuer? :
  1. Retirer la ressource (La préemption) :
     - Certaines ressources, de par leur nature peuvent être retirées temporairement (parfois avec une intervention humaine).

Détecter et résoudre

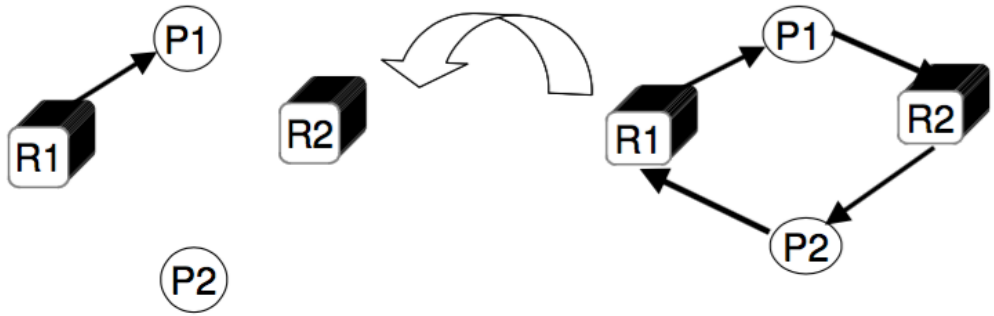
- Graphe de dépendance + Détection des cycles



- Comment continuer? :
  1. Retirer la ressource (La préemption)
  2. Suppression d'un processus :
     - on peut tuer un processus pris en inter-blocage afin de libérer ses ressources.
     - Un autre processus qui n'est pas dans le coup peut également être tué, si ça marche... ça marche

Détecter et résoudre

- Graphe de dépendance + Détection des cycles



- Comment continuer? :
  1. Retirer la ressource (La préemption)
  2. Suppression d'un processus
  3. Gestion de processus avec des points de reprise (rollback) :
     - à certains points clé, les états des processus sont enregistrés afin de pouvoir être restaurés ultérieurement.
     - Lors d'un inter-blocage, un processus est ramené à un point de reprise antérieur à l'acquisition de la ressource.

Prévenir l'inter-blocage

- **Exclusion mutuelle** :
  - empêcher un processus de s'accaparer une ressource (spooling)
- Attente circulaire :
  - les processus doivent demander les ressources dans un certain ordre, fixé par le SE.
- Occupation et attente :
  - forcer les processus à demander toutes leurs ressources au début
  - exécuter un processus *ssi* toutes les ressources sont disponibles
  - réquisition

Allouer « correctement »

- ☛ Méthode de l'allocation globale des ressources à une tâche
  - immobilisation non productive de ressources
- ☛ « Algorithme du banquier » :
  - L'algorithme du banquier utilise la métaphore d'un banquier d'une petite ville qui dispose d'un certain montant d'argent à prêter à ses clients.
  - Les clients sont des processus, l'argent correspond aux ressources et le banquier à l'OS.
  - L'algorithme vérifie si une requête mène à un état non sûr et la refuse en telle cas.
  - Sinon, tant que les requêtes laissent un état sûr derrière, elles sont allouées
    - Annonce des besoins
    - Déterminer à quel moment la ressource peut être alloué en sécurité

La vérité

- ☛ Le programmeur doit utiliser son cerveau et être responsable
- ☛ Organiser le système pour minimiser les risques :
  - Tous les appels systèmes susceptibles de bloquer sont implémentés avec des délais d'attente

Sommaire

Accès concurrents

- Le problème des accès concurrents se produit quand deux processus partagent une ressource, matérielle ou logicielle, alors que celle-ce ne peut pas être partagée (accès exclusif).
  - L'accès en lecture ne pose de problème contrairement aux accès en écriture

Exemple 1

- *P_1 et P_2 sont deux ressources voulant accéder à une imprimante*
- *L'imprimante est gérée par le processus deamon-plt qui inspecte :*
  1. *une variable de type tableau contenant les fichiers à imprimer*
  2. *et 2 variables entières prochain (qui indique numéro du prochain fichier à imprimer) et libre (qui indique le premier emplacement libre où déposer le fichier)*
- *Supposons le scénario suivant :*

Accès concurrents

Exemple 2

- *P_1 et P_2 sont deux ressources voulant accéder à une imprimante*
- *L'imprimante est gérée par le processus deamon-plt qui inspecte :*
  1. *une variable de type tableau contenant les fichiers à imprimer*
  2. *et 2 variables entières prochain (qui indique numéro du prochain fichier à imprimer) et libre (qui indique le premier emplacement libre où déposer le fichier)*
- *Supposons le scénario suivant :*
  1. *P_1 lit libre, la trouve à 5 puis est immédiatement interrompu*
  2. *P_2 lit libre, la trouve à 5 puis met son fichier à cet emplacement, incrémente libre à 6 et est ensuite interrompu*
  3. *P_1 reprends et écrase le fichier de P_2 par le sien*
- *Solution : ne pas interrompre P_1 au moment où il a été interrompu*

☛ *Notions de **section critique** et **exclusion mutuelle***

Section critique

Définition

- Une ressource est dite **ressource critique** lorsque des accès concurrents à cette ressources, il peut en résulter un état incohérent
- On parle aussi de situation de compétition (race condition) pour décrire une situation dont l'issue dépend de l'ordre dans lequel les opérations sont effectuées
- Une section critique est une section de programme manipulant une ressource critique.
- **Section critique** = partie du processus où il peut y avoir un conflit d'accès
  - Contient des variables et/ou ressources partagées

Exclusion mutuelle

Définition

☛ **Exclusion mutuelle** : si une ressource a été accédée par un processus P_1, aucun autre processus ne peut y accéder tant qu'elle n'a pas été libérée par P_1.

- Si P_1 est interrompu, il faut attendre à ce qu'il reprenne son exécution pour qu'il puisse libérer la ressource
- Une section de programme est dite atomique lorsqu'elle ne peut pas être interrompue par un autre processus manipulant les mêmes ressources critiques.
- C'est donc une atomicité relative à la ressource
- Un mécanisme d'exclusion mutuelle sert à assurer l'atomicité des sections critiques relatives à une ressource critique

Section critique et Exclusion mutuelle

Résolution des problèmes d'accès concurrents

1. **Les sémaphores** (au programme cette année)
2. Les moniteurs
3. Communication entre processus : primitives *send* et *receive*
4. Solutions matérielles

Sémaphores [E. W. Dijkstra, 1965] : présentation

Définition 1

- Un **sémaphore (S)** est un compteur entier qui désigne le nombre d'autorisations d'accès à une section critique
- Un sémaphore est une variable entière qui compte le nombre de processus en attente d'une section critique.
- Un sémaphore est une variable protégée (implémentation cachée), dont une "donnée-membre" est un compteur "NATURAL" (entier positif ou nul) initialisé à sa création à une valeur s_0 de type "NATURAL"
- Les sémaphores sont un outil élémentaire de synchronisation qui évitent l'attente active
- Le sémaphore est un objet de type abstrait. Il est caractérisé par la donnée d'un compteur et d'une file d'attente

Manipulation des Sémaphores

Deux types de sémaphores

1. **Sémaphore binaire** : peut prendre comme valeur 0 ou 1. Il est utilisé pour réaliser l'exclusion mutuelle
2. **Sémaphore n-aire** : peut prendre *n* valeurs. Il sert à spécifier le nombre d'accès maximal à une ressource (en lecture, évidement !)

Manipulation des Sémaphores

- Un sémaphore S =
  - un entier E(S) ;
  - une file d'attente F(S) ;
  - deux primitives P(S) et V(S).

- ☛ Manipulés au moyen des opérations **P** ou **wait** et **V** ou **signal**
- ☛ L'opération $P(S)$ décrémente la valeur du sémaphore S si cette dernière est supérieure à 0
- ☛ Sinon le processus appelant est mis en attente
- ☛ Le test du sémaphore, le changement de sa valeur et la mise en attente éventuelle sont effectués en une seule opération atomique indivisible
- ☛ L'opération $V(S)$ incrémente la valeur du sémaphore S, si aucun processus n'est bloqué par l'opération P(S)
- ☛ Sinon, l'un d'entre eux sera choisi et redeviendra prêt. V est aussi une opération indivisible

Manipulation des Sémaphores

☛ **Soit p le processus qui effectue P(S) ou V(S).**

P(S)

- E(S) = E(S) - 1;
  - Si E(S) < 0 alors
    - état(p) = bloqué;
    - entrer(p, F(S))

➠ Si le sémaphore est binaire, alors une seule exécution de la section critique

V(S)

- E(S) = E(s) + 1;
- si E(s) <= 0 alors
  - sortir(q, F(S));
  - état(q) = éligible;
  - entrer(q, F(éligibles));

Sémaphores d'exclusion Mutuelle

- But : protéger l'accès à une ressource unique (e.g. variable, imprimante, etc)
  - E(S) est initialisée à 1.
  - Utilisation :
    - P(S)
      < Section critique >
    - V(S)
  - Tous les processus doivent suivre la même règle.

Sémaphores de Synchronisation

☛ But : un processus doit en attendre un autre pour continuer (ou commencer) son exécution.

- ➠ E(S) est initialisée à 0.
- ➠ Utilisation :

Processus 1

- 1er travail
- V(s) // réveil processus 2

Processus 2

- P(s) // attente processus 1
- 2ème travail

Sommaire

Introduction

- Les processus coopèrent souvent pour traiter un même problème
- Ces processus s'exécutent en parallèle sur un même ordinateur (mono-processeur ou multiprocesseurs) ou bien sur des ordinateurs différents.
- Ils doivent alors s'échanger des informations : **communication inter-processus**
- Il existe plusieurs moyens de communication inter-processus :
  1. les signaux
  2. les messages :
     - les tubes de communication
     - les sockets

les signaux

- Les **interruptions logicielles** ou **signaux** sont utilisées par le système d'exploitation pour aviser les processus utilisateurs de l'occurrence d'un événement important.
- De nombreuses erreurs détectées par le matériel, comme l'exécution d'une instruction non autorisée (une division par 0, par exemple) ou l'emploi d'une adresse non valide, sont converties en signaux qui sont envoyés au processus fautif.
- Ce mécanisme permet à un processus de réagir à cet événement sans être obligé d'en tester en permanence l'arrivée.
- Les processus peuvent indiquer au système ce qui doit se passer à la réception d'un signal

les signaux

- On peut ainsi ignorer le signal, ou bien le prendre en compte, ou encore laisser le SE appliquer le comportement par défaut :
  - en général tuer le processus
- Certains signaux ne peuvent être ni ignorés, ni capturés
- Si un processus choisit de prendre en compte les signaux qu'il recroit, il doit alors spécifier la procédure de gestion de signal.
- Quand un signal arrive, la procédure associée est exécutée.
- A la fin de l'exécution de la procédure, le processus s'exécute à partir de instruction qui suit celle durant laquelle l'interruption a eu lieu

Synthèse

- Les signaux sont utilisés pour établir une communication minimale entre processus, une communication avec le monde extérieure et faire la gestion des erreurs.

les signaux

Décision prise à l'arrivée d'un signal

- Tous les signaux ont une routine de service, ou une action par défaut. Cette action peut être du type :
  - terminaison du processus
  - ignorer le signal
  - Créer un fichier *core*
  - Stopper le processus
  - La procédure de service ne peut pas être modifiée
  - Le signal ne peut pas être ignoré

Les messages

- Un autre mécanisme de communication entre processus est l'échange de **messages**
- Chaque message véhicule des données
- Un processus peut envoyer un message à un autre processus se trouvant sur la même machine ou sur des machines différentes
- Unix offre plusieurs mécanismes de communication pour l'envoie de messages : les tubes de communication sans nom, nommés et les sockets

Présentation générale

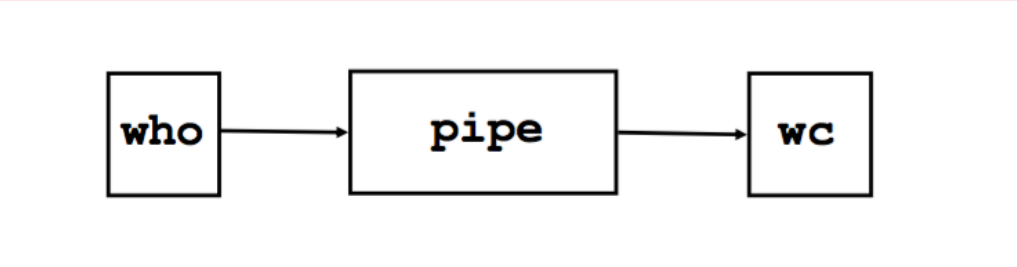
- Les tubes de communication ou pipes permettent à deux ou plusieurs processus d'échanger des informations.
- On distingue deux types de tubes :
  1. **tubes sans nom** (unamed pipe)
  2. **tubes nommés** (named pipe)

Les tubes sans nom

- Les tubes sans nom sont des liaisons de communication unidirectionnelles.
- La taille maximale d'un tube sans nom varie d'une version à une autre d'Unix, mais elle est approximativement égale à 4 Ko
- Les tubes sans nom peuvent être créés par le shell ou par l'appel système pipe()
- Les tubes sans nom du shell sont crées par l'opérateur binaire « | » :
  - la sortie standard d'un processus vers l'entrée standard d'un autre processus
  - Les tubes de communication du shell sont supportés par toutes les versions d'Unix

Les tubes sans nom : exemple

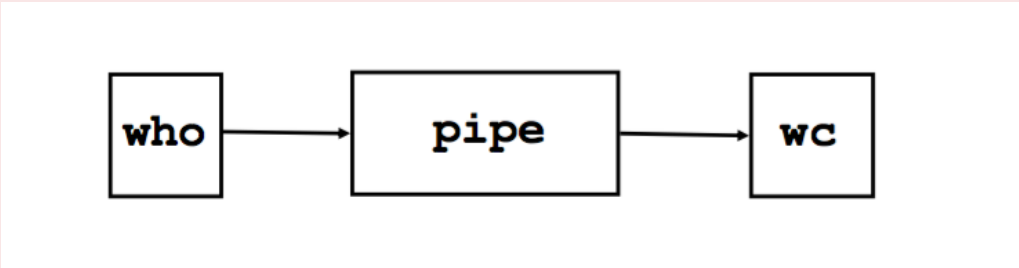
- Soit la commande suivante :
  - **who | wc -l**
- Elle crée deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe



1. Elle détermine le nombre d'utilisateurs connectés au système en appelant who, puis en comptant les lignes avec wc
2. Le premier processus réalise la commande who. Le second processus exécute la commande wc -l

Les tubes sans nom : exemple

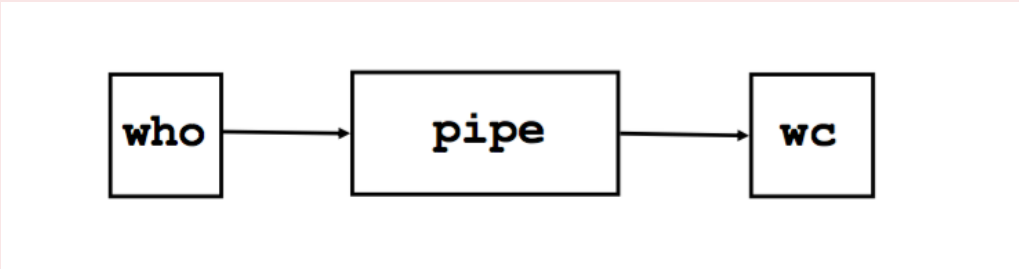
- Soit la commande suivante :
  - **who | wc -l**
- Elle crée deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe



4. Le processus réalisant la commande **who** ajoute dans le tube une ligne d"information par utilisateur du système
5. Le processus réalisant la commande **wc -l** récupère ces lignes d'information pour en calculer le nombre total
6. Le résultat est affiché à l?écran

Les tubes sans nom : exemple

- Soit la commande suivante :
  - **who | wc -l**
- Elle crée deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe

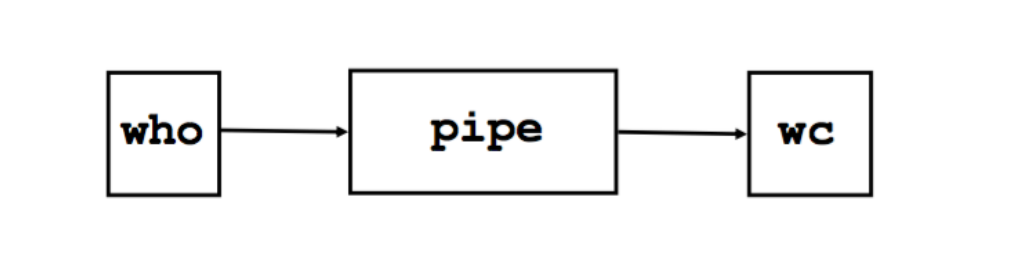


7 Les deux processus s'exécutent en parallèle, les sorties du premier processus sont stockées sur le tube de communication.

8 Lorsque le tube devient plein, le premier processus est sus- pendu jusqu'à ce qu'il y ait libération de l'espace nécessaire pour stocker une ligne d'information

Les tubes sans nom : exemple

- Soit la commande suivante :
  - **who | wc -l**
- Elle crée deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe



8 Lorsque le tube devient plein, le premier processus est sus- pendu jusqu'à ce qu'il y ait libération de l'espace nécessaire pour stocker une ligne d'information

9 De même, lorsque le tube devient vide, le second processus est suspendu jusqu'à ce qu'il y ait au moins une ligne d'information sur le tube.

Tubes de communication nommés

- Linux supporte un autre type de tubes de communication, beaucoup plus performants
- Il s'agit des tubes nommés (named pipes).
- Les tubes de communication nommés fonctionnent aussi comme des files du type FIFO (First In First Out)
- Ils sont plus intéressants que les tubes sans nom car ils offrent, en plus, les avantages suivants :
  1. Ils ont chacun un nom qui existe dans le système de fichiers (une entrée dans la Table des fichiers)
  2. Ils sont considérés comme des fichiers spéciaux.
  3. Ils peuvent être utilisés par des processus indépendants, à condition qu'ils s'exécutent sur une même machine
  4. Ils existeront jusqu'à ce qu'ils soient supprimés explicitement
  5. Ils sont de capacité plus grande, d'environ 40 Ko.

Sockets

- Les tubes de communication permettent d'établir une communication unidirectionnelle entre deux processus d'une même machine
- Le système Unix/Linux permet la communication entre processus s'exécutant sur des machines différentes au moyen de **sockets**.
- Les sockets ou prises sont les mécanismes traditionnels de communication inter-processus du système Unix
- Elles permettent la communication entre processus s'exécutant sur des machines différentes connectées par un réseau de communication.
- Par exemple, l'utilitaire **rlogin** qui permet à un utilisateur d'une machine de se connecter à une autre machine, est réalisé au moyen de sockets
- Les sockets sont également utilisé pour imprimer un fichier se trouvant sur une autre machine et pour transférer un fichier d'une machine à autre.

Sommaire

Les Appels systèmes

- Un appel système est un moyen de communiquer directement avec le noyau de la machine
- Le noyau regroupe toutes les opérations vitales de la machine
- Ainsi il est impossible d'écrire directement sur le disque dur
- L'utilisateur doit passer par des appels systèmes qui contrôlent les actions qu'il fait
- Ceci permet de garantir :
  1. la sécurité des données car le noyau interdira à un utilisateur d'ouvrir les fichiers auxquels il n'a pas accès
  2. l'intégrité des données sur le disque. Un utilisateur ne peut pas par mégarde effacer un secteur du disque ou modifier son contenu
- Ainsi, les appels système sont les fonctions permettant la communication avec le noyau
  - open, read, write, etc.

Les Appels système : utilisation

- Travaillent en relation directe avec le noyau
- Retournent un entier positif ou nul en cas de succès et -1 en cas d'échec
- Par défaut le noyau peut bloquer les appels systèmes et ainsi bloquer l'application si la fonctionnalité demandée ne peut pas être servie immédiatement
- Ne resservent pas de la mémoire dans le noyau.
- Les résultats sont obligatoirement stockés dans l'espace du processus (dans l'espace utilisateur)
- il faut prévoir cet espace par allocation de variable (statique, pile) ou de mémoire (malloc(). . .)

Chapitre 04 : Inter-bocage, Synchronisation et Communication de Processus

Dr Mandicou BA
mandicou.ba@esp.sn
http://www.mandicouba.net

Diplôme Universitaire de Technique (DUT, 2^e année)
Diplôme Supérieure de Technologie (DST, 2^e année)
Option : **Informatique**

ECOLE SUPERIEURE POLYTECHNIQUE
www.esp.sn