Systèmes et Réseaux – MIAGE 2
Systèmes d'Exploitation

Gestion de processus

Sara Bouchenak

Sara.Bouchenak@imag.fr
http://sardes.inrialpes.fr/~bouchena/teaching

Exemple introductif : Opérations bancaires

solde initial = 1000

Programme 1	Programme 2
1. courant1 = lire_compte (1867A)	**1. courant2 = lire_compte (1867A)**
2. nouveau1 = courant + 1000	**2. nouveau2 = courant + 3000**
3. ecrire_compte (1867A, nouveau1)	**3. ecrire_compte (1867A, nouveau2)**
Exécuté par processus p1	**Exécuté par processus p2**

Exécution des opérations bancaires

- Les processus p1 et p2 sont lancés depuis deux agences différentes
- Les deux processus se déroulent en parallèle
- L'exécution des opérations peut être entrelacée dans un ordre quelconque, à condition de respecter l'ordre local pour chacun des processus

1er exemple d'exécution

solde initial = 1000

Processus p1	Processus p2
1. courant1 = lire_compte (1867A)	**1. courant2 = lire_compte (1867A)**
2. nouveau1 = courant + 1000	**2. nouveau2 = courant + 3000**
3. ecrire_compte (1867A, nouveau1)	**3. ecrire_compte (1867A, nouveau2)**

- p1.1 ; p1.2 ; p1.3 ; p2.1 ; p2.2 ; p2.3
- Quel est le solde résultant ?

↳ *5000*

2ème exemple d'exécution

solde initial = 1000

Processus p1	Processus p2
1. courant1 = lire_compte (1867A)	1. courant2 = lire_compte (1867A)
2. nouveau1 = courant + 1000	2. nouveau2 = courant + 3000
3. ecrire_compte (1867A, nouveau1)	3. ecrire_compte (1867A, nouveau2)

- p1.1 ; p2.1 ; p1.2 ; p2.2 ; p1.3; p2.3
- Quel est le solde résultant ?

↳ *4000*

Problème …

- Quel est le problème ?
- Que peut-on conclure ?

↳ *Problème d'accès concurrent à une variable partagée*

↳*Pour l'éviter,* assurer que l'ensemble des opérations sur cette variable (consultation + mise à jour) est exécuté de manière indivisible (atomique)

Solution

	Processus p1		Processus p2
A1	1. courant1 = lire_compte (1867A) 2. nouveau1 = courant + 1000 3. ecrire_compte (1867A, nouveau1)	A2	1. courant2 = lire_compte (1867A) 2. nouveau2 = courant + 3000 3. ecrire_compte (1867A, nouveau2)

- Si A1 et A2 sont atomiques, le résultat de l'exécution parallèle de A1 et A2 ne peut être que celui de <A1 ; A2> ou de <A2 ; A1>, à l'exclusion de tout autre
- On dit aussi que la séquence d'actions <1; 2; 3> (dans p1 et dans p2) est une section critique
- Une section critique est exécutée en exclusion mutuelle (un seul processus au plus peut être dans sa section critique à un instant donné)

Plan

Relations entre processus : Compétition vs. coopération

- Compétition
  - Situation dans laquelle plusieurs processus doivent utiliser simultanément une ressource à accès exclusif
  - La ressource ne pouvant être utilisée que par un seul processus à la fois
  - Exemples
    - processeur (cas du pseudo-parallélisme)
    - imprimante
  - Une solution possible (mais non la seule) : faire attendre les processus demandeurs jusqu'à ce que l'occupant actuel ait fini (premier arrivé, premier servi)

Relations entre processus : Compétition vs. coopération

- Coopération
  - Situation dans laquelle plusieurs processus collaborent à une tâche commune
  - Les processus doivent se synchroniser pour réaliser cette tâche
  - Exemples
    - p1 produit un fichier, p2 imprime le fichier
    - p1 met à jour un fichier, p2 consulte le fichier
  - La synchronisation se ramène au cas suivant : un processus doit attendre qu'un autre processus ait franchi un certain point de son exécution

Faire attendre un processus

- Dans les deux types de relations (compétition ou coopération), on est conduit à faire attendre un processus.
- Comment réaliser cette attente ?

- Solution 1 : attente active

p1

```
while (ressource occupée)
  { };
ressource occupée = true;
utiliser ressource;
ressource occupée = false;
```

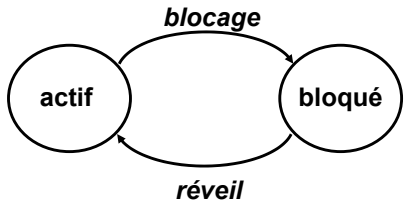
p2

```
while (ressource occupée)
  { };
ressource occupée = true;
utiliser ressource;
ressource occupée = false;
```

  - très peu économique si pseudo-parallélisme
  - difficulté d'une solution correcte (à voir plus tard)

Faire attendre un processus

- Solution 2 : blocage du processus
  - On définit un nouvel état pour les processus, l'état bloqué.
  - L'exécution d'un processus bloqué est arrêtée, jusqu'à son réveil explicite par un autre processus



Plan

1. *Introduction aux systèmes d'exploitation*
2. *Processus*
3. **Gestion des processus**
   - *Relations entre processus*
   - Section critique
   - Interblocage
   - Coopération entre processus
4. Fichiers
5. Mémoire
6. Etudes de cas

Réalisation d'une section critique

- Schéma général

déclaration et initialisation de variables communes, partagées

processus p1	**processus p2**
...	...
entrée en section critique	**entrée en section critique**
section critique	**section critique**
sortie de section critique	**sortie de section critique**
...	...

- Les opérations "entrée en section critique", "sortie de section critique" doivent garantir l'exclusion mutuelle

Modes de réalisation d'une section critique

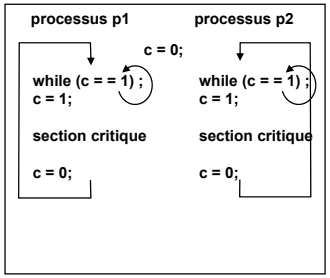
- Il existe plusieurs modes de réalisation d'une section critique
  - En utilisant des primitives spéciales
  - Par attente active

Section critique en utilisant des primitives spéciales

- En utilisant des primitives spéciales
  - fournies par le système
  - elles-mêmes atomiques
  - verrous (voir plus loin)
- Exemple : Comment assurer qu'un seul système Netscape est actif

```
/* lancer une session Netscape */
if ((lock_descr = create("~/.netscape/lock", 0)) == -1) (
        /* afficher message d'erreur */
        ...
} else {
        /* lancer navigateur */
}
...
/* terminer session */
close (lock_descr); unlink ("~/.netscape/lock");
```

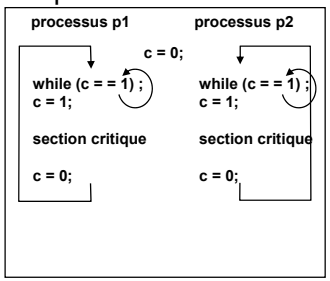
Section critique par attente active

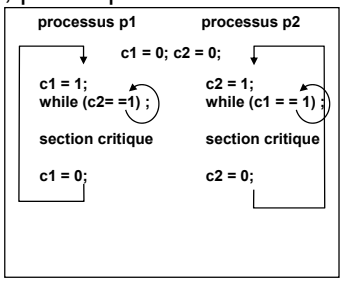


- Caractéristiques
  - très inefficace s'il y a un seul processeur
  - utilisée pour des séquences brèves en multiprocesseur

Réalisation d'une section critique par attente active (1)

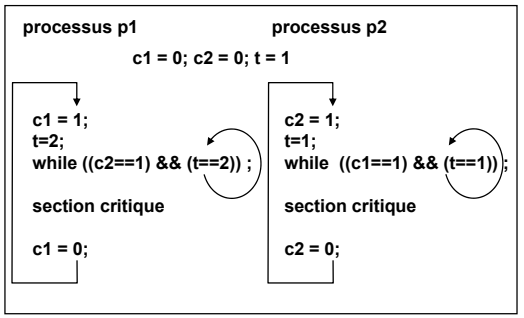
- Réaliser l'exclusion mutuelle par attente active est plus difficile qu'il n'y paraît …
- Exemples de "fausse solution", pour 2 processus





Réalisation d'une section critique par attente active (2)

- Une solution correcte pour l'exclusion mutuelle par attente active pour 2 processus (*Peterson, 1981*)



Opérations de verrouillage

- Les opérations de verrouillage sont une manière de réaliser l'exclusion mutuelle, pour une opération particulière
- Deux opérations de base
  - Verrouillage avec accès exclusif
  - Déverrouillage

Exemple : Verrouillage de fichiers

- Exemple d'opération nécessitant un verrouillage
  - Accès à un fichier
- Deux opérations
  - *v-excl (f)* : verrouille le fichier f avec accès exclusif
  - *dev (f)* : déverrouille le fichier f

  Remarque :
  ces noms sont symboliques,
  la réalisation en Unix est donnée plus loin

Verrouillage des fichiers (suite)

- Rappel des opérations de verrouillage
  - les processus p1 et p2 partagent un fichier f, dans lequel ils écrivent
  - chaque séquence d'accès à f est une section critique (l'accès à f est exclusif)

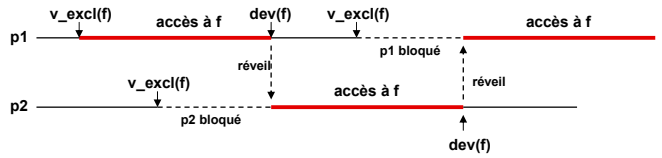
processus p1

```
...
v-excl (f)
accès au fichier f (section critique)
dev (f)
...
```

processus p2

```
...
v-excl (f)
accès au fichier f (section critique)
dev (f)
...
```


Verrouillage exclusif



Verrouillage partagé

- Problème
  - Le verrouillage exclusif n'est pas toujours nécessaire
- Exemple
  - Les processus p1 et p2 lisent le fichier f (sans le modifier)
  - Le processus p3 écrit dans le fichier
- Besoins
  - Lectures multiples :
    - p1 et p2 peuvent effectuer leurs opérations de lecture en même temps
  - Ecriture exclusive :
    - si p3 écrit dans f, pas de lecture ni d'autre écriture en parallèle

Verrouillage partagé : nouveau mode de verrouillage

- Verrouillage en accès partagé
  - un fichier f verrouillé en accès partagé par un processus ne peut pas être verrouillé par un autre processus en mode exclusif, mais peut être verrouillé en mode partagé
- Verrouillé en accès exclusif
  - un fichier f verrouillé en accès exclusif par un processus ne peut pas être verrouillé par un autre processus (en mode exclusif ou partagé)
- une opération de verrouillage bloque le processus qui l'exécute si l'une des règles ci-dessus s'applique
- l'opération de déverrouillage réveille un processus en attente du verrou (et un seul)
- Exemple
  - on peut permettre l'accès simultané à f de p1 et p2 (mais non p1 et p3, ou p2 et p3)

Verrouillage partagé : Exercice

- Enoncé
  - En utilisant v-excl, v-part et dev,
  - Programmer l'accès d'un ensemble de processus à un fichier,
  - Processus "lecteurs" ne font que lire le fichier
  - Processus "rédacteurs" peuvent lire et modifier le fichier.
  - Écrire le programme des lecteurs et le programme des rédacteurs
- Indications : utiliser des compteurs pour compter
  - le nombre de lecteurs actifs
  - et le nombre de lecteurs en attente

Verrouillage de fichiers dans Unix

- Caractéristiques
  - On peut verrouiller un fichier entier ou seulement une partie (permet d'augmenter le parallélisme)
  - On peut aussi tester si le fichier est déjà verrouillé (opération non bloquante)
- Opérations disponibles
  - Deux primitives (appels système) sont utilisables
    - fcntl : fonction très générale, permet tout type de verrouillage
    - lockf : fonction plus restreinte, verrouillage exclusif

Utilisation simultanée de plusieurs fichiers (1)

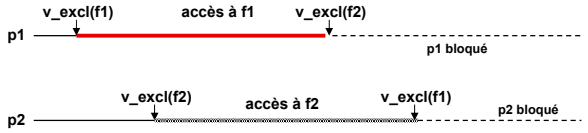
- Les processus p1 et p2 partagent deux fichiers f1 et f2
- p1 et p2 utilisent les fichiers en accès exclusif

processus p1	processus p2
...	...
(1a) v-excl (f1)	(2a) v-excl (f2)
accès à f1	accès à f2
...	...
(1b) v-excl (f2)	(2b) v-excl (f1)
accès à f1 et f2	accès à f1 et f2
dev (f2)	dev (f1)
dev (f1)	dev (f2)

Utilisation simultanée de plusieurs fichiers (2)

- 1ère exécution possible : 1a ; 1b ; 2a ; 2b ; ...
- 2ème exécution possible : 1a ; 2a ; 1b ; 2b ; ...
  Que se passe-t-il ?

processus p1	processus p2
...	...
(1a) v-excl (f1)	(2a) v-excl (f2)
accès à f1	accès à f2
...	...
(1b) v-excl (f2)	(2b) v-excl (f1)
accès à f1 et f2	accès à f1 et f2
dev (f2) dev (f1)	dev (f1) dev (f2)



Utilisation simultanée de plusieurs fichiers (3)

- Exécution dans l'ordre 1a ; 2a ; 1b ; 2b ; ...
- Situation après le deuxième v_excl(f1)
  - p1 et p2 sont bloqués, et le resteront indéfiniment
  - pour réveiller p1, il faut faire dev(f2) qui ne peut être fait que par p2, bloqué
  - pour réveiller p2, il faut faire dev(f1) qui ne peut être fait que par p1, bloqué

↳ **Interblocage** (*deadlock*)

Plan

1. *Introduction aux systèmes d'exploitation*
2. *Processus*
3. **Gestion des processus**
   - ***Relations entre processus***
   - ***Section critique***
   - **Interblocage**
   - **Coopération entre processus**
4. Fichiers
5. Mémoire
6. Etudes de cas

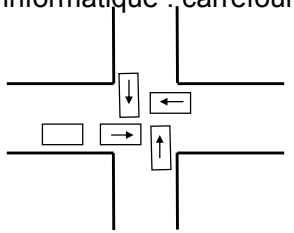
Interblocage : caractérisation

- Définition de l'interblocage
  - Situation dans laquelle plusieurs processus (au moins 2) sont bloqués
  - Chacun des processus ne peut être réveillé que par une action de l'un des autres processus
  - On ne peut pas sortir d'une situation d'interblocage sans intervention extérieure

Interblocage : Conditions d'apparition

- l'interblocage se produit lorsque plusieurs processus sont en compétition pour utiliser simultanément plusieurs ressources
- et lorsque les demandes de ces ressources sont mal coordonnées (attente circulaire)

- Un exemple hors informatique : carrefour



Remarque : cas où seule la marche avant est possible

Comment prévenir l'interblocage ? (1)

- Solution 1 : réservation globale
  - Principe
    - chaque processus demande globalement (en bloc) toutes les ressources dont il a besoin
  - Inconvénient
    - réduit les possibilités de parallélisme
  - Exemple
    - solution adoptée pour les carrefours (le parallélisme n'est pas l'exigence première)

Comment prévenir l'interblocage ? (2)

- Solution 2 : requêtes ordonnées
  - L'interblocage est impossible si tous les processus demandent les ressources dont ils ont besoin dans le même ordre
  - Exemple

processus p1	processus p2
...	...
v-excl (f1)	v-excl (f1)
accès à f1	accès à f1
...	...
v-excl (f2)	v-excl (f2)
accès à f1 et f2	accès à f1 et f2
dev (f2)	dev (f1)
...	...
dev (f1)	dev (f2)

Lutter contre l'interblocage (1)

- On ne peut pas sortir d'une situation d'interblocage sans *perdre quelque chose*

- Possibilités de "guérison"
  - Faire revenir un ou plusieurs processus en arrière, dans un état antérieur (on perd le travail réalisé depuis cet état)
    - nécessite d'avoir conservé l'état antérieur
  - Tuer un ou plusieurs processus pour libérer les ressources

Lutter contre l'interblocage (2)

- Conclusion
  - Pour choisir entre prévention et guérison, il faut apprécier les coûts relatifs de l'une et de l'autre
    - coût de la prévention : application d'une politique systématique de réservation de ressources
    - coût de la guérison : détection de l'interblocage + pertes résultant du retour en arrière

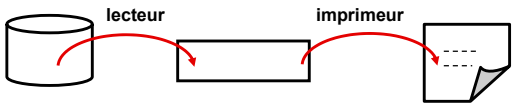
Plan

Compétition vs. coopération entre processus

- Jusqu'ici, on n'a vu que des situations de compétition entre processus
  - pour l'utilisation du processeur (pseudo-parallélisme et ordonnancement des processus)
  - pour l'accès à un fichier (accès exclusif ou non)
- Coopération entre processus …

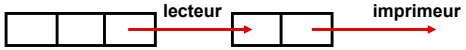
Exemple de coopération entre processus (1)

- Un exemple de situation de coopération
  - un processus *lecteur* lit un fichier depuis le disque vers la mémoire centrale
  - un processus *imprimeur* récupère le fichier dans la mémoire centrale et l'envoie à l'imprimante



Exemple de coopération entre processus (2)

- Contraintes
  - on veut exécuter le lecteur et l'imprimeur en parallèle
  - la place en mémoire est restreinte
- Solution
  - un "tampon" (ensemble de n "cases") ; ici n = 2, et le fichier a une taille de 3 cases



Coopération entre processus

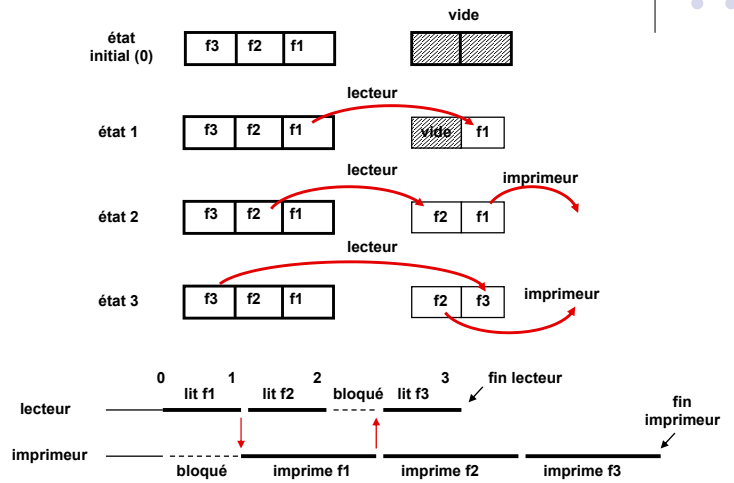
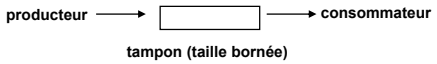


Schéma producteur – consommateur (1)

- Le schéma lecteur -> tampon -> imprimeur est un exemple du schéma général de coopération dit "producteur - consommateur"



- Utilité
  - On cherche à augmenter le parallélisme entre producteur et consommateur, malgré une différence possible de leurs vitesses d'exécution
  - La taille du tampon est d'autant plus grande que la différence de vitesses est grande

Schéma producteur – consommateur (2)

- Fonctionnement
  - condition de blocage du producteur : le tampon est plein (on ne peut pas y déposer de l'information sans "écraser" de l'information encore non consommée)
  - condition de blocage du consommateur : le tampon est vide
  - en fait, le fonctionnement est symétrique (le consommateur est un producteur de cases vides)
- Applications
  - entrées-sorties tamponnées (*spool*)
  - traitements en pipe-line, cf. les tubes d'Unix (*pipes*)

Plan

1. *Introduction aux systèmes d'exploitation*
2. *Processus*
3. **Gestion des processus**
   - ***Relations entre processus***
   - ***Section critique***
   - ***Interblocage***
   - ***Coopération entre processus***
4. Fichiers
5. Mémoire
6. Etudes de cas

Références

- **Systèmes d'exploitation – 2ème édition**, A. Tanenbaum, Pearson Education, 2003.
- **Practical UNIX Programming**, K. A. Robbins, S. Robbins, Prentice Hall, 1996
- **Principe des systèmes d'exploitation des ordinateurs**, S. Krakowiak, Dunod, 1985.
- Ce cours a été conçu à partir d'autres supports :
  - Sacha Krakowiak, http://sardes.inrialpes.fr/people/krakowia/
  - Fabienne Boyer, http://sardes.inrialpes.fr/people/boyer/cours/SR/