



Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel

Sémaphores

Moniteurs

Applications

# Systèmes d'Exploitation

## Synchronisation des processus

Didier Verna

didier@lrde.epita.fr  
<http://www.lrde.epita.fr/~didier>



# Table des matières

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel

Sémaphores

Moniteurs

Applications

- 1 Théorie
- 2 Synchronisation matérielle
- 3 Synchronisation logicielle
  - Sémaphores
  - Moniteurs
- 4 Applications classiques



# Nécessité de la synchronisation

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

## ■ Problématiques

- ▶ Communication (IPC, pipes *etc.*) : gestion des dépendances, séquençage
- ▶ Concurrence d'accès aux données : risque de corruption

## ■ Remarques

- ▶ Problématique n° 2 identique au niveau intra-processus (threads)
- ▶ temps partagé  $\Rightarrow$  concurrence au niveau système



# Mise en évidence

## Problème du « buffer limité »

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

### ■ Description

- ▶ Stockage d'information dans un tableau de taille fixe
- ▶ Un « producteur » d'information remplit le tableau
- ▶ Un « consommateur » d'information le vide
- ▶ Exemple typique : le pipe d'UNIX
- ▶ Schéma de type producteur / consommateur

### ■ Problèmes

- ▶ **Synchronisation** : garantir l'atomicité de certaines opérations
- ▶ **Niveaux d'atomicité** : matérielle, logicielle, logique



# Notion de section Critique

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

- Un seul processus à la fois peut exécuter sa section critique (exclusion mutuelle).
- Un processus candidat ne doit pas être bloqué par un processus non demandeur.
- Un processus ne doit pas attendre indéfiniment de pouvoir rentrer dans sa section critique.

**Repeat**

*Section d'entrée*

***Section critique***

*Section de sortie*

***Section restante***

**Until false**



# Algorithme du boulanger

Solution correcte pour N processus

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

- (1) enchoix [ i ]  $\leftarrow$  vrai.
- (2) numéro [ i ]  $\leftarrow$  max (numéro [ x  $\neq$  i ]) + 1.
- (3) enchoix [ i ]  $\leftarrow$  faux.
- (4) **pour** j  $\leftarrow$  0  $\rightarrow$  N - 1:
- (5)     **tant que** enchoix [ j ]:
- (6)         Ne rien faire.
- (7)     **tant que** numéro [ j ]  $\neq$  0 **et** P<sub>j</sub> < P<sub>i</sub>:
- (8)         Ne rien faire.

- (1) numéro [ i ]  $\leftarrow$  0.



# Solutions matérielles

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

- **Interdire les interruptions** pendant une section critique
  - ▶ Outil dangereux aux mains des utilisateurs
  - ▶ Ne marche pas sur des systèmes multiprocesseurs
  - ▶ Impossible pour des horloges mises à jour par interruption
- **Disposer d'instructions matérielles** donc atomiques

**Test-And-Set (TAS) :**

```
ret = X;  
X = true;  
return ret;
```

**Swap :**

```
tmp = a;  
a = b;  
b = tmp;
```



# Solution avec TAS

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

- |  |   |
|--|---|
| (1) attends [ i ] $\leftarrow$ vrai.             | (1) $j \leftarrow i + 1 \bmod N$ .                        |
| (2) clé $\leftarrow$ vrai.                       | (2) <b>tant que</b> $j \neq i$ <b>et</b> ! attends [ j ]: |
| (3) <b>tant que</b> attends [ i ] <b>et</b> clé: | (3) $j \leftarrow j + 1 \bmod N$ .                        |
| (4) clé $\leftarrow$ TAS (verrou).               | (4) <b>si</b> $i = j$ :                                   |
| (5) attends [ i ] $\leftarrow$ faux.             | (5) verrou $\leftarrow$ faux.                             |
|  | (6) <b>sinon</b> :  |
|  | (7) attends [ j ] = faux.                                 |

## ■ Problèmes

- ▶ **Attente active** (« busy waiting »)  
**Verrou actif** (« spin lock »)
- ▶ **Inversion de priorités** : un processus prioritaire bloqué derrière un processus standard





# Sémaphores

Dijkstra (1965)  $\implies$  Algol 68

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel

Sémaphores

Moniteurs

Applications

- Extension du schéma `sleep / wakeup` :  
mémoriser le nombre de réveils en attente
- Compteur associé à une file d'attente de processus

**P / wait** (*proberen*)

tant que `S.cnt <= 0`, attendre;  
`S.cnt -= 1;`

```
S.cnt -= 1;  
if (S.cnt < 0)  
    push (P, S.fifo);
```

**V / signal** (*verhogen*)

`S.cnt += 1;`

```
S.cnt += 1;  
if (S.cnt <= 0)  
    pop (S.fifo);
```

- Le système d'exploitation garanti l'atomicité du point de vue des processus utilisateurs.
- La file d'attente n'a pas besoin d'être FIFO, tant qu'une stratégie correcte d'ordonnancement est utilisée.



# Moniteurs

Hoare (1974) / Hansen (1975)

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel  
Sémaphores  
Moniteurs

Applications

## ■ Principe

- ▶ Module de procédures et variables
- ▶ Accès externe aux variables interdit
- ▶ Exclusion mutuelle sur les procédures

## ■ Implémentation

- ▶ Construction langagière
- ▶ Le compilateur garanti l'exclusion mutuelle
- ▶ Exemple : Java + classes + `synchronized`

## ■ Outil annexe « variable conditionnelle »

- ▶ Opérations `wait` et `signal`
- ▶ Pas de mémorisation (`signal` peut être sans effet)



# Solution avec moniteur

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel

Sémaphores

Moniteurs

Applications

**ProdCons** → **produire**

- (1) **si**  $C = N$ :
- (2)     **wait** (room)
- (3)      $\text{buf}[\text{in}] \leftarrow \text{elt}$
- (4)      $\text{in} \leftarrow \text{in} + 1 \bmod N$
- (5)      $C \leftarrow C + 1$
- (6)     **si**  $C = 1$ :
- (7)         **signal** (info)

**ProdCons** → **consommer**

- (1) **si**  $C = 0$ :
- (2)     **wait** (info)
- (3)      $\text{elt} \leftarrow \text{buf}[\text{out}]$
- (4)      $\text{out} \leftarrow \text{out} + 1 \bmod N$
- (5)      $C \leftarrow C - 1$
- (6)     **si**  $C = N - 1$ :
- (7)         **signal** (room)



# Applications classiques

Systèmes  
d'Exploitation

Didier Verna  
EPITA

Théorie

Matériel

Logiciel

Sémaphores  
Moniteurs

Applications

- Problème du buffer limité
- Implémentation des sémaphores de comptage à partir de sémaphores binaires
- Problème des lecteurs / rédacteurs :
  - ▶ Premier problème : priorité aux lecteurs
  - ▶ Deuxième problème : priorité aux rédacteurs

Note : risque de famine

- Problème du dîner des philosophes
- Problème du barbier endormi
- *etc.*