# Travaux Dirigés n°5

## Cours de Parallélisme

*—Maîtrise Informatique—*

---

Ce TD continue sur le problème de la section critique en considérant le cas général de $n$ processus concurrents. Nous voyons deux types d'algorithme qui correspondent respectivement à des solutions dites en centralisé et en distribué.

---

Les algorithmes présentés lors des séances précédentes concernaient le problème de la section critique dans le cas de deux processus. Ces algorithmes étaient tous *centralisés*. Cela signifie que les processus impliqués utilisaient plusieurs variables matérialisant un certain état du système de processus et parmi ces variables, au moins une était accessible à la fois en lecture et écriture par l'ensemble des processus. Ce type de variable est appelé *variable partagée*.

▸ **Exercice 1.** (l'algorithme de Eisenberg et McGuire, 1972) *Dijkstra a généralisé en 1965 la solution de Dekker au cas de $n$ processus. Cette solution assure l'exclusion mutuelle, l'absence d'interblocage mais n'exclut pas la famine. La première solution équitable a été donnée par Knuth en 1966. Elle a subit des modifications pour réduire la borne exponentielle de l'attente maximale qu'elle induisait. Ces modifications conduisent à la version de Eisenberg et McGuire où* `n` *processus partagent les variables suivantes :*

```
typedef enum {passif, demande, en_sc} Bool;
Bool drapeau[n]={passif, ..., passif};
int tour=0;
```

*Le* `i`*-ième processus exécute l'algorithme suivant et utilise une variable locale* `j` *de type entier :*

```
for (;;)
{ do
  { drapeau[i]=demande;
    j=tour;
    while (j!=i) if (drapeau[j]!=passif) j=tour; else j=(j+1)%n;
    drapeau[i]=en_sc;
    j=0;
    while ((j<n) && ((j==i) or (drapeau[j]!=en_sc))) j++;
  }
  while ((j>=n) && ((tour==i) or (drapeau[tour]==passif)));
  tour=i;
  section_critique();
  j=(tour+1)%n;
  while ((j!=tour) && (drapeau[j]==passif)) j=(j+1)%n;
  tour=j;
  drapeau[i]=passif;
}
```

1. *Quelle propriété (exclusion mutuelle, atteignabilité de la section critique, absence de famine, etc.) se déduit du fait que la variable* `tour` *n'est modifiée que par un processus entrant ou quittant sa section critique.*

2. *Prouver l'absence de famine et donner la borne maximale du nombre de tours d'attente pour entrer en section critique.*

3. *Prouver l'exclusion mutuelle de la section critique.*

Face aux algorithmes résolvant le problème de la section critique en centralisé, se trouvent les algorithmes en *distribué*, qui n'utilisent aucune variable partagée.

Lamport a été le premier en 1974 à donner un algorithme ne s'appuyant sur aucun dispositif centralisé. Cet "algorithme de la boulangerie" s'appuie sur le principe très simple (que l'on trouve dans certains magasins) selon lequel chaque client reçoit un ticket en entrant qui permet de servir les clients dans leur ordre d'arrivée. Mais puisque l'on ne veut pas dans cet algorithme se servir d'une information centralisée, chacun des processus choisit son propre numéro en fonction des numéros pris par les autres et si deux processus ont pris des numéros identiques, un protocole (arbitraire) résoud le conflit.

**Exercice 2.** (l'algorithme de Lamport, dit "de la boulangerie") *On considère* `n` *processus partageant en lecture les variables suivantes :*

```
typedef enum {faux, vrai} Bool;
Bool choix[n]={faux, ..., faux};
int num[n]={0, ..., 0};
```

*On considère l'algorithme suivant exécuté par chaque processus* $P_{\texttt{i}}$ *pour* $\texttt{i} = 0, \ldots, \texttt{n} - 1$ *:*

```
choix[i]=vrai;
num[i]=1+max(num[0], ..., num[n-1]);
choix[i]=faux;
for(j=0;j<n;j++)
{ if(j!=i)
  { while (choix[j]==vrai);
    while ((num[j]!=0) && ((num[j],j)<(num[i],i)));
  }
}
section_critique();
num[i]=0;
section_non_critique();
```

*L'ordre sur les couples correspond à l'ordre* $(a, b) < (c, d)$ *si* $a < c$ *ou si* $a = c$ *et* $b < d$*.*

1. *Montrer que si la variable* `choix` *n'est pas utilisée, l'algorithme est faux, c'est-à-dire n'interdit pas la situation où deux processus seraient en section critique en même temps.*

2. *Montrer que l'algorithme satisfait bien l'exclusion mutuelle, l'atteignabilité de la section critique et l'attente bornée.*