

# Le modèle Producteur - Consommateur

- [Définition](#)
- [Solution à une case](#)
- [Solution à  \$n\$  cases](#)
- [Solution à  \$p\$  producteurs et  \$c\$  consommateurs](#)

## Définition

Le producteur et le consommateur sont deux processus **cycliques**.

Producteur	Consommateur
...	...
<i>produire(messageP);</i>	<i>retirer(case, messageC);</i>
<i>déposer(case, messageP);</i>	<i>consommer(messageC);</i>
...	...

**Problèmes** : déposer un message alors que le consommateur n'a pas retiré le précédent ou retirer un message alors que le producteur n'a rien déposé.

## Solution à une case

On utilise 2 sémaphores *plein* et *vide* initialisé à **0** et **1**.

*plein* indique si la case est pleine et *vide* ...

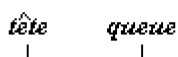
Producteur	Consommateur
<b>P(vide)</b>	<b>P(plein)</b>
<i>produire(messageP);</i>	<i>retirer(case, messageC);</i>
<i>déposer(case, messageP);</i>	<i>consommer(messageC);</i>
<b>V(plein)</b>	<b>V(vide)</b>

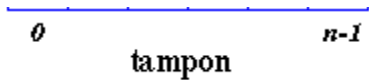
**Amélioration :**

Producteur	Consommateur
<i>produire(messageP);</i>	<b>P(plein)</b>
<b>P(vide)</b>	<i>retirer(case, messageC);</i>
<i>déposer(case, messageP);</i>	<b>V(vide)</b>
<b>V(plein)</b>	<i>consommer(messageC);</i>

## Solution à $n$ cases

**Hypothèse** : tampon à  $n$  cases.





Il faut gérer le tampon. C'est-à-dire :

- si le tampon est *vide*, le consommateur ne peut rien *retirer* ;
- si le tampon est *plein*, le producteur ne peut rien *déposer* ;
- le tampon est *circulaire*, il faut empêcher que les indices *tête* et *queue* se *chevauchent*.

On utilise 2 sémaphores *plein* et *vide* initialisé à **0** et **n**.

Les indices *tête* et *queue* sont initialisés à **0**.

*plein* indique le *nombre* de cases pleines et *vide* ...

Producteur	Consommateur
<code>produire(messageP);</code> <code>P(vide);</code> <code>tampon[tête] = messageP;</code> <code>tête = (tête + 1) mod n;</code> <code>V(plein);</code>	<code>P(plein);</code> <code>messageC = tampon[queue];</code> <code>queue = (queue + 1) mod n;</code> <code>V(vide);</code> <code>consommer(messageC);</code>

## Solution à *p* producteurs et *c* consommateurs

Hypothèses :

- tampon à *n* cases ;
- *p* producteurs (e.g. utilisateurs déposant des requêtes d'impression) ;
- *c* consommateurs (e.g. pool d'imprimantes banalisées).

Il faut protéger l'utilisation des indices.

On utilise 2 sémaphores *mutexprod* et *mutexcons* d'exclusion mutuelle initialisés à **1**.

Producteur	Consommateur
<code>produire(messageP);</code> <code>P(vide);</code> <code>P(mutexprod);</code> <code>tampon[tête] = messageP;</code> <code>tête = (tête + 1) mod n;</code> <code>V(mutexprod);</code> <code>V(plein);</code>	<code>P(plein);</code> <code>P(mutexcons);</code> <code>messageC = tampon[queue];</code> <code>queue = (queue + 1) mod n;</code> <code>V(mutexcons);</code> <code>V(vide);</code> <code>consommer(messageC);</code>



[Retour au sommaire.](#)