

La gestion des processus



GIF-1001 Ordinateurs: Structure et Applications, Hiver 2019
Jean-François Lalonde

Analogies



- Un illusionniste :
 - Fait disparaître certaines limites du matériels
 - Donne l'illusion que la machine a une mémoire infinie et une infinité de processeurs



- Un gouvernement :
 - Protège les utilisateurs les uns des autres
 - Partage des ressources de façon efficace et équitable

Programmes, processus et threads

- Un **programme** est un ensemble d'instructions et de variables dont le but est d'accomplir une tâche précise. Un programme est habituellement créé par un programmeur doté du compilateur adéquat.
- Un **processus** est composé d'un programme et de l'ensemble des ressources reliées à l'exécution du programme. Ces ressources incluent de la mémoire, des fichiers ouverts par le programme, du temps de CPU et autres. Un processus n'est pas un programme! Un processus est créé par:
 - une requête de l'utilisateur (ex: exécution de programme),
 - le système d'exploitation
 - ou un autre processus.

Programmes, processus et threads

- Un ***thread*** est une partie d'un processus qui peut être exécutée indépendamment des autres éléments du processus (en parallèle).
- Un *thread* a ses propres registres (incluant le compteur de programme) et sa propre pile, mais il partage le reste de ses ressources avec les autres constituant du processus.
- Les *threads* sont créés habituellement au début d'un programme. Ils peuvent servir à répondre à des événements dans des programmes (exemples d'événement: clic de souris, touche de clavier enfoncée, message reçu par port série, etc.).

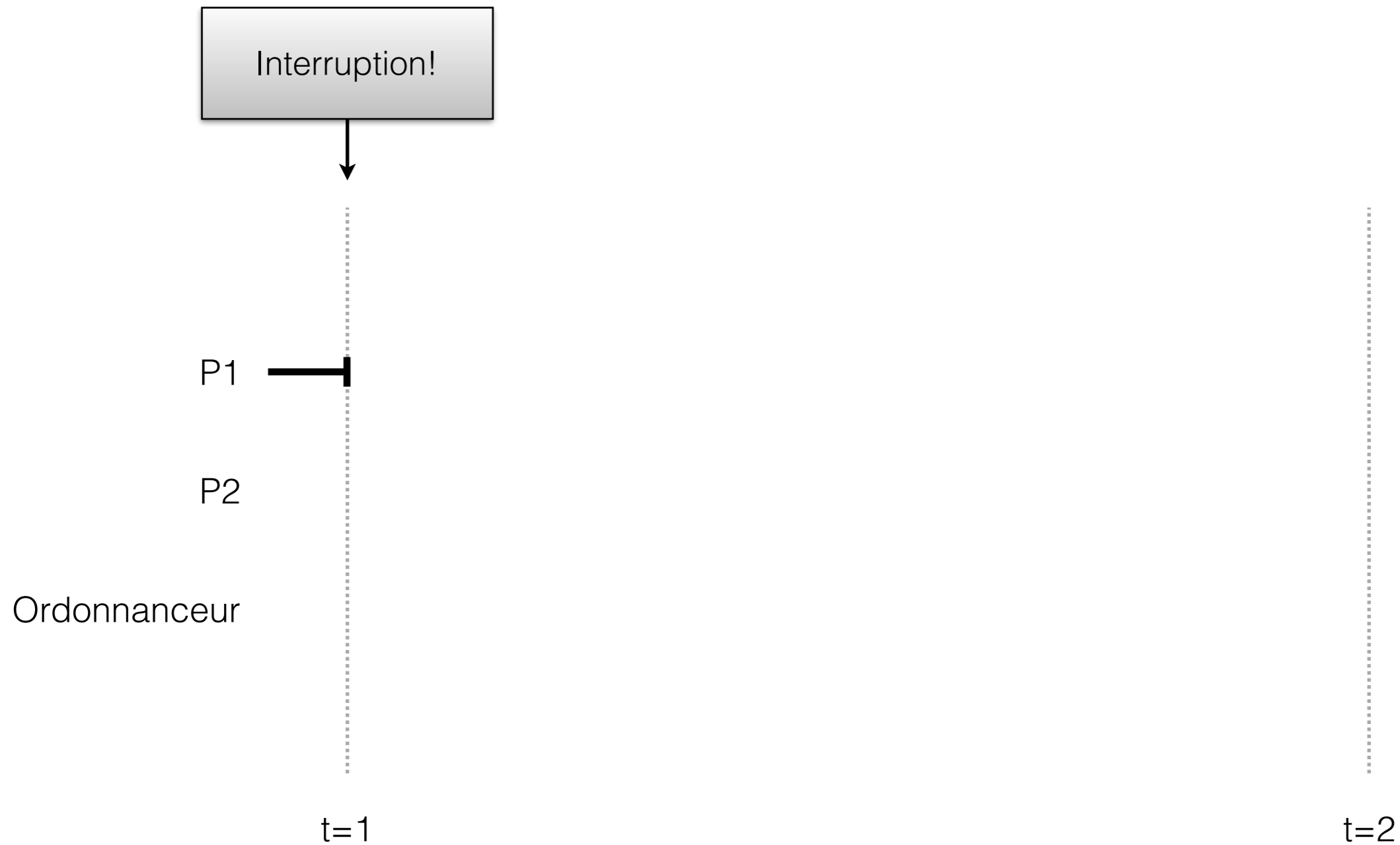
Rappel: Interruptions et système d'exploitation

- Les interruptions permettent l'exécution de plusieurs processus
- Comment?
 - Une horloge génère des interruptions périodiquement
 - À chaque interruption, on change le processus à exécuter
- Un système qui possède des interruptions permettant de sélectionner un nouveau processus à exécuter est dit « préemptif »

Un « quantum », des « quanta »

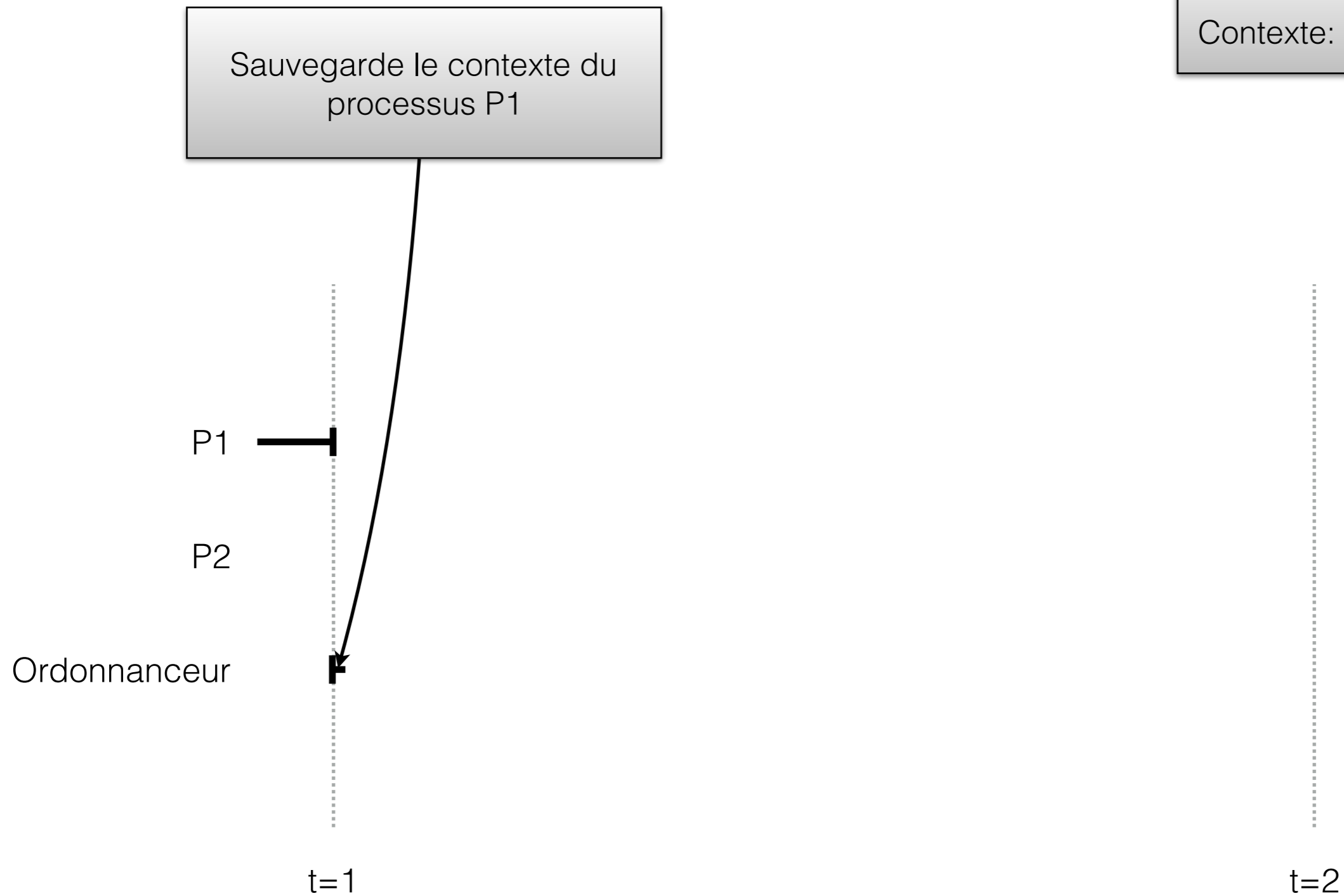
- On divise le temps du microprocesseur en petits morceaux, des « quanta » (*timeslice*) de temps
 - par ex., 6ms est la valeur par défaut dans Linux (v4.15.13)
- À chaque quantum, une interruption est soulevée
- La routine de traitement de l'interruption:
 - Sauvegarde le contexte du processus en exécution (PC, CPSR, registres)
 - Sélectionne un autre processus à exécuter grâce à un programme nommé **l'ordonnanceur**
 - Restaure le contexte de ce nouveau processus (PC, CPSR, registres)
 - Reprend l'exécution du processus où il était rendu

Exécution de 2 processus

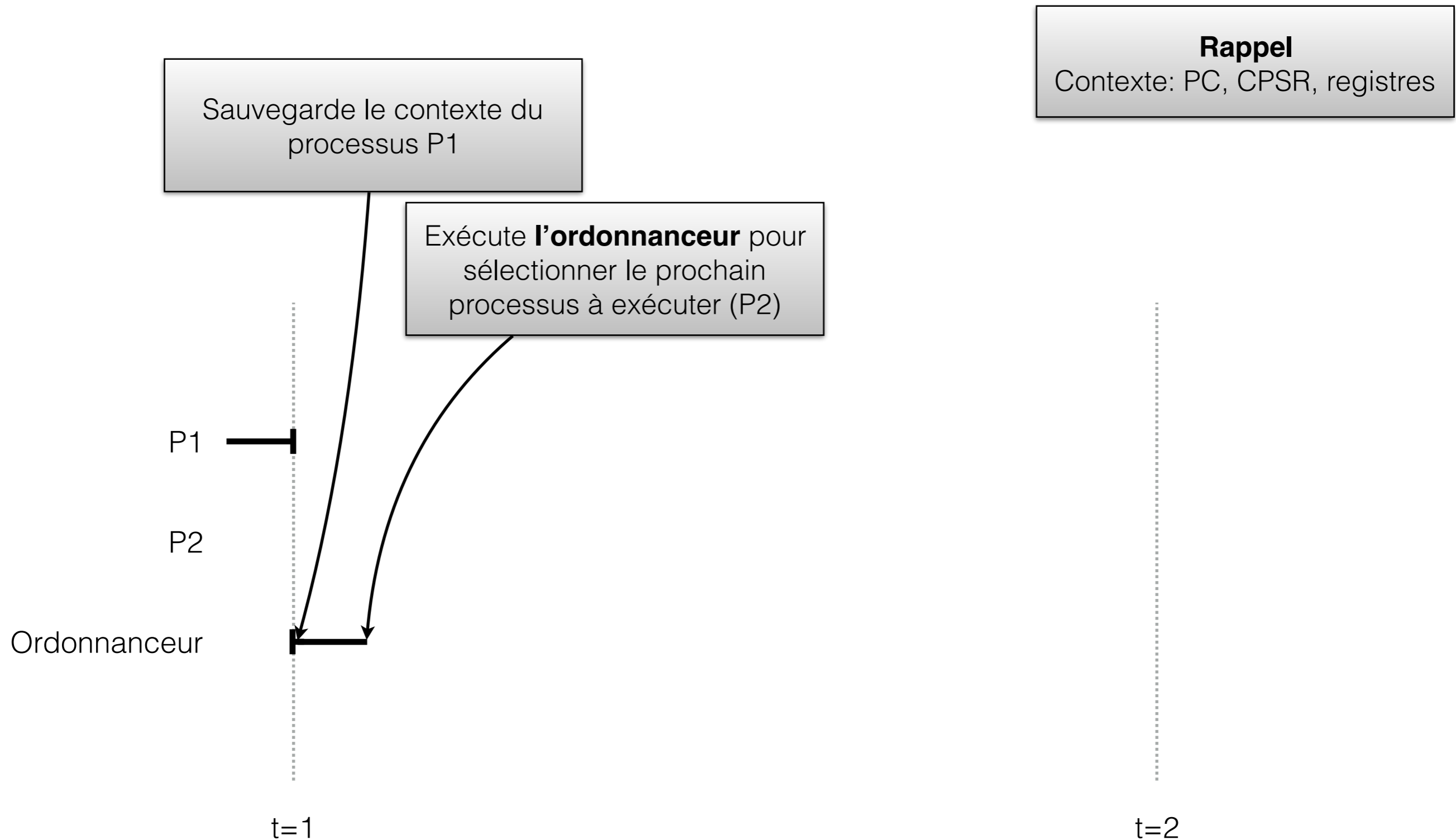


Exécution de 2 processus

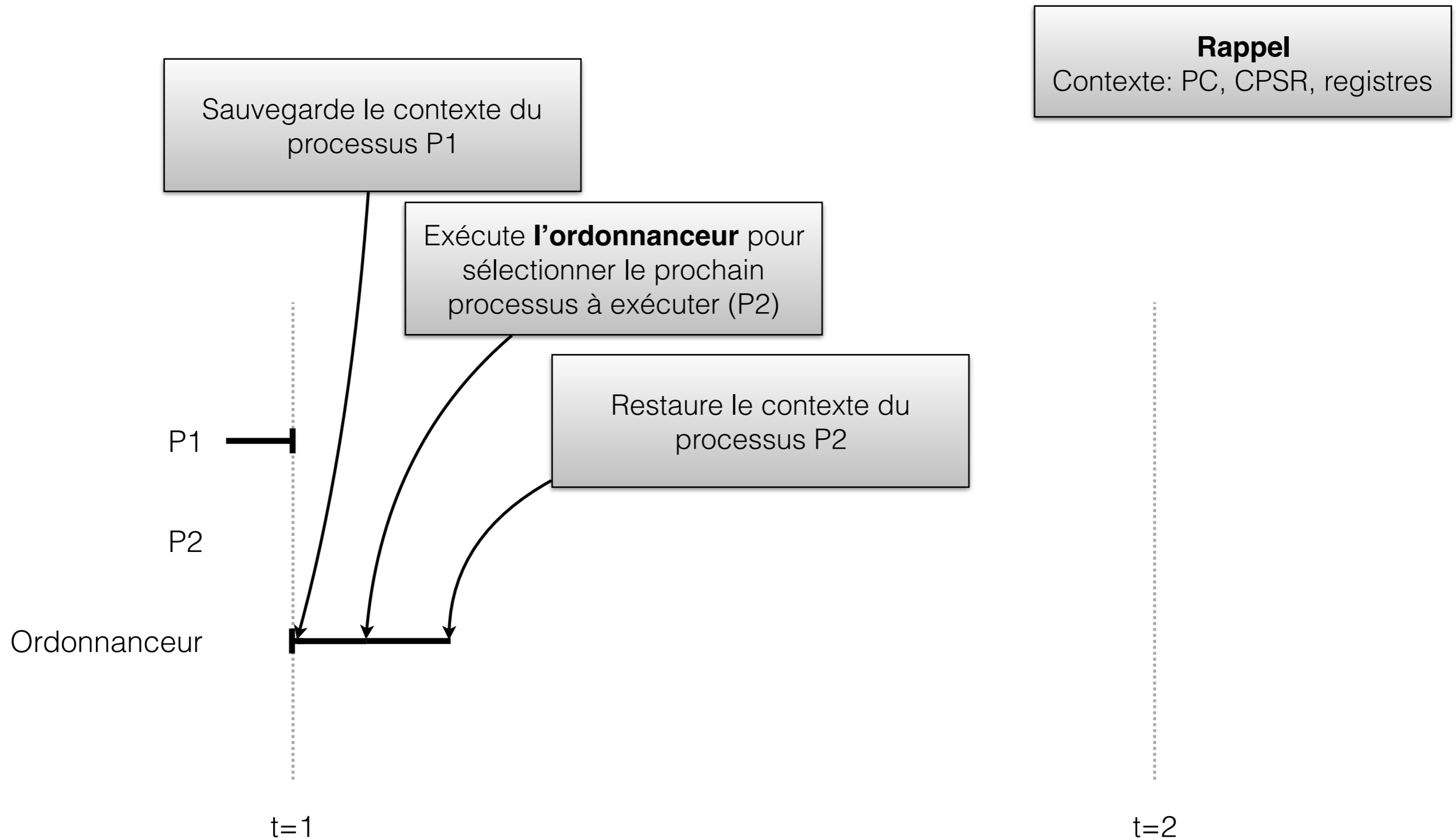
Rappel
Contexte: PC, CPSR, registres



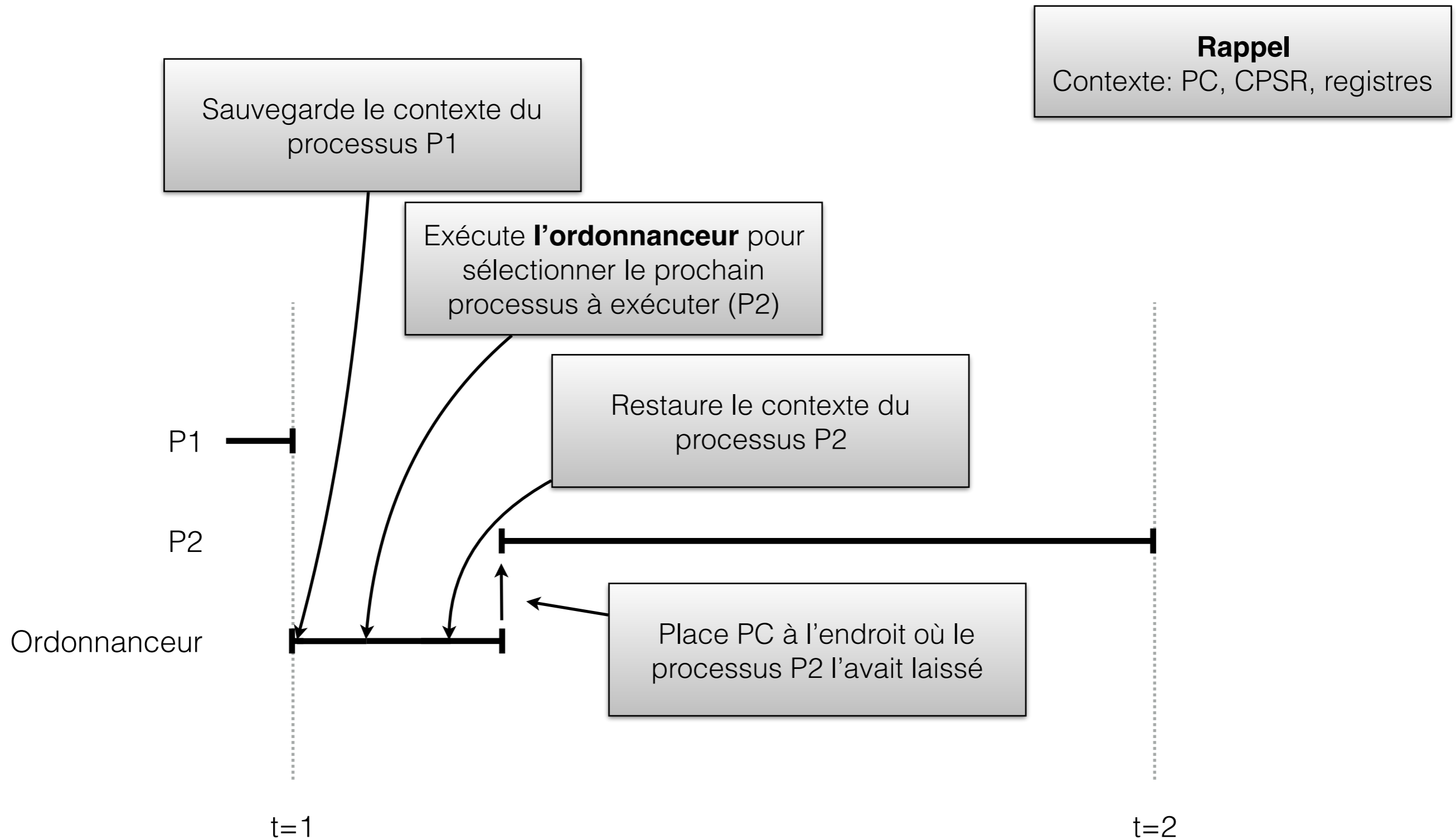
Exécution de 2 processus



Exécution de 2 processus

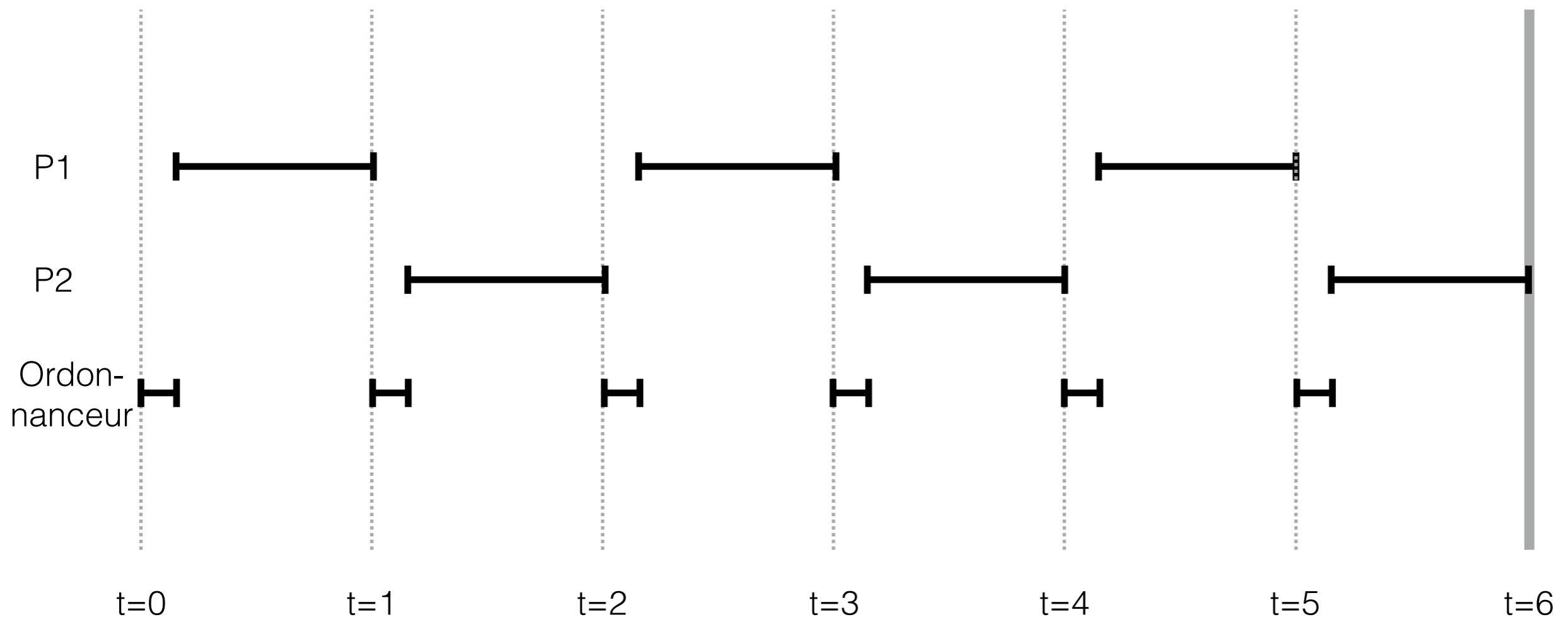


Exécution de 2 processus



Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés.
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum.



Est-ce qu'un processus « sait » qu'il a été interrompu?

Non!

À ses yeux, il ne fait que continuer son exécution
comme s'il était seul sur Terre.

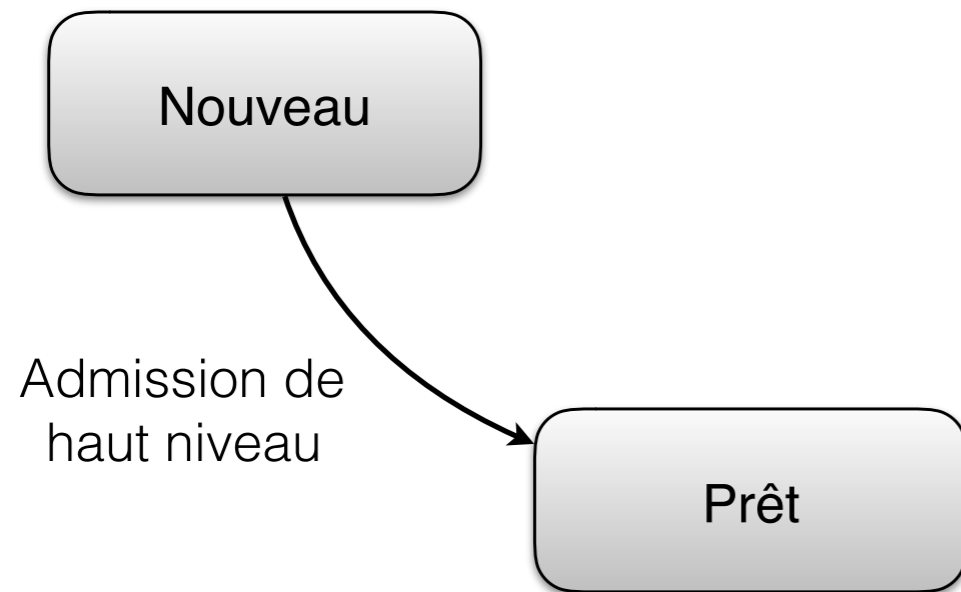
Un processus... dans tous ses états

- À tout moment, un processus possède un **état**.
- Le système d'exploitation (SE) tient un registre de tous les processus à exécuter, ainsi que leur état.
- En pratique, le SE stocke cette information dans un ***Process Control Block*** (PCB) pour chaque processus, qui contient:
 - son identifiant unique;
 - son état;
 - ses registres, sa partie de la mémoire et sa pile;
 - d'autres informations, comme sa priorité.

Un processus... dans tous ses états

- Lorsqu'un processus est démarré, il est tout d'abord examiné par l'*admission de haut niveau* (*high level scheduler*)
- Celui-ci détermine si l'ordinateur possède les ressources nécessaires pour exécuter le processus
- Si oui (la vaste majorité du temps), le processus est admis et tombe dans l'état **prêt**.

Un processus... dans tous ses états



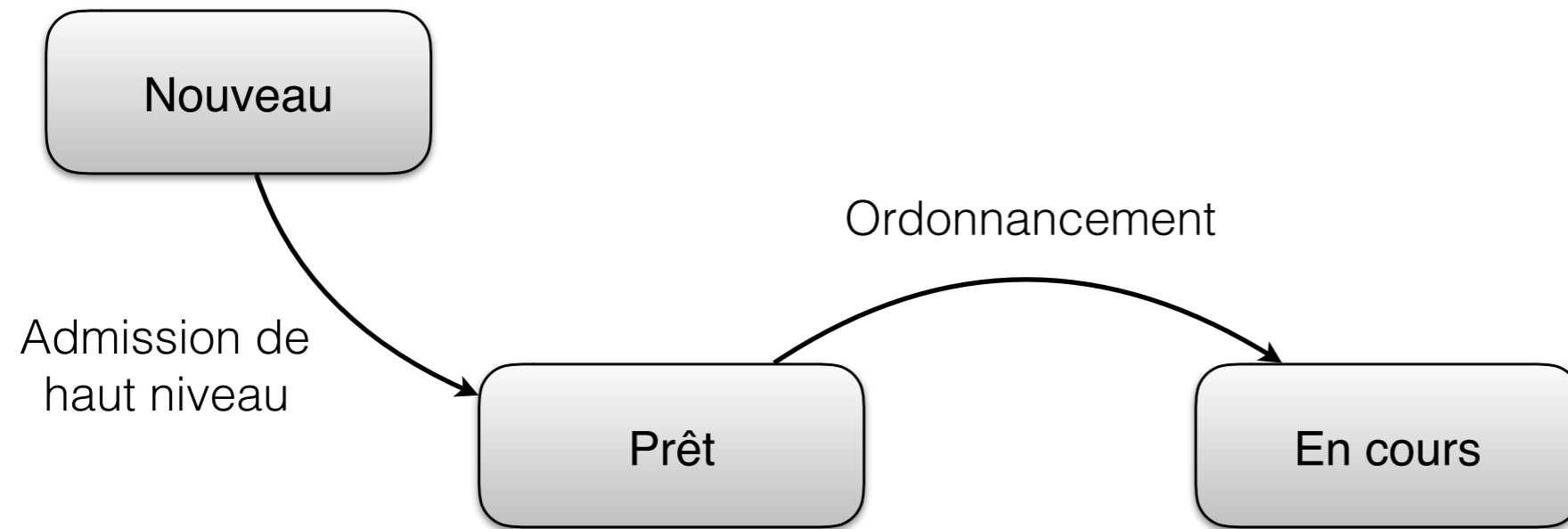
Processus: état « prêt »

- Le micro-processeur est occupé à exécuter un autre processus
- Le processus est prêt d'être exécuté, mais il ne peut pas: il est *en attente du micro-processeur*
- Il reste dans l'état **prêt** jusqu'à ce que *l'ordonnanceur* le sélectionne comme prochain processus à exécuter.

L'ordonnanceur

- Programme (du système d'exploitation) qui sélectionne le prochain processus à exécuter
 - S'exécute périodiquement grâce à des interruptions qui redonnent le contrôle au système d'exploitation
 - Il en existe plusieurs variantes (voir plus loin...)

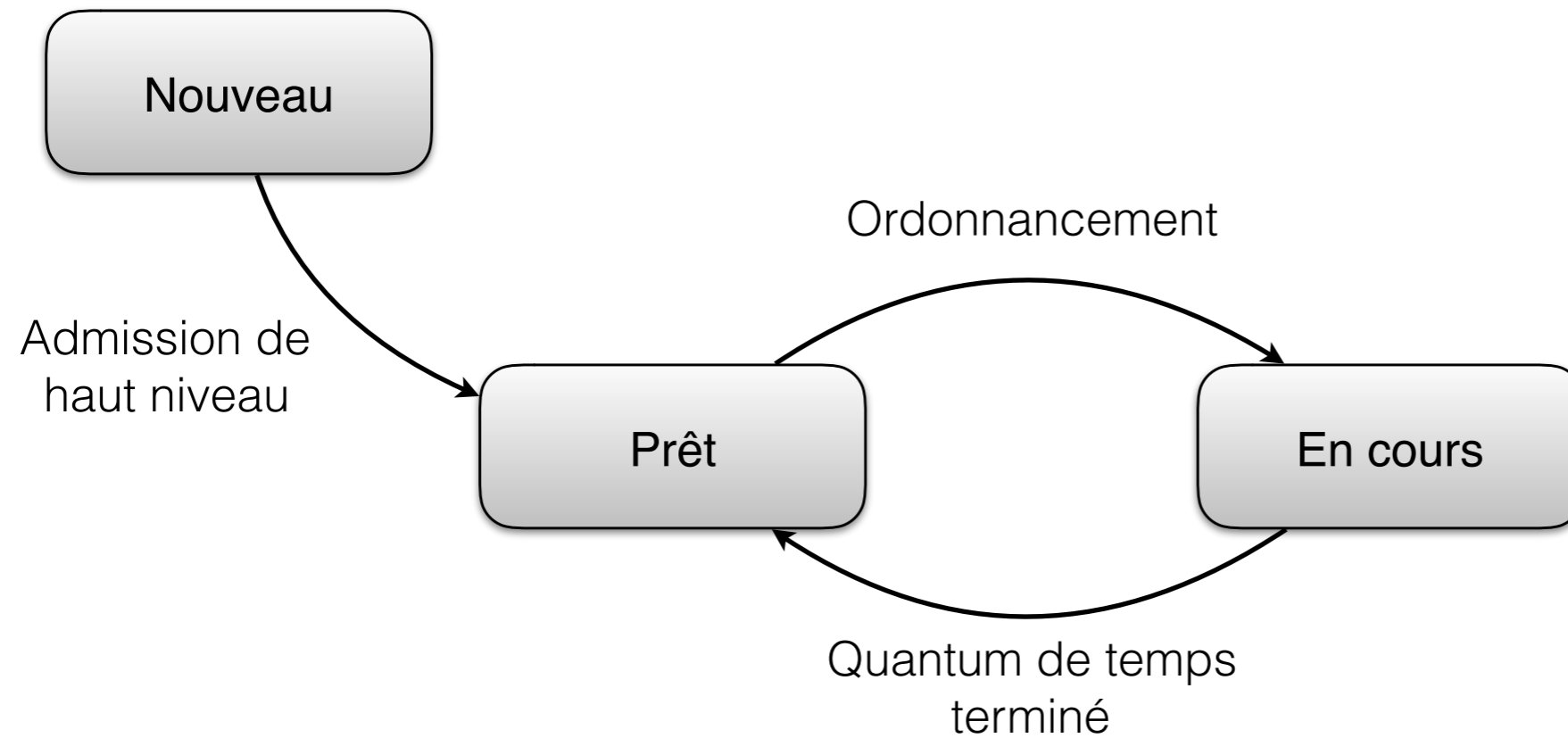
Un processus... dans tous ses états



Processus: état « en cours »

- Le processus contrôle le micro-processeur!
- Il le fait jusqu'à ce que:
 - l'ordonnanceur reprenne le contrôle pour pouvoir exécuter un autre processus
 - il revient alors à l'état **prêt**

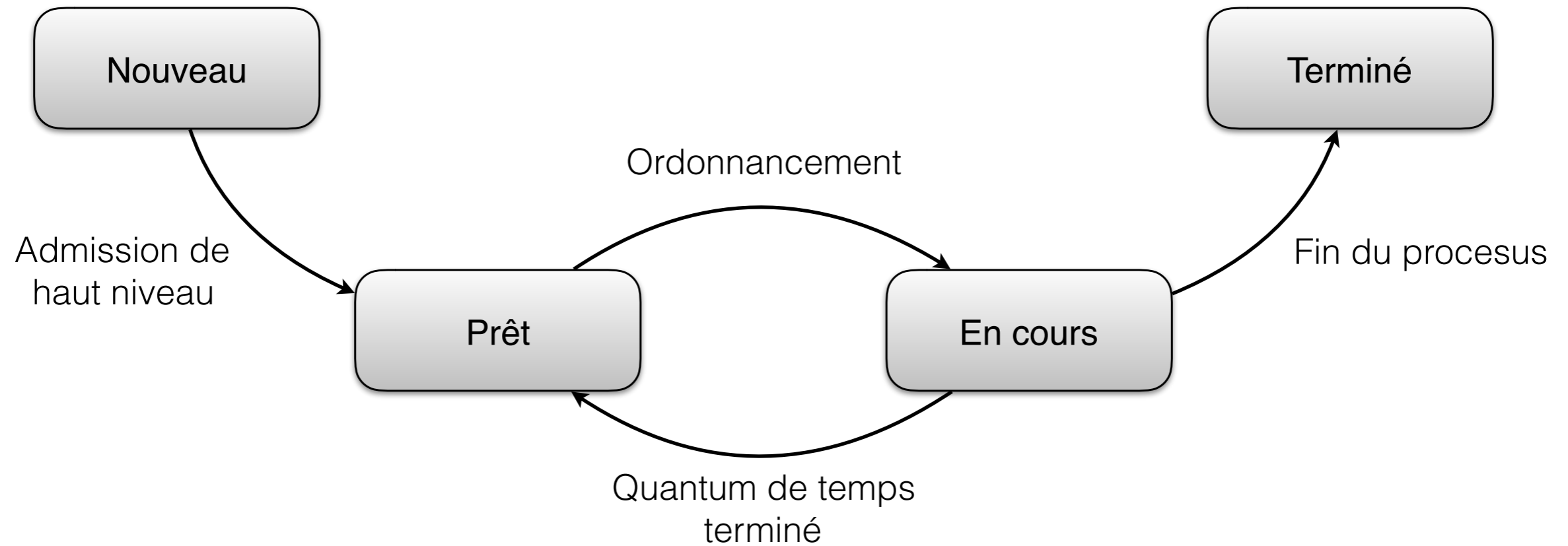
Un processus... dans tous ses états



Processus: état « terminé »

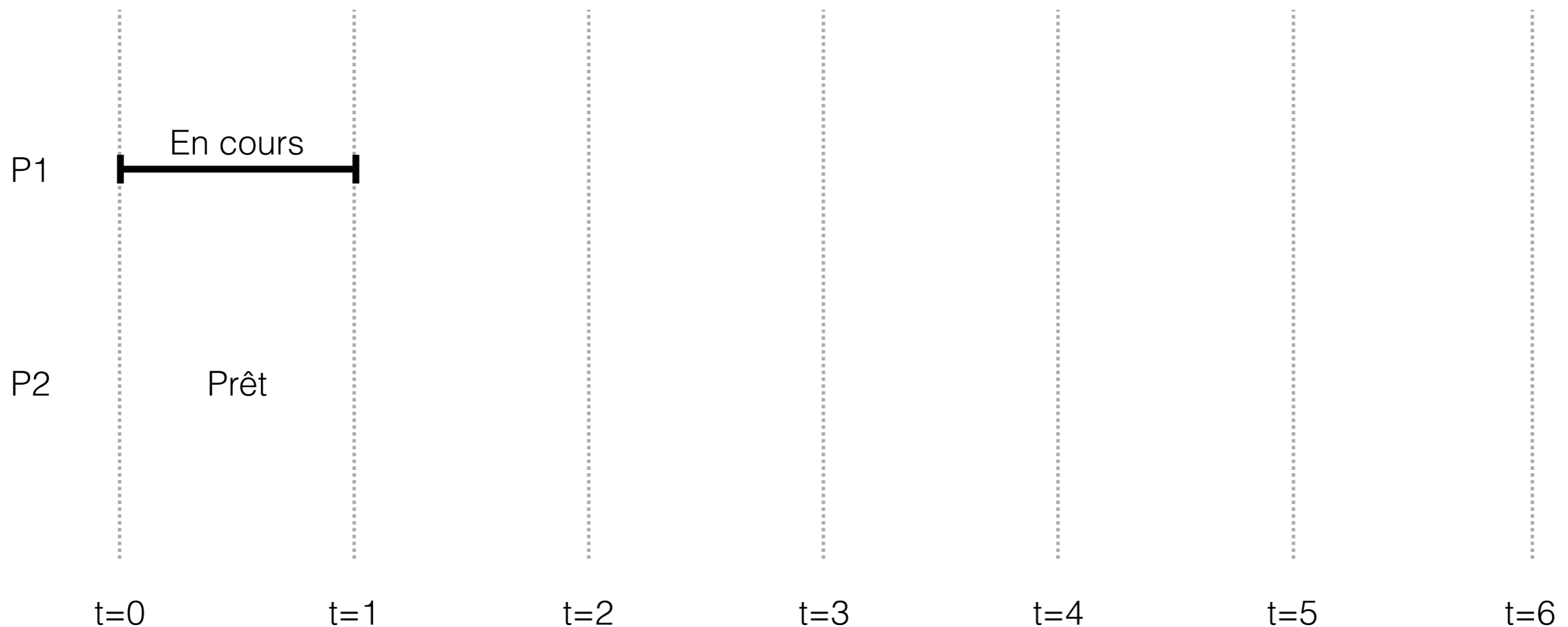
- Lorsque le processus a complété son exécution, ou lorsqu'on le ferme volontairement, il devient dans l'état **terminé**.

Un processus... dans tous ses états



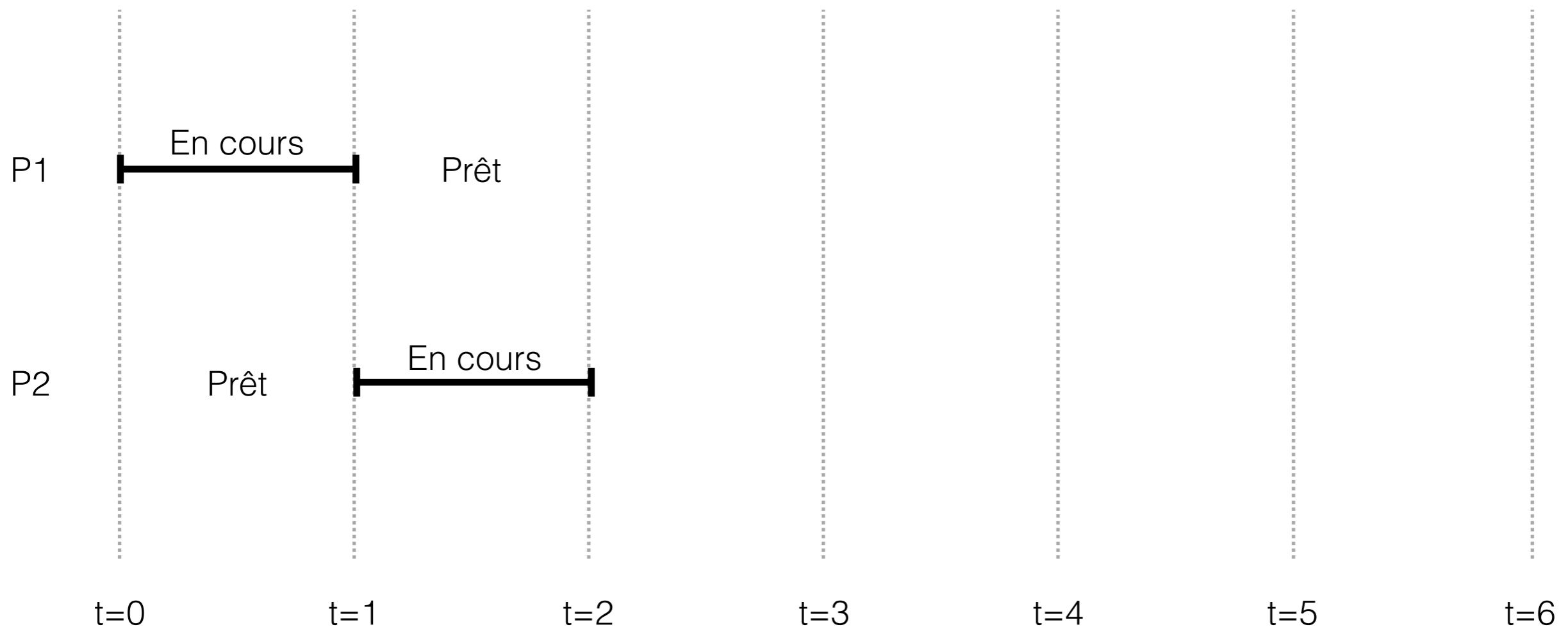
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



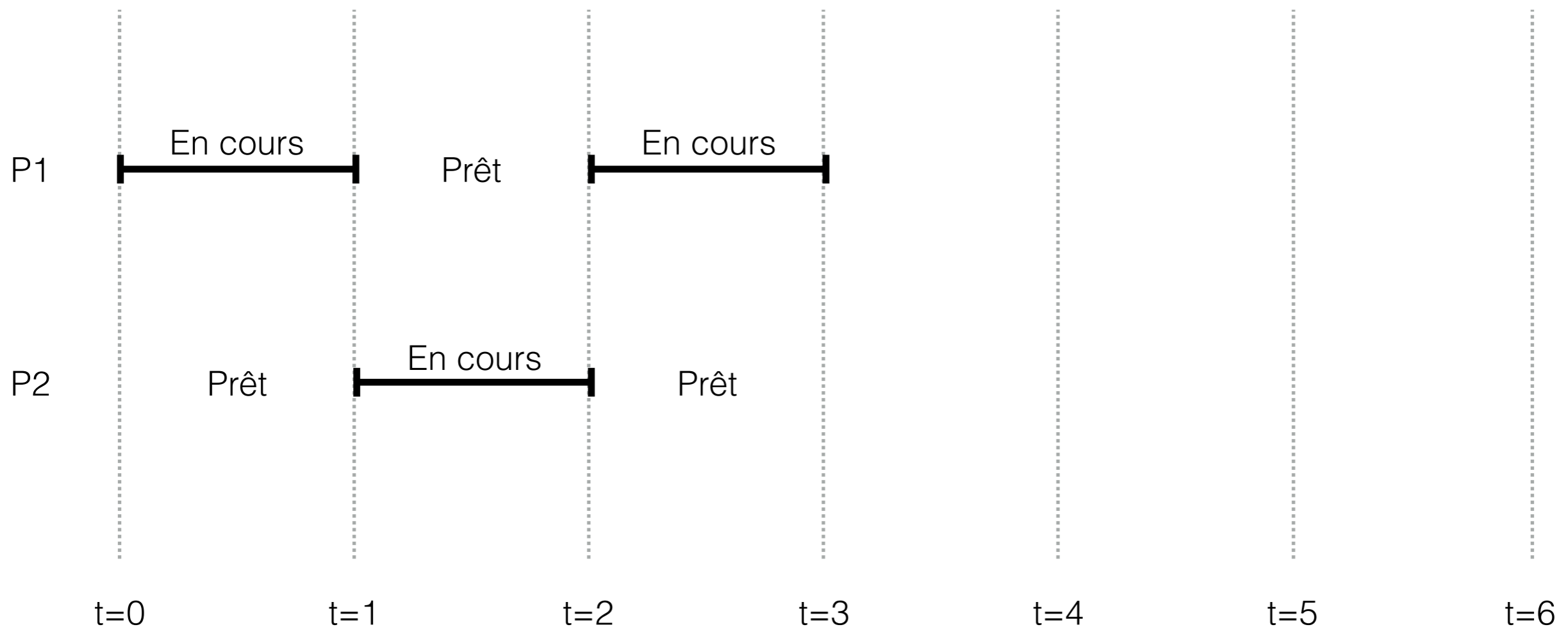
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



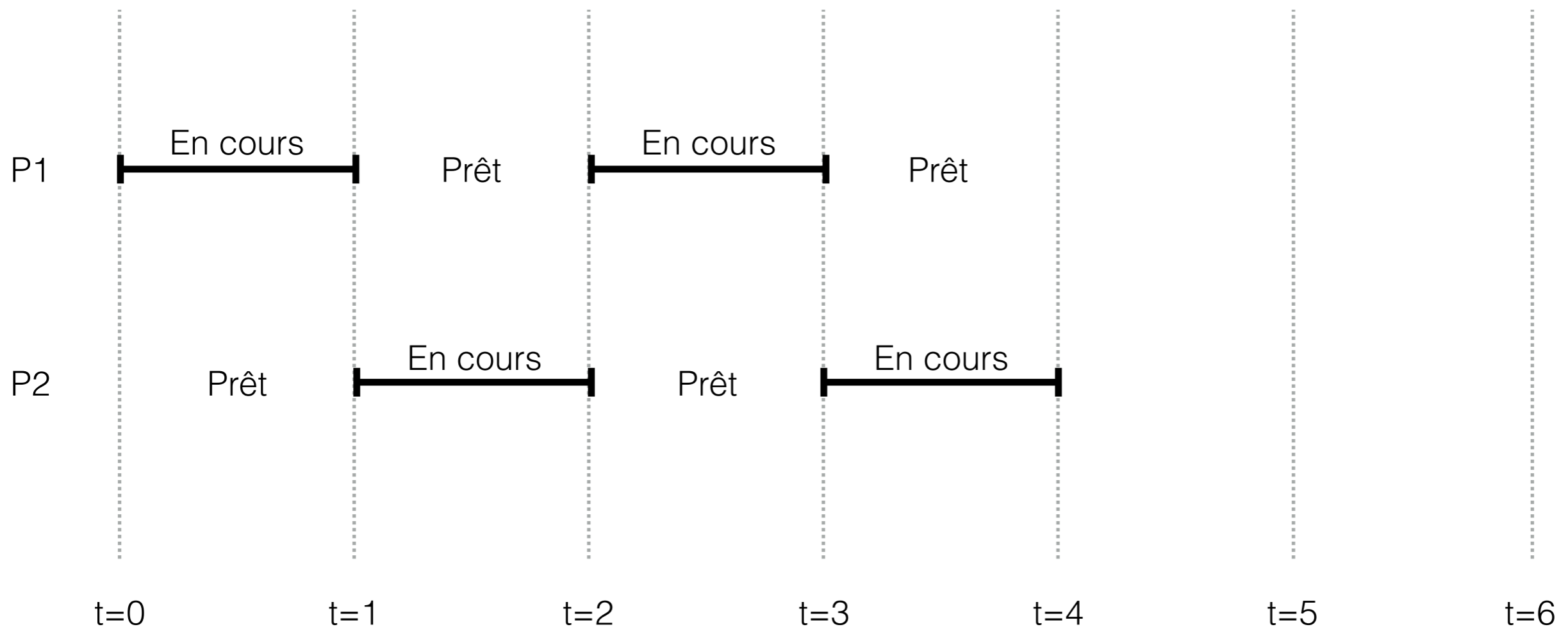
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



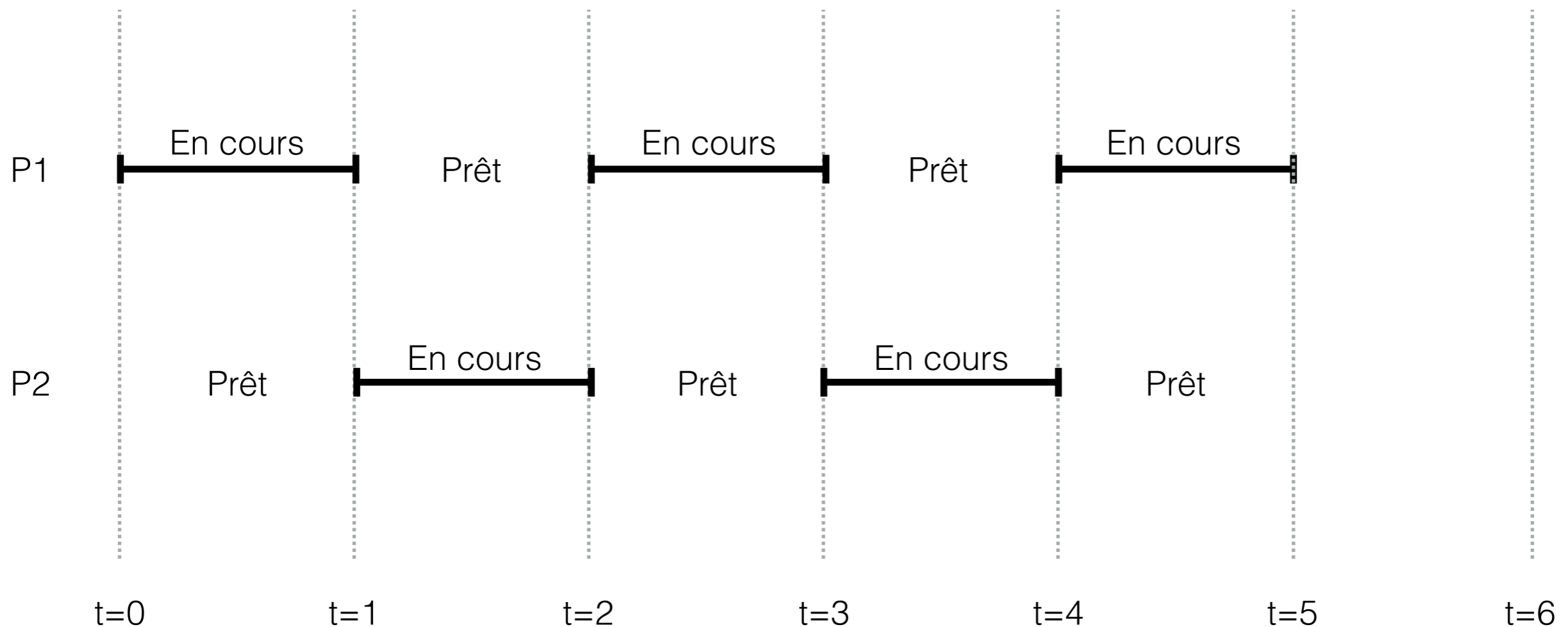
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



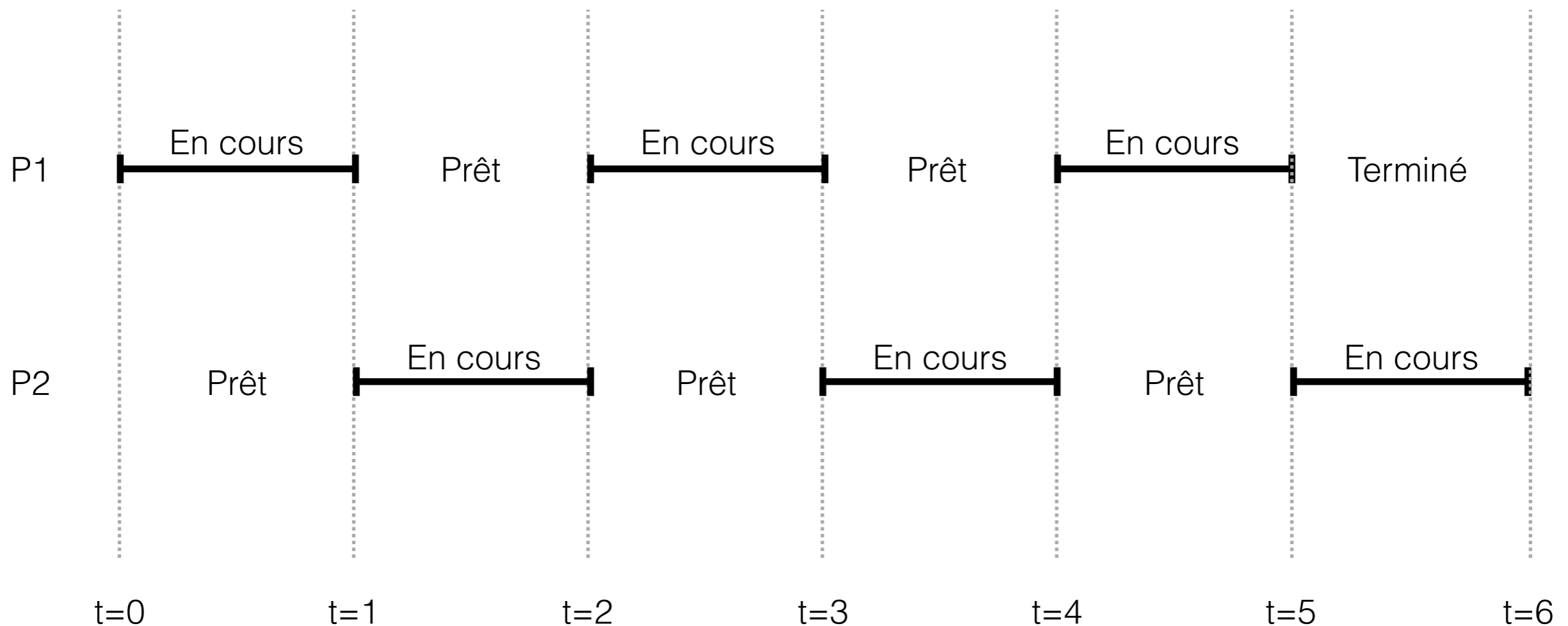
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



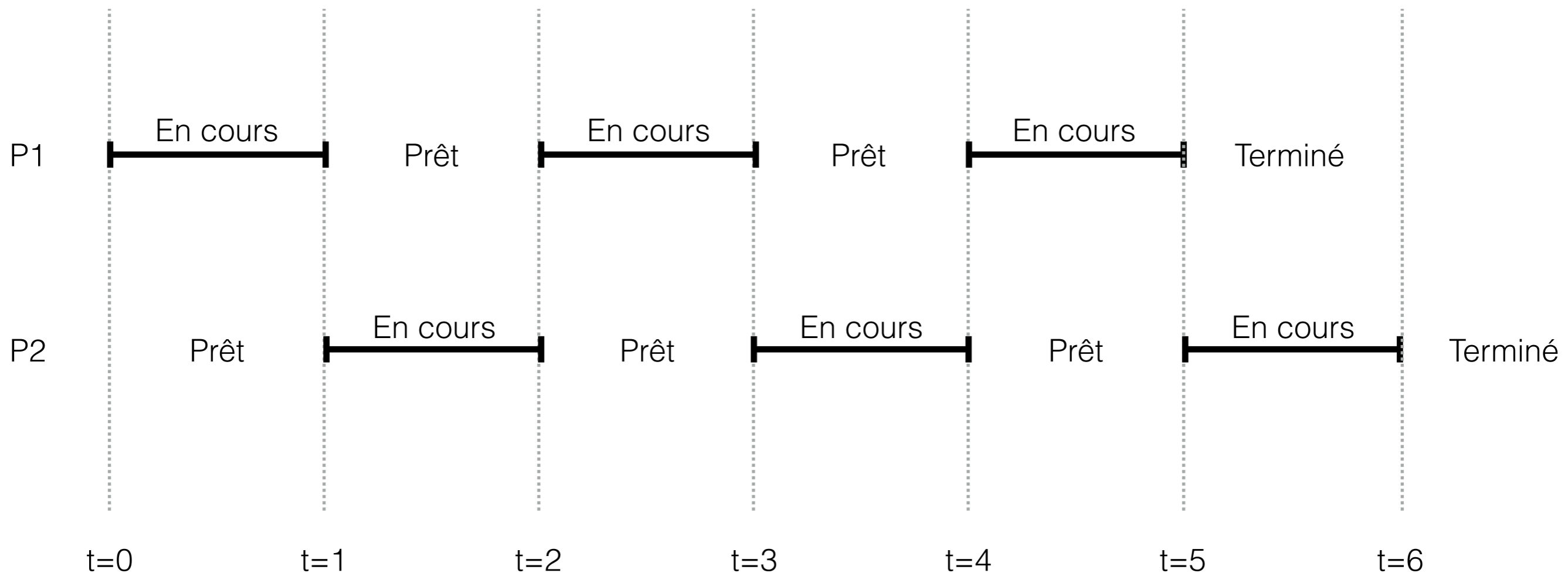
Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. **Chacun nécessite 3 quanta de temps au total.**
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on doit les exécuter en alternance.
- C'est **l'ordonnanceur** qui décide quel processus exécuter à chaque quantum (pas affiché ici).
- Affichons les états de chaque processus.



Ordonnancement (Dispatching)

- L'ordonnancement des processus consiste simplement à décider quel processus sera exécuté dans le quantum suivant.
- Un algorithme d'ordonnancement a les objectifs suivants:

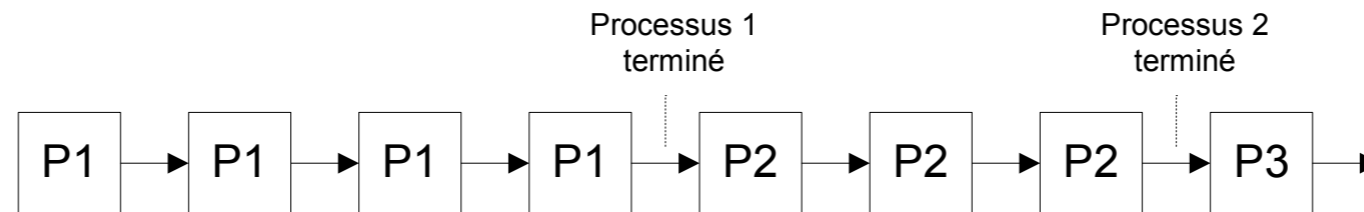
Assurer l'équité	Tous les processus sont traités également
Maximiser l'exécution	Terminer le plus de processus possible
Temps d'exécution min.	Temps d'exécution le plus court possible
Utilisation max du CPU	Le CPU doit être utilisé au maximum
Utilisation max ressources	Les ressources doivent être utilisées au max.
Détérioration graduelle	Un système surchargé doit ralentir, pas planter
Temps d'attente min.	Petit délai entre l'admission et l'exécution
Temps de réponse correct	Tâches longues, longues et tâches courtes, courtes
Prévenir la famine (starvation)	L'exécution d'un processus ne doit pas être reportée indéfiniment.

Analogie de la vie de tous les jours™

NETFLIX

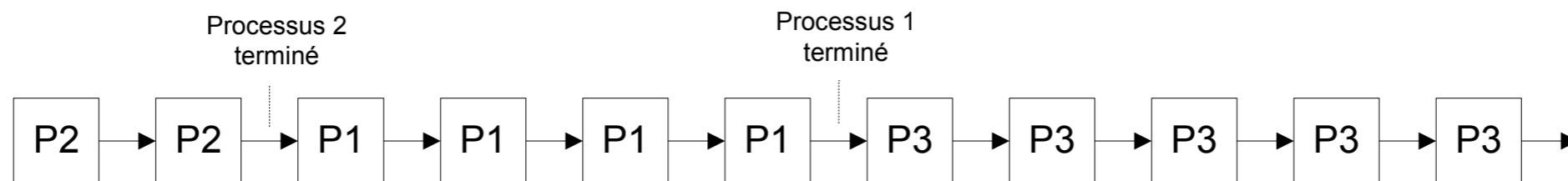


Algorithmes d'ordonnement



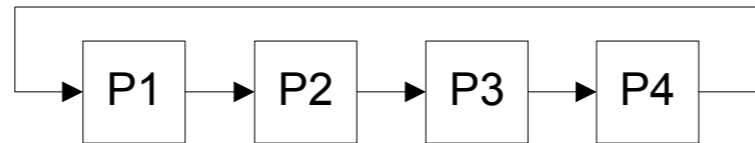
- **Premier arrivé, premier servi**
(exemple de *mauvais* algorithme)
 - Le premier processus admis est exécuté jusqu'à sa fin. Puis, on exécute le suivant.
 - Cet algorithme n'est jamais utilisé. Il équivaudrait à remplacer le noyau du système d'exploitation par une file d'exécution des processus et détruirait l'illusion d'une exécution simultanée de plusieurs processus.

Algorithmes d'ordonnement



- **Le plus court d'abord:** On exécute le processus le plus court d'abord.
 - Avantages: Maximise l'exécution et le temps d'exécution
 - Désavantages: Famine possible, inéquitable

Algorithmes d'ordonnancement



- **Le tourniquet (round-robin):** On exécute les processus à tour de rôle.
 - Avantages: Tous les processus ont du temps de CPU, très équitable
 - Désavantages: Ne maximise pas l'exécution ni le temps d'exécution

Ordonnancement: algorithme général

1. S'il y a un nouveau processus:
 - 1.1. L'admettre en mémoire;
 - 1.2. (**Tourniquet seulement**): placer le processus à la fin de la file d'attente *correspondant à sa priorité.*
2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnancement;
3. Calculer la durée restante du processus choisi;
 - 3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;
 - 3.2. (**Tourniquet seulement**): Sinon, placer le processus à la fin de la file d'attente;
4. Passer au quanta suivant.

Exercice #1

- Les processus suivants sont admis en mémoire (dans l'ordre):
 - P1, durée: 5
 - P2, durée: 3
- Écrivez quel processus est exécuté par le micro-processeur à chaque quantum de temps si l'algorithme utilisé par l'ordonnanceur est:
 - premier arrivé, premier servi
 - plus court d'abord
 - tourniquet

1. S'il y a un nouveau processus:
 - 1.1. L'admettre en mémoire;
 - 1.2. (**Tourniquet seulement**): placer le processus à la fin de la file d'attente correspondant à sa priorité.
2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnement;
3. Calculer la durée restante du processus choisi;
 - 3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;
 - 3.2. (**Tourniquet seulement**): Sinon, placer le processus à la fin de la file d'attente;
4. Passer au quanta suivant.

Exercice #1 : solution

- Les processus suivants sont admis en mémoire (dans l'ordre):

- P1, durée: 5
- P2, durée: 3

- Écrivez quel processus est exécuté par le micro-processeur à chaque quantum de temps si l'algorithme utilisé par l'ordonnanceur est:

- premier arrivé, premier servi: P1—P1—P1—P1—P1—P2—P2—P2
- plus court d'abord: P2—P2—P2—P1—P1—P1—P1—P1
- tourniquet: P1—P2—P1—P2—P1—P2—P1—P1

1. S'il y a un nouveau processus:

1.1. L'admettre en mémoire;

1.2. (**Tourniquet seulement**): placer le processus à la fin de la file d'attente correspondant à sa priorité.

2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnancement;

3. Calculer la durée restante du processus choisi;

3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;

3.2. (**Tourniquet seulement**): Sinon, placer le processus à la fin de la file d'attente;

4. Passer au quanta suivant.

Arrivée dynamique des processus

- En pratique, les processus arrivent à différents moments. Il faut donc recalculer quel processus sera exécuté en prochain.
- Plus court d'abord
 - On maintient une liste des processus, et on sélectionne le plus court
- Tourniquet
 - On maintient une file d'attente. Lorsqu'un nouveau processus arrive, on le place à la *fin* de la file d'attente.

Exercice #2

- Les processus suivants sont admis en mémoire (dans l'ordre):
 - P1, durée: 5, temps d'arrivée: 0
 - P2, durée: 4, temps d'arrivée: 0
 - P3, durée: 1, temps d'arrivée: 2
 - P4, durée: 3, temps d'arrivée: 5
- Écrivez quel processus est exécuté par le micro-processeur à chaque quantum de temps si l'algorithme utilisé par l'ordonnanceur est:
 - premier arrivé, premier servi
 - plus court d'abord
 - tourniquet

1. S'il y a un nouveau processus:
 - 1.1. L'admettre en mémoire;
 - 1.2. (**Tourniquet seulement**): placer le processus à la fin de la file d'attente correspondant à sa priorité.
2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnancement;
3. Calculer la durée restante du processus choisi;
 - 3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;
 - 3.2. (**Tourniquet seulement**): Sinon, placer le processus à la fin de la file d'attente;
4. Passer au quanta suivant.

Exercice #2: tourniquet

Quantum: 0

File d'attente:



Processus en mémoire

P1:



P2:



P3:



P4:



Ordonnancement:



Voici une bonne façon de visualiser les étapes des algorithmes d'ordonnancement de processus.

On écrit:

1. le quantum
2. la (ou les) file(s) d'attente (si nécessaire)
3. la liste des processus en mémoire
4. l'ordonnancement résultant.

Ensuite, on exécute les étapes de l'algorithme pas à pas en mettant à jour chaque section.

Exercice #2: tourniquet

Quantum: 0



Processus en mémoire



Exercice #2: tourniquet

Quantum: 0



Processus en mémoire



Exercice #2: tourniquet

Quantum: 1



Processus en mémoire



Exercice #2: tourniquet

Quantum: 1



Processus en mémoire



Exercice #2: tourniquet

Quantum: 2

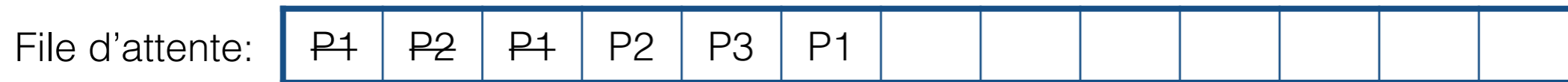


Processus en mémoire



Exercice #2: tourniquet

Quantum: 2



Processus en mémoire



Exercice #2: tourniquet

Quantum: 3



Processus en mémoire



Exercice #2: tourniquet

Quantum: 3



Processus en mémoire



Exercice #2: tourniquet

Quantum: 4

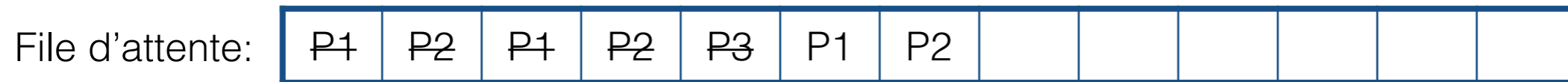


Processus en mémoire

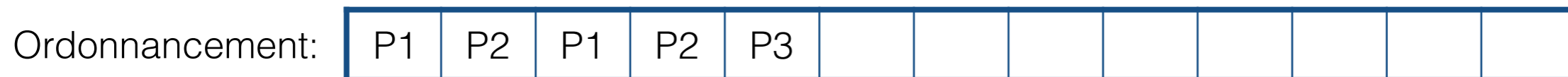


Exercice #2: tourniquet

Quantum: 4



Processus en mémoire



Exercice #2: tourniquet

Quantum: 5

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4						
----	----	----	----	----	----	----	----	--	--	--	--	--	--

Processus en mémoire

P1:

5	4	3											
---	---	---	--	--	--	--	--	--	--	--	--	--	--

P2:

4	3	2											
---	---	---	--	--	--	--	--	--	--	--	--	--	--

P3:

1	0												
---	---	--	--	--	--	--	--	--	--	--	--	--	--

P4:

3													
---	--	--	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3									
----	----	----	----	----	--	--	--	--	--	--	--	--	--

Exercice #2: tourniquet

Quantum: 5

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1				
----	----	----	----	----	----	----	----	----	--	--	--	--

Processus en mémoire

P1:

5	4	3	2									
---	---	---	---	--	--	--	--	--	--	--	--	--

P2:

4	3	2										
---	---	---	--	--	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3												
---	--	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1							
----	----	----	----	----	----	--	--	--	--	--	--	--

Exercice #2: tourniquet

Quantum: 6

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2			
----	----	----	----	----	----	----	----	----	----	--	--	--

Processus en mémoire

P1:

5	4	3	2									
---	---	---	---	--	--	--	--	--	--	--	--	--

P2:

4	3	2	1									
---	---	---	---	--	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3												
---	--	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2						
----	----	----	----	----	----	----	--	--	--	--	--	--

Exercice #2: tourniquet

Quantum: 7

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4		
----	----	----	----	----	----	----	----	----	----	----	--	--

Processus en mémoire

P1:

5	4	3	2									
---	---	---	---	--	--	--	--	--	--	--	--	--

P2:

4	3	2	1									
---	---	---	---	--	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2											
---	---	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4					
----	----	----	----	----	----	----	----	--	--	--	--	--

Exercice #2: tourniquet

Quantum: 8

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	
----	----	----	----	----	----	----	----	----	----	----	----	--

Processus en mémoire

P1:

5	4	3	2	1								
---	---	---	---	---	--	--	--	--	--	--	--	--

P2:

4	3	2	1									
---	---	---	---	--	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2											
---	---	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1				
----	----	----	----	----	----	----	----	----	--	--	--	--

Exercice #2: tourniquet

Quantum: 9

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	
----	----	----	----	----	----	----	----	----	----	----	----	--

Processus en mémoire

P1:

5	4	3	2	1								
---	---	---	---	---	--	--	--	--	--	--	--	--

P2:

4	3	2	1	0								
---	---	---	---	---	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2											
---	---	--	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2			
----	----	----	----	----	----	----	----	----	----	--	--	--

Exercice #2: tourniquet

Quantum: 10

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	P4
----	----	----	----	----	----	----	----	----	----	----	----	----

Processus en mémoire

P1:

5	4	3	2	1								
---	---	---	---	---	--	--	--	--	--	--	--	--

P2:

4	3	2	1	0								
---	---	---	---	---	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2	1										
---	---	---	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4		
----	----	----	----	----	----	----	----	----	----	----	--	--

Exercice #2: tourniquet

Quantum: 11

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	P4
----	----	----	----	----	----	----	----	----	----	----	----	----

Processus en mémoire

P1:

5	4	3	2	1	0							
---	---	---	---	---	---	--	--	--	--	--	--	--

P2:

4	3	2	1	0								
---	---	---	---	---	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2	1										
---	---	---	--	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	
----	----	----	----	----	----	----	----	----	----	----	----	--

Exercice #2: tourniquet

Quantum: 12

File d'attente:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	P4
----	----	----	----	----	----	----	----	----	----	----	----	----

Processus en mémoire

P1:

5	4	3	2	1	0							
---	---	---	---	---	---	--	--	--	--	--	--	--

P2:

4	3	2	1	0								
---	---	---	---	---	--	--	--	--	--	--	--	--

P3:

1	0											
---	---	--	--	--	--	--	--	--	--	--	--	--

P4:

3	2	1	0									
---	---	---	---	--	--	--	--	--	--	--	--	--

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4	P1	P4
----	----	----	----	----	----	----	----	----	----	----	----	----

Exercice #2: solution

- Les processus suivants sont admis à l'ordonnanceur (dans l'ordre):
 - P1, durée: 5, temps d'arrivée: 0
 - P2, durée: 4, temps d'arrivée: 0
 - P3, durée: 1, temps d'arrivée: 2
 - P4, durée: 3, temps d'arrivée: 5
- Écrivez quel processus est exécuté par le micro-processeur à chaque quantum de temps si l'algorithme utilisé par l'ordonnanceur est:
 - premier arrivé, premier servi
 - P1—P1—P1—P1—P1—P2—P2—P2—P2—P3—P4—P4—P4
 - plus court d'abord
 - P2—P2—P3—P2—P2—P4—P4—P4—P1—P1—P1—P1—P1
 - tourniquet
 - P1—P2—P1—P2—P3—P1—P2—P4—P1—P2—P4—P1—P4

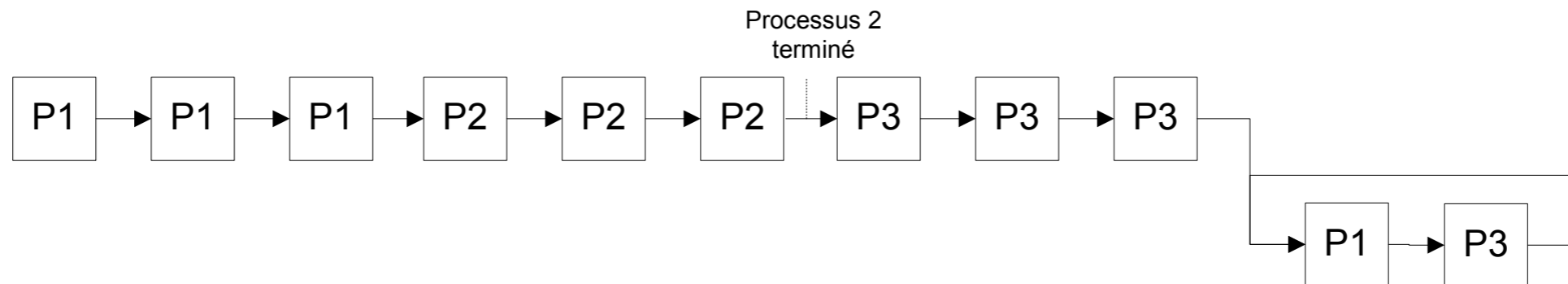
Algorithmes d'ordonnement

- **Priorité:** On exécute les processus selon leur priorité.
 - Avantages: Temps de réponse correct, temps d'attente minimum
 - Désavantages: Ne maximise pas l'exécution ni le temps d'exécution, famine possible, le programmeur doit déclarer des priorités

Algorithmes d'ordonnement

- **Priorité Variable:** On exécute les processus selon leur priorité. La priorité d'un processus change dynamiquement en fonction d'évènements (fin d'attente, le processus a faim, admission, ...).
 - Avantages: Temps de réponse correct, temps d'attente minimum
 - Désavantages: Même que l'algorithme de priorité, mais avec un impact beaucoup moindre.

Algorithmes d'ordonnancement

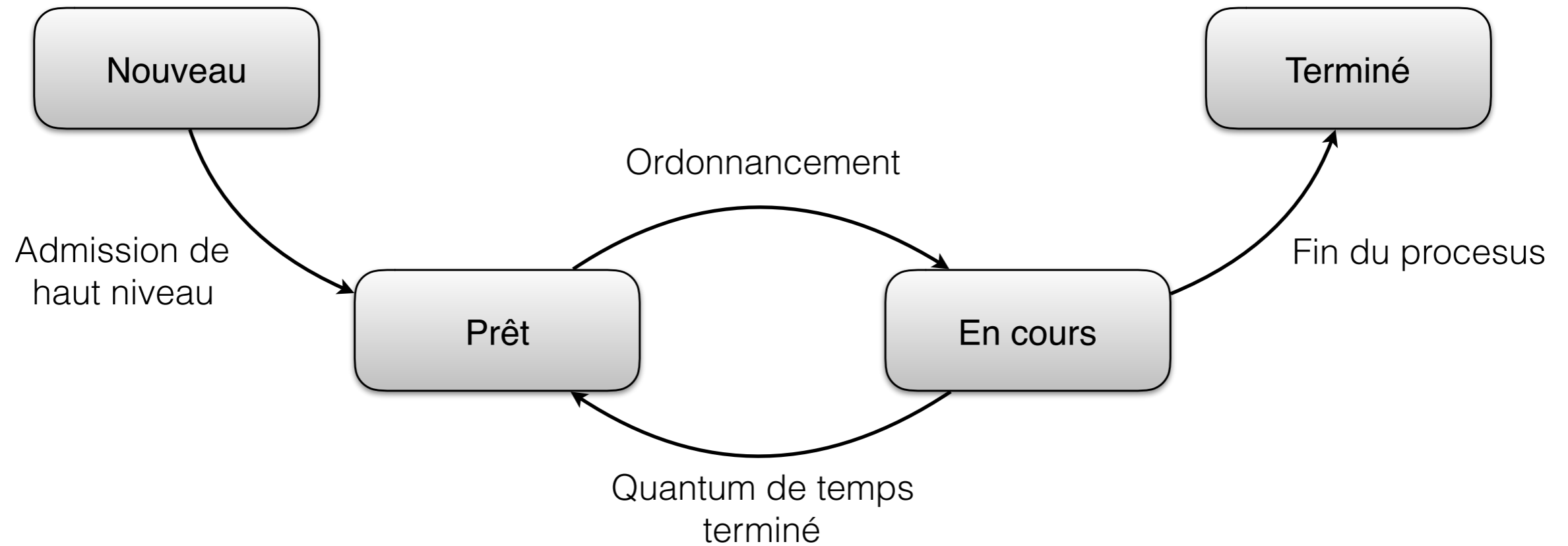


- **File avec tourniquet (round-robin):** On exécute plusieurs fois les processus nouvellement admis, puis les processus non terminés sont mis dans un tourniquet.
- Avantages: Tous les processus ont du temps de CPU; très équitable.
- Désavantages: Même que tourniquet, mais avec un impact moindre

Algorithmes d'ordonnement

- **Le plus critique (RTOS surtout):** Le processus devant être exécuté dans les plus brefs délais est d'abord exécuté.
 - Avantages: Garantit l'exécution d'un processus critique à l'intérieur d'un certain temps.
 - Désavantages: Ne maximise pas l'exécution ni le temps d'exécution, famine possible, le programmeur doit déclarer des priorités

Un processus... dans tous ses états



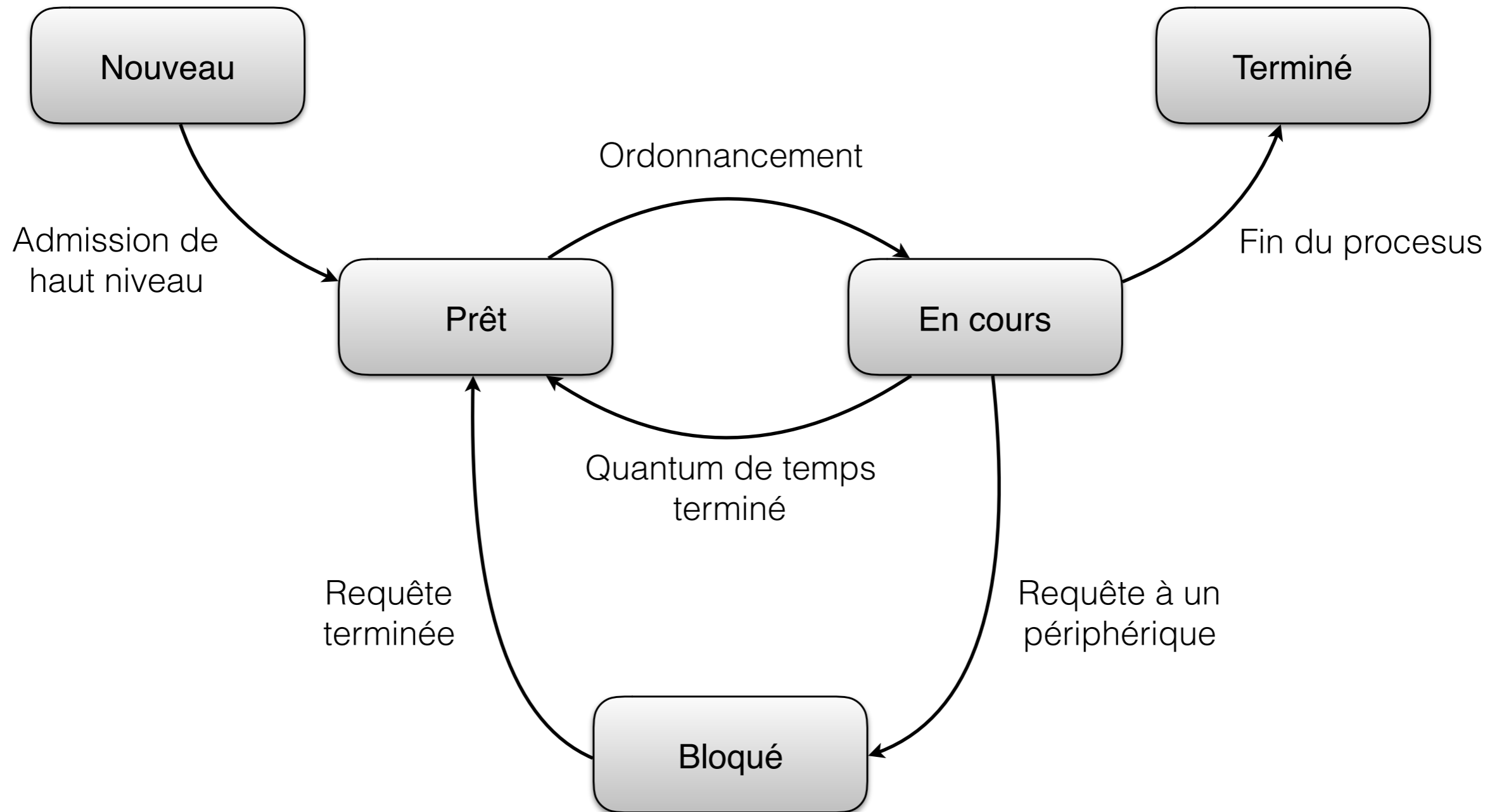
Processus: état « en cours »

- Le processus contrôle le micro-processeur!
- Il le fait jusqu'à ce que:
 - l'ordonnanceur reprenne le contrôle pour pouvoir exécuter un autre processus
 - il revient alors à l'état **prêt**
 - il demande à utiliser un périphérique (entrée-sortie)
 - il tombe alors dans l'état **bloqué**

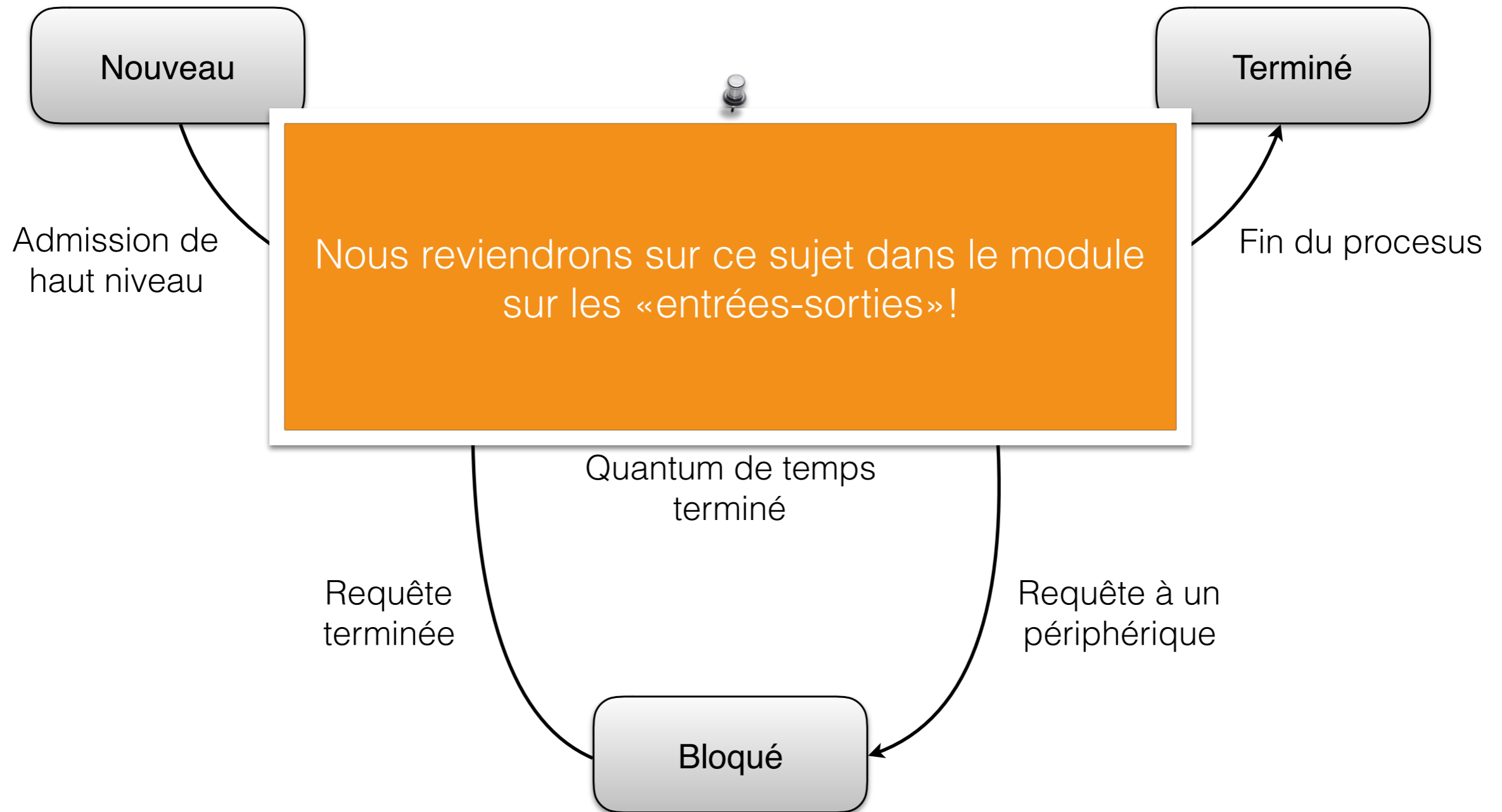
Processus: état « bloqué »

- Le processus est en attente d'un périphérique (entrée-sortie)
- Il doit donc attendre après ce périphérique!
 - Jusqu'à ce que ce dernier soulève une interruption pour indiquer au système d'exploitation qu'il a terminé la tâche qu'on lui a demandé de faire
 - Le processus est alors remis dans l'état **prêt**

Un processus... dans tous ses états



Un processus... dans tous ses états



Ordonnancement — Linux

- Algorithme: « Completely Fair Scheduler »
- Assigne un « score » à chaque processus basé sur une comparaison avec un OS idéal
 - OS idéal: donne le *même* temps de traitement à tous les processus
- Le score est la différence entre:
 - le temps de traitement réellement alloué au processus
 - le temps qu'un « OS idéal » aurait alloué au processus
- Plus cette différence est élevée, plus un processus a été laissé de côté.
- Les scores des processus sont ordonnés (grâce à un arbre «rouge-noir»), et le processus avec le plus grand score est choisi.

À retenir

- L'utilisation d'une **interruption** pour l'**ordonnancement** des processus
 - Sauvegarde et restauration du **contexte** de chaque processus
- Les **états** d'un processus:
 - nouveau, prêt, en cours, bloqué, terminé
- Algorithmes d'ordonnancement:
 - premier-arrivé/premier-servi, plus court d'abord, priorité, **tourniquet**

1. S'il y a un nouveau processus:
 - 1.1. L'admettre en mémoire;
 - 1.2. (**Tourniquet seulement**): placer le processus à la fin de la file d'attente *correspondant à sa priorité.*
2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnancement;
3. Calculer la durée restante du processus choisi;
 - 3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;
 - 3.2. (**Tourniquet seulement**): Sinon, placer le processus à la fin de la file d'attente;
4. Passer au quanta suivant.