La gestion des processus

GIF-1001 Ordinateurs: Structure et Applications

Jean-François Lalonde

Image: fugly.com

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

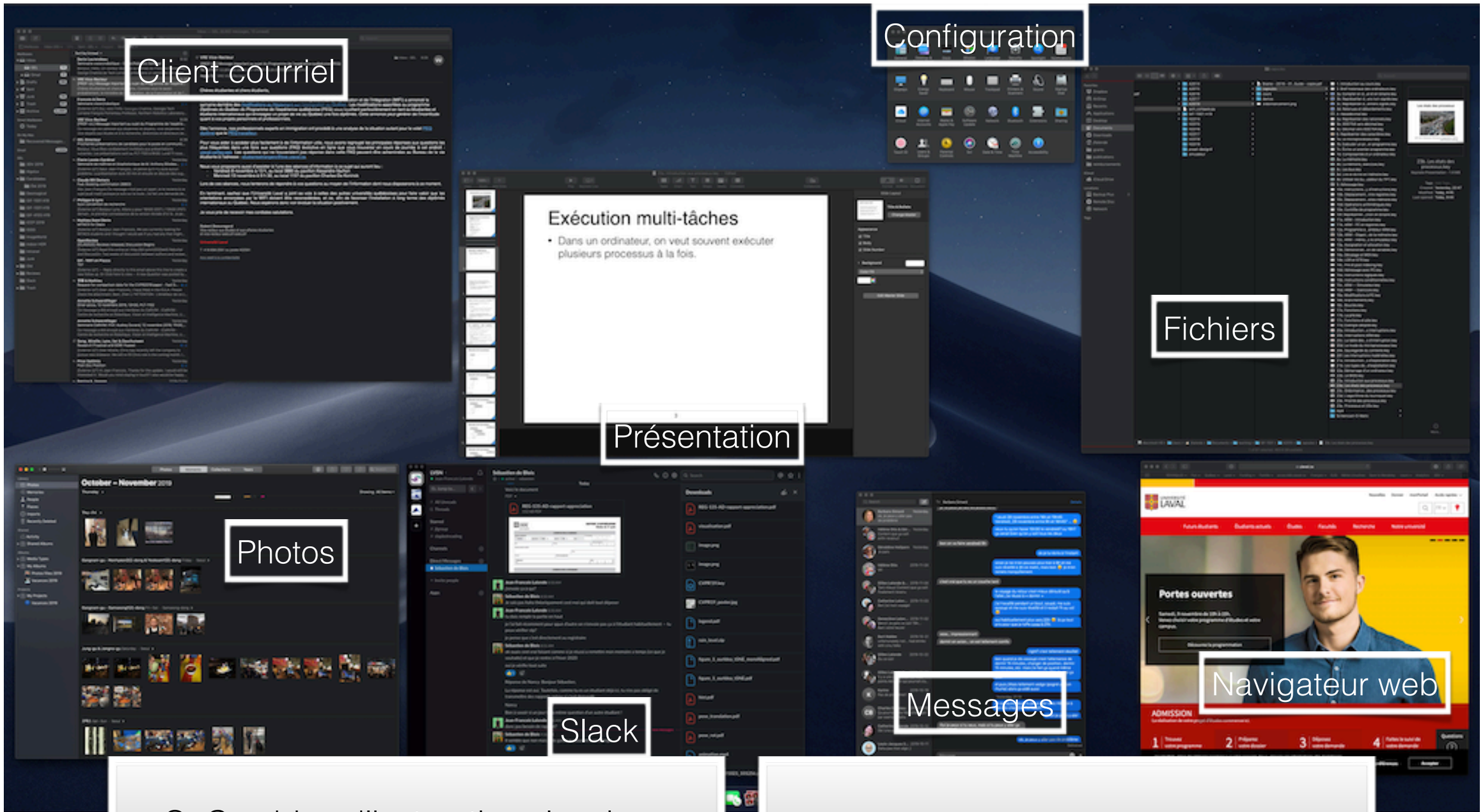
Crédit: P. Giguère

Programmes et processus

- Un **programme** est un ensemble d'instructions et de variables dont le but est d'accomplir une tâche précise.
  - C'est donc le code que l'on écrit et qui est traduit en langage binaire par le compilateur.
- Un **processus** est composé
  - du **programme**;
  - et de l'ensemble des **ressources** reliées à l'exécution du programme. Ces ressources incluent:
    - un espace réservé en mémoire;
    - des fichiers ouverts;
    - du temps de CPU.

Exécution multi-tâches

- Dans un ordinateur, on veut souvent exécuter plusieurs processus à la fois.



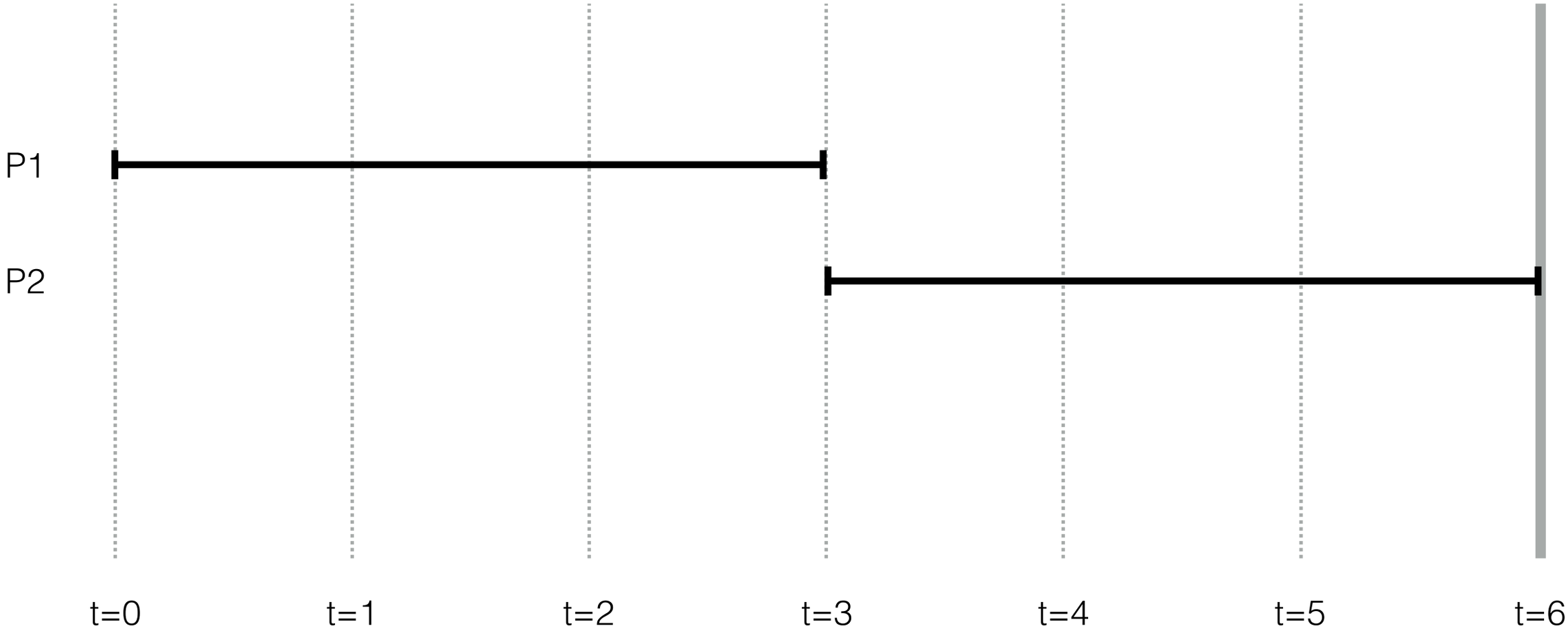
Q. Combien d'instructions le micro-processeur peut-il exécuter à la fois?

R. Une seule!

Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. Chacun dure 3 unités de temps.

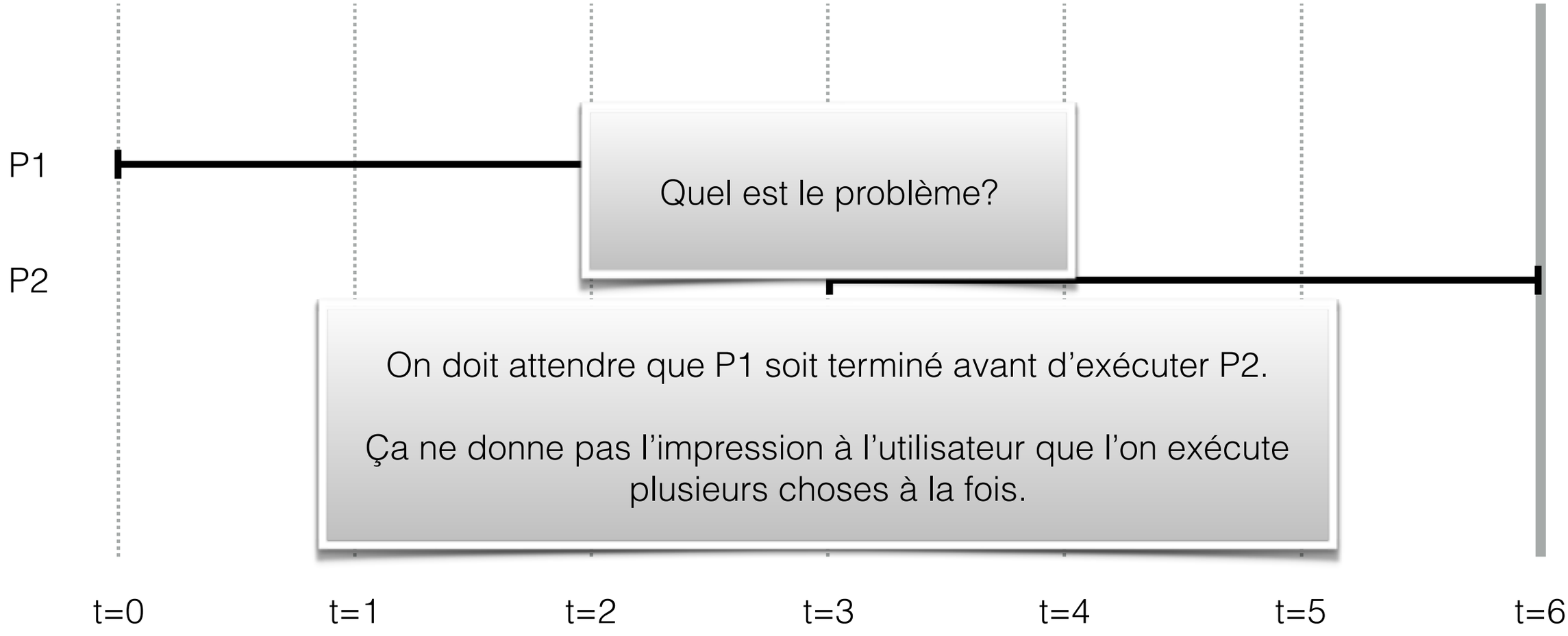
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on exécute P1, puis on exécute P2 lorsque P1 est terminé



Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. Chacun dure 3 unités de temps.

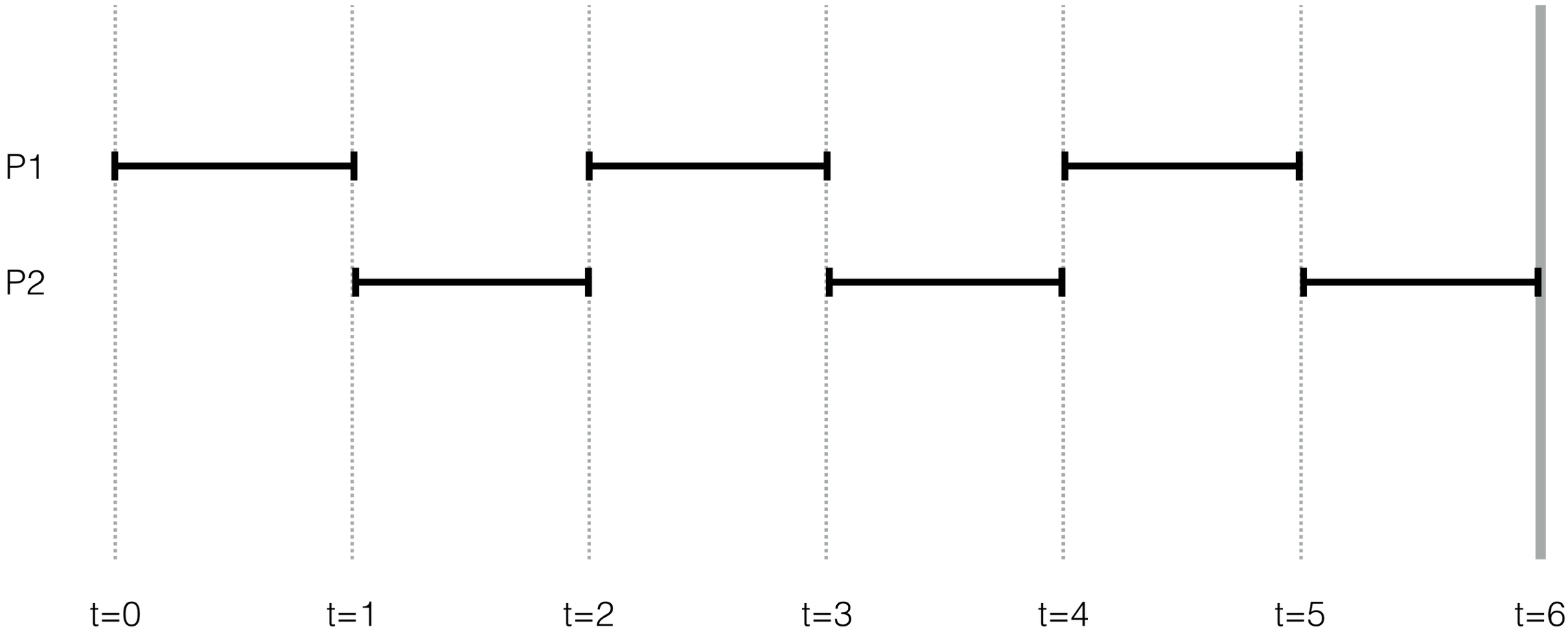
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on exécute P1, puis on exécute P2 lorsque P1 est terminé



Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. Chacun dure 3 unités de temps.

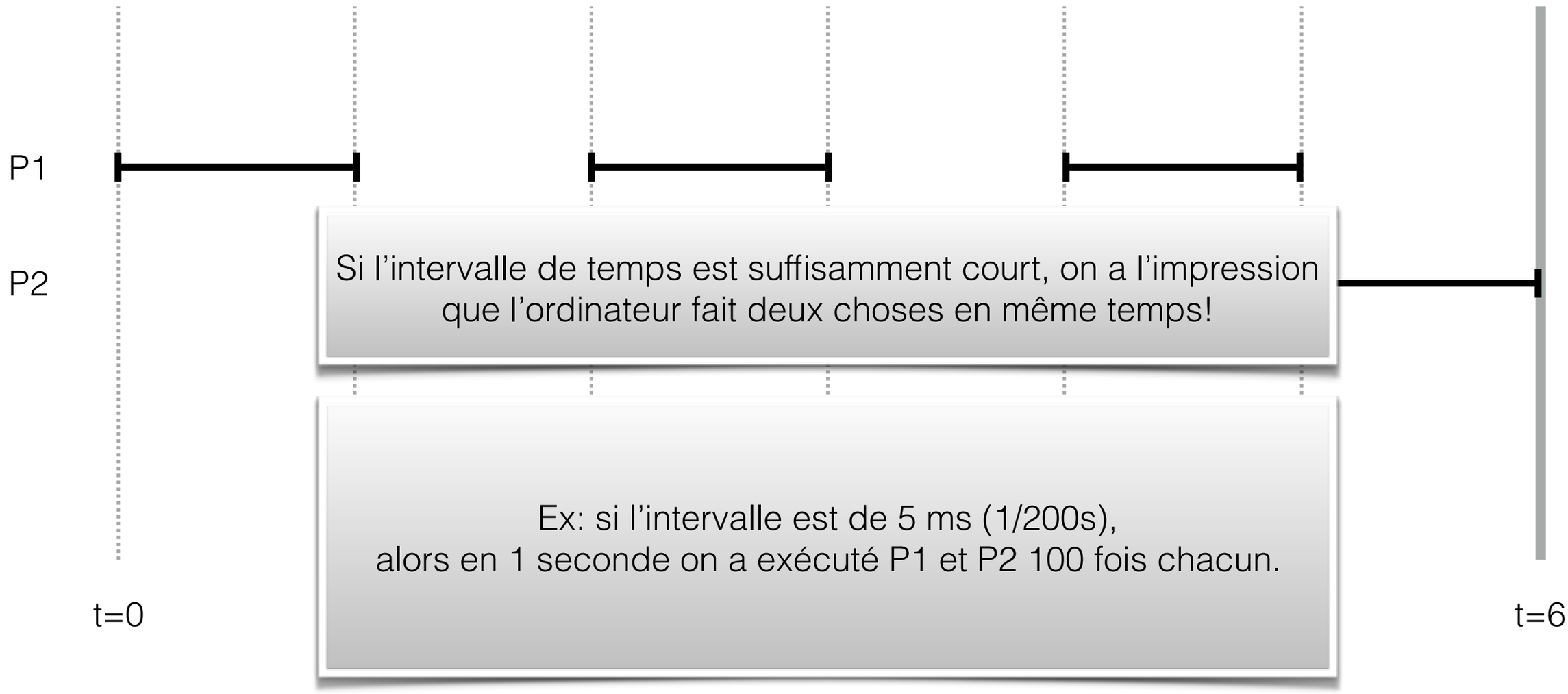
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on **les exécute en alternance**.



Exécution de 2 processus

- Deux processus, P1 et P2, doivent être exécutés. Chacun dure 3 unités de temps.

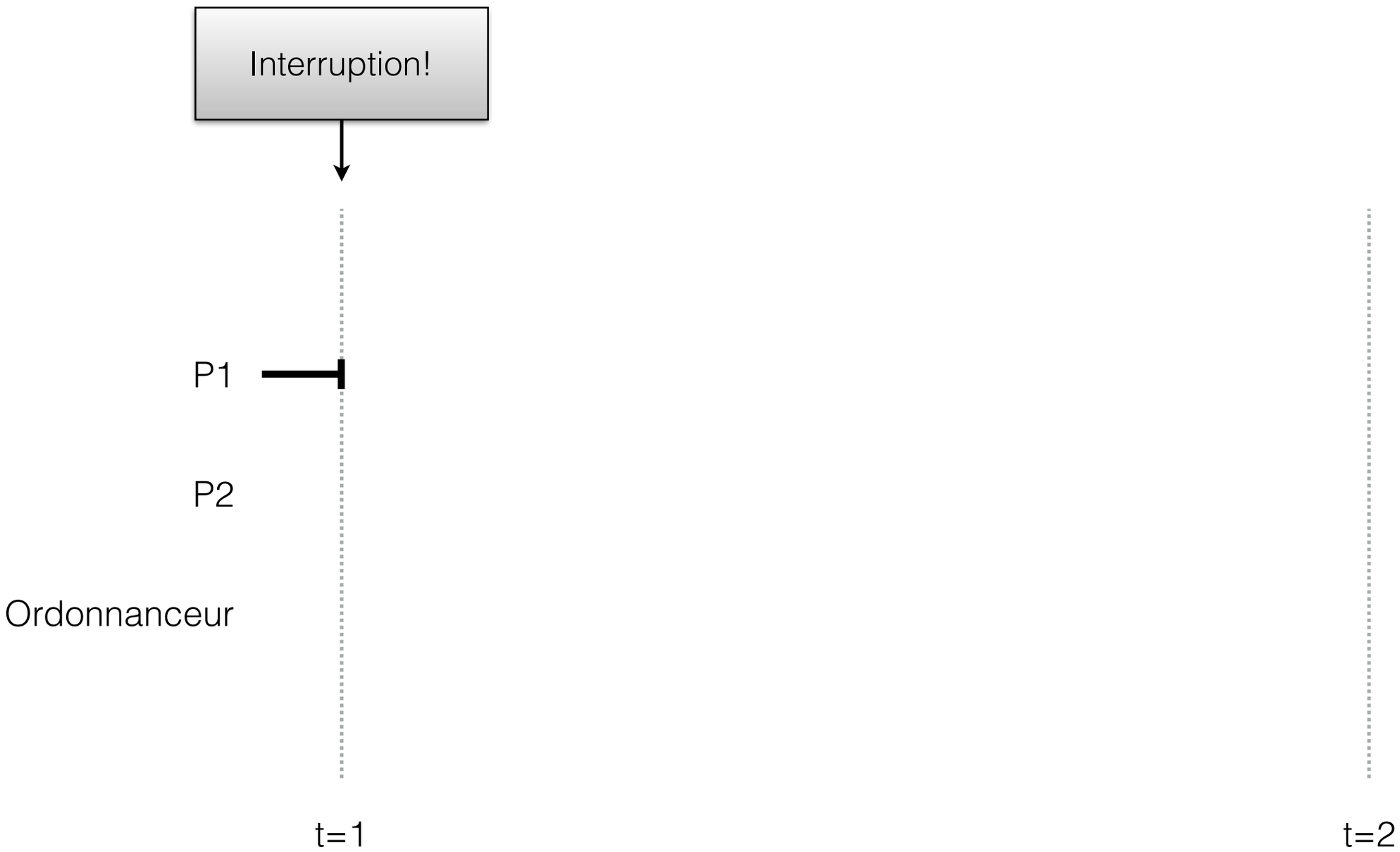
- Comme le microprocesseur ne peut faire qu'une chose à la fois, on **les exécute en alternance**.



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

Sauvegarde le contexte du processus P1

P1

P2

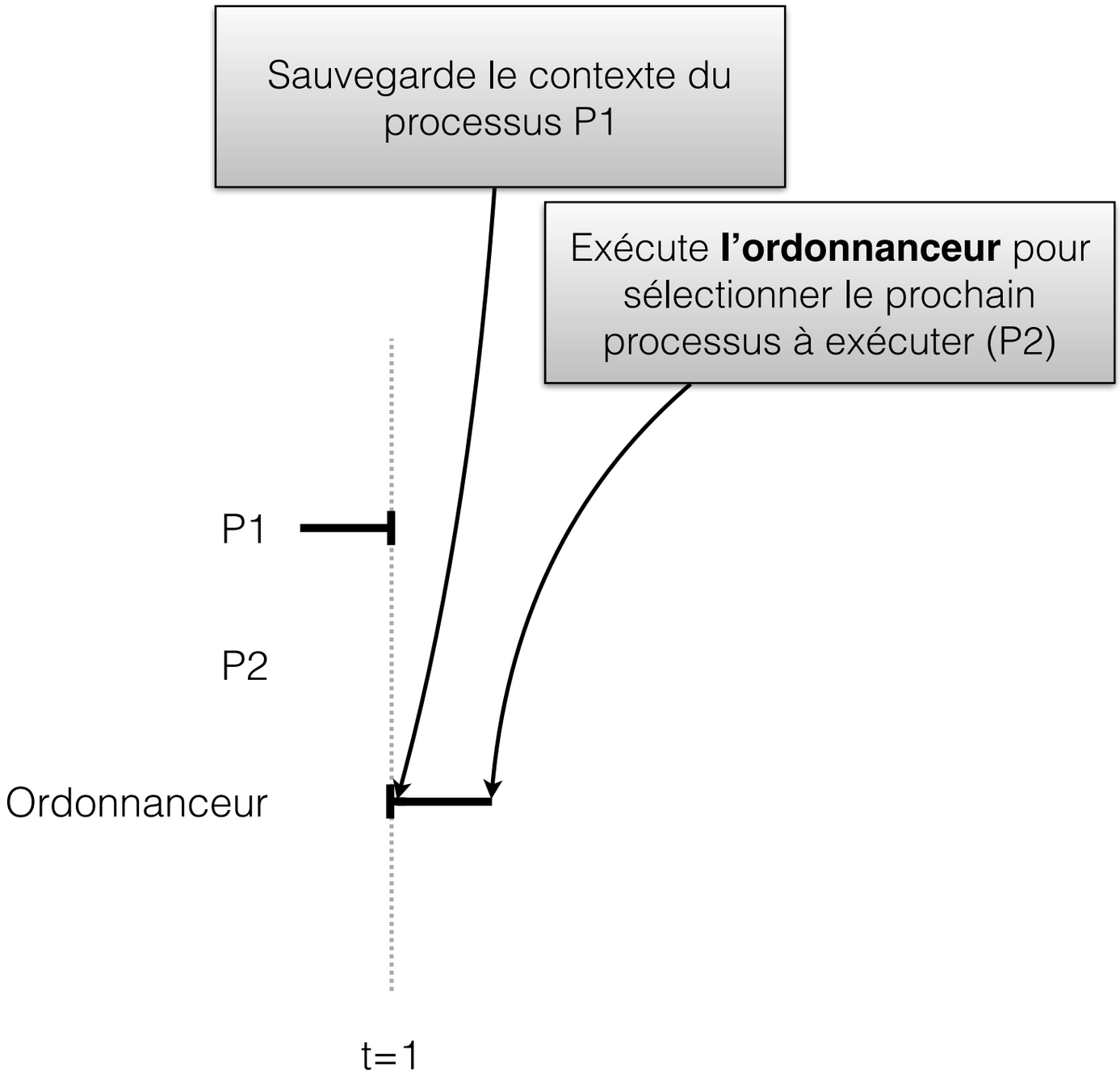
Ordonnanceur

t=1

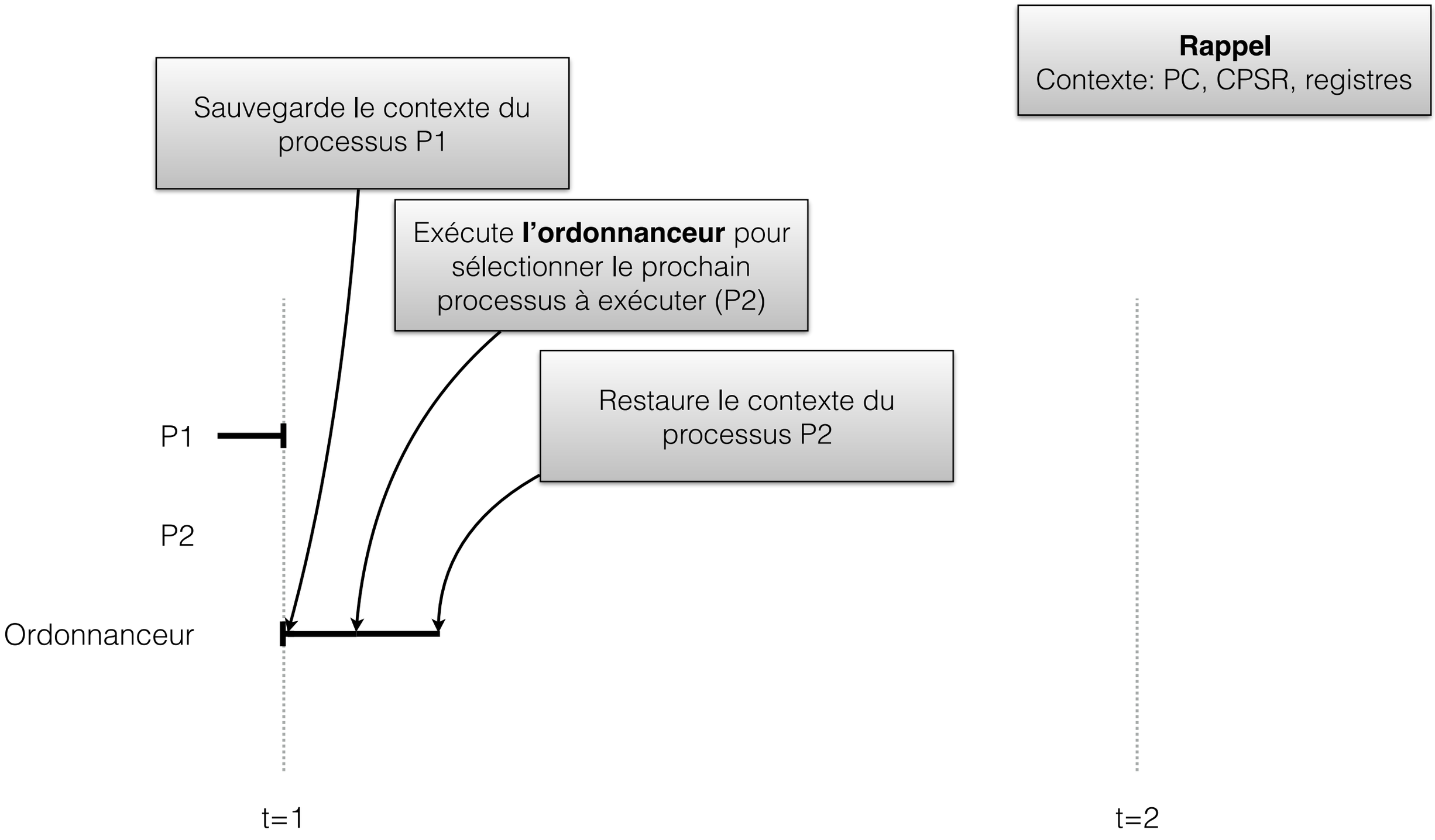
t=2

Exécution de 2 processus

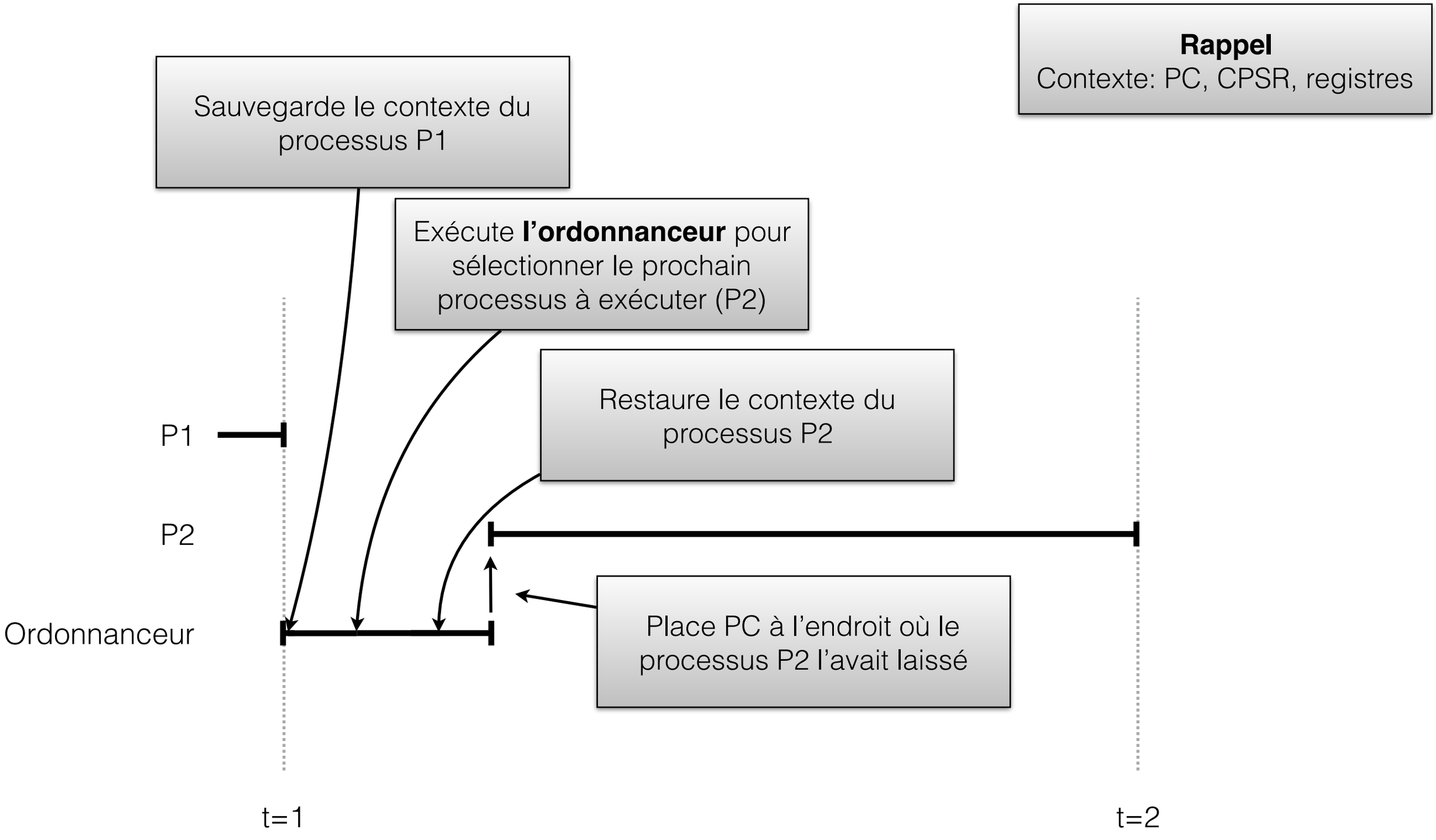
Rappel
Contexte: PC, CPSR, registres



Exécution de 2 processus

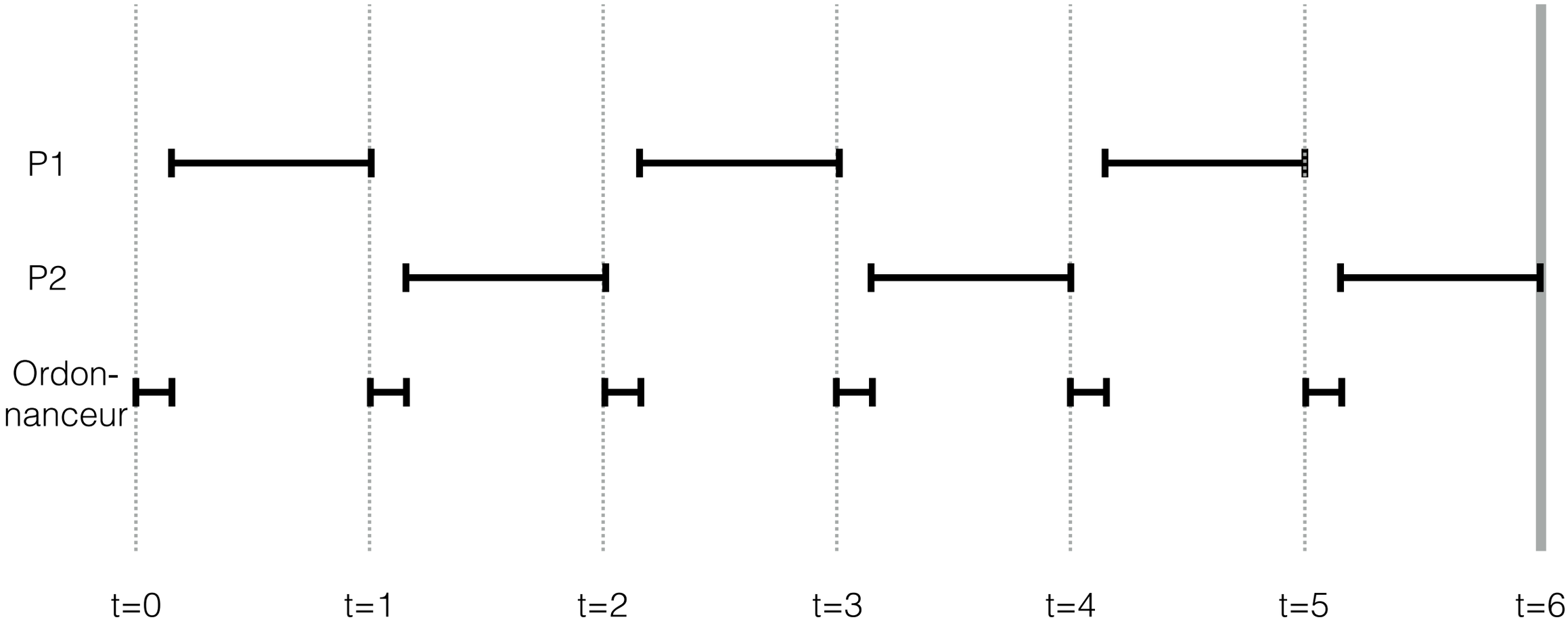


Exécution de 2 processus



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 une liste de tous les processus à exécuter, ainsi que leur état.

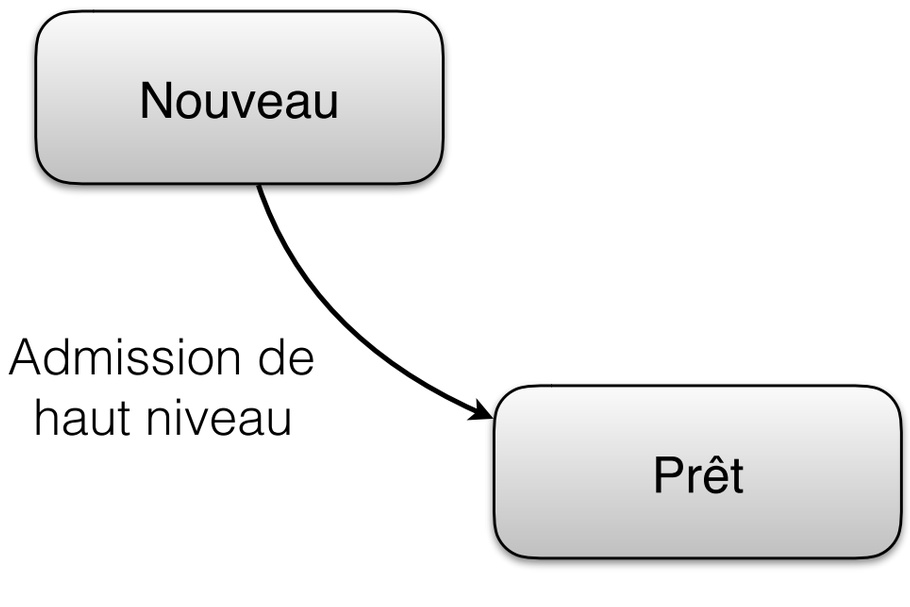
  - le SE stocke cette information dans un ***Process Control Block*** (PCB) pour chaque processus, qui contient:

    - 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

- Un nouveau processus peut être démarré par:
  - une requête de l'usager (ex: exécution de programme);
  - le système d'exploitation;
  - ou un autre processus.
- 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 majorité du temps), le processus est admis et tombe dans l'état **prêt**.
  - Sinon, le processus ne peut être démarré.

Un processus… dans tous ses états



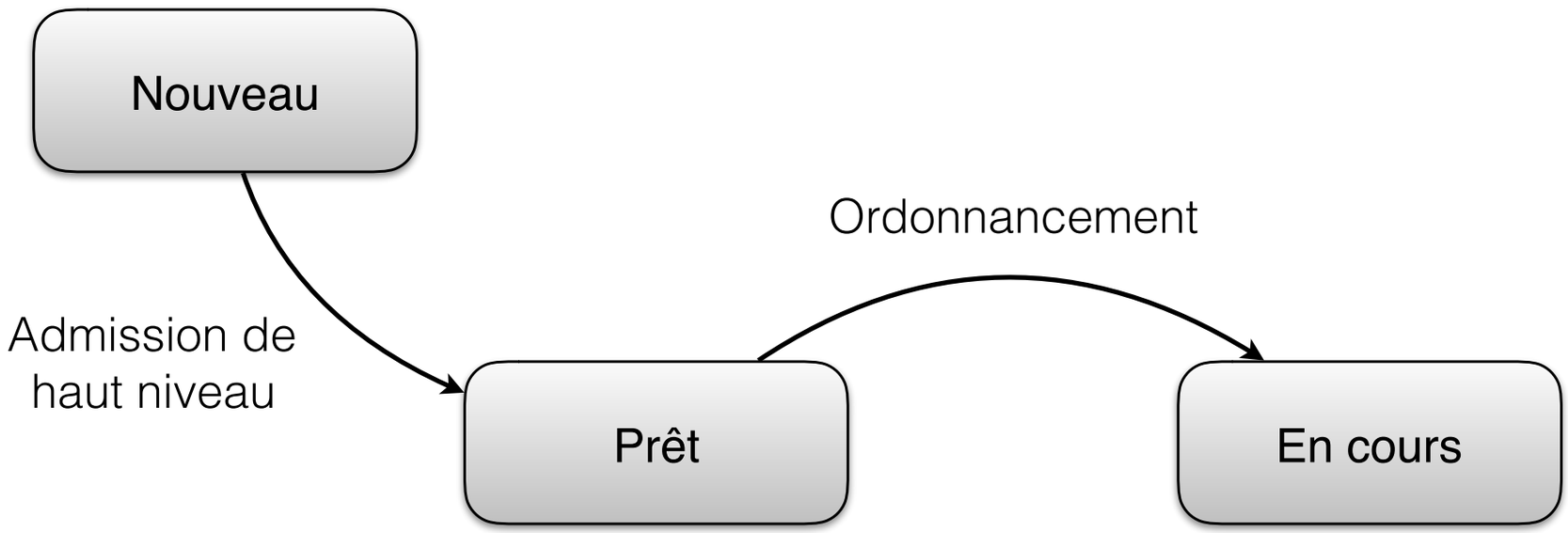
Processus: état « prêt »

- Le processus est **prêt** d'être exécuté, mais il ne peut pas: il est **en attente du micro-processeur**
  - Le micro-processeur est alors occupé à exécuter un autre processus
- 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…)

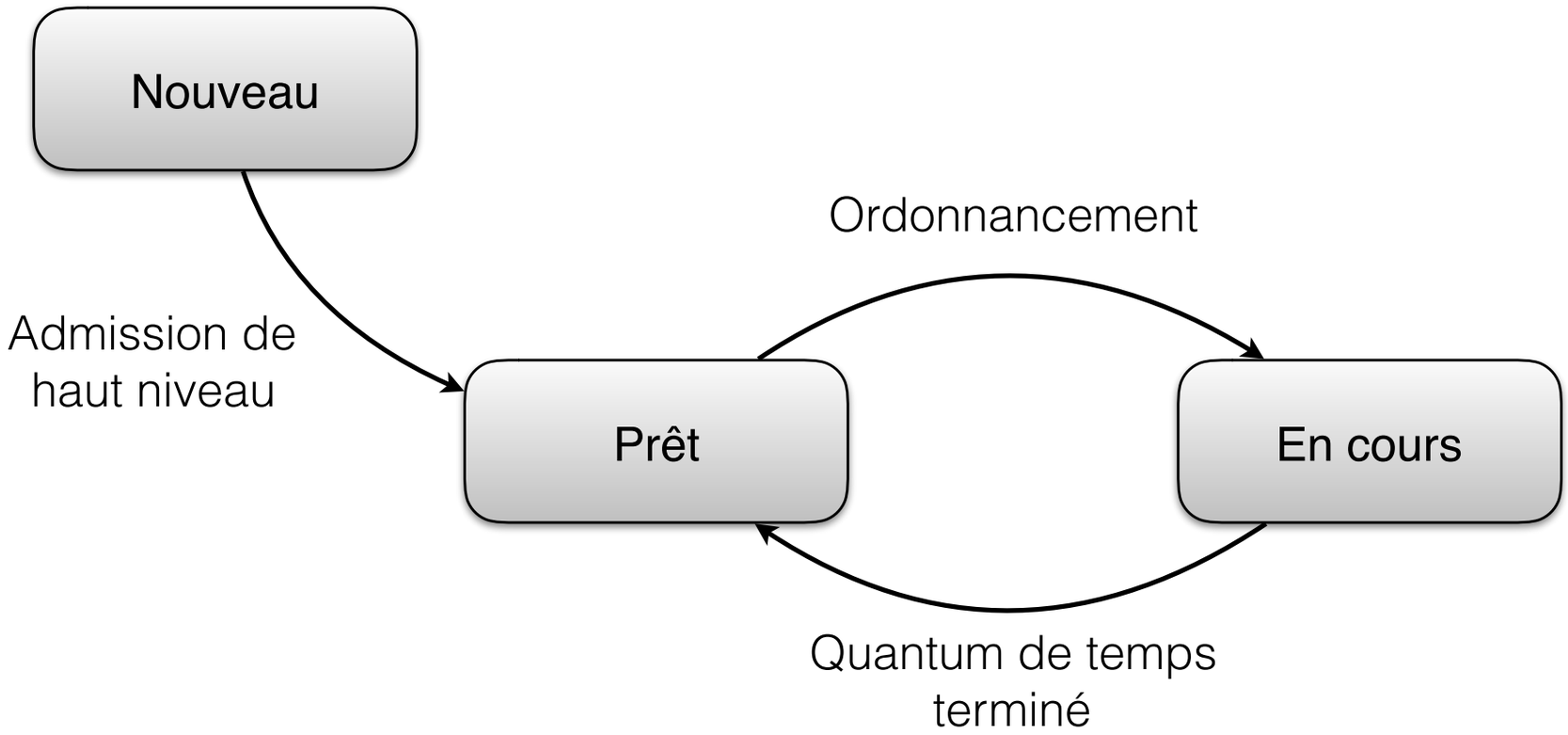
Un processus… dans tous ses états



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**

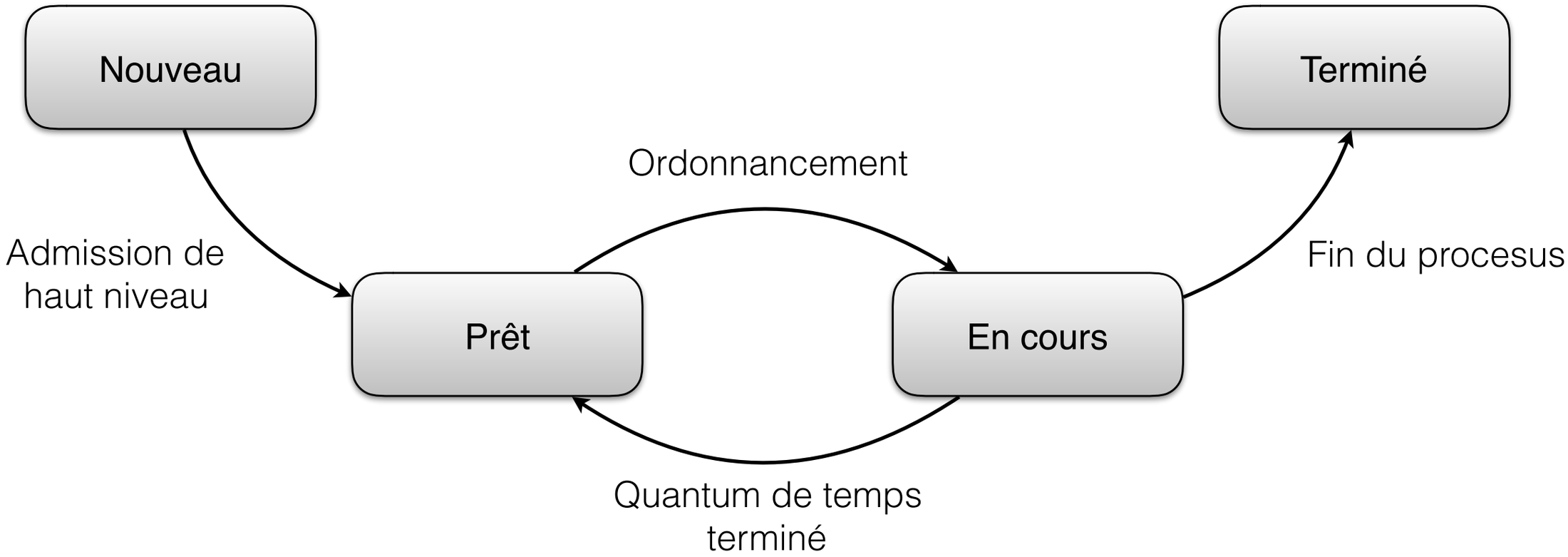
Un processus… dans tous ses états



Processus: état « terminé »

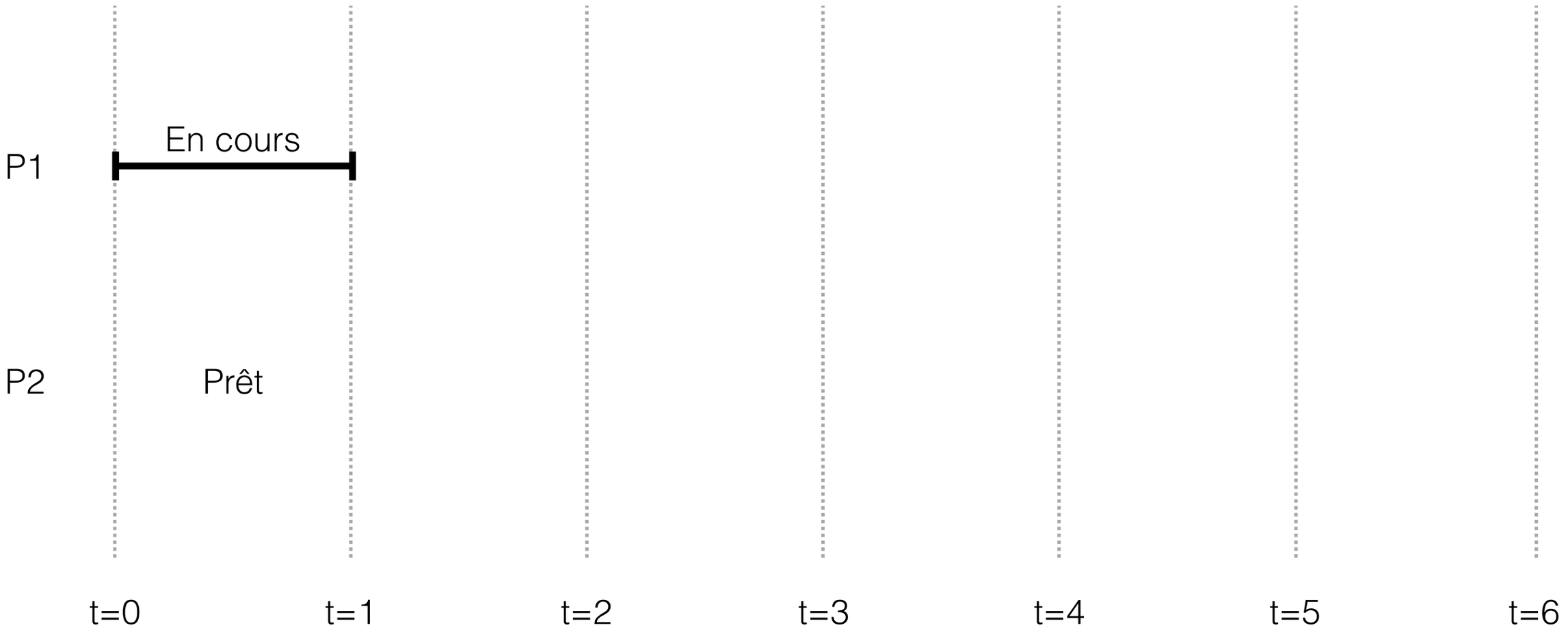
- Lorsque le processus a complété son exécution, ou lorsqu'on le ferme volontairement, il devient dans l'état **terminé**.

Un processus… dans tous ses états



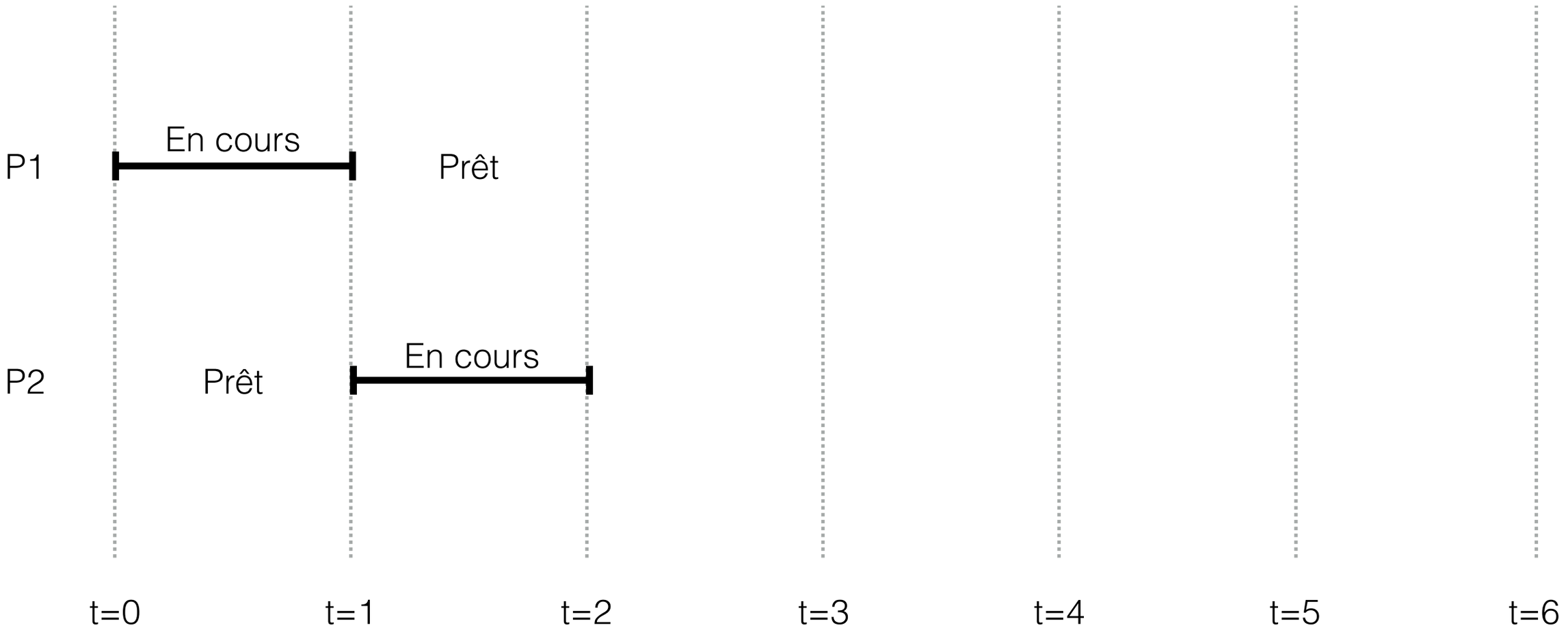
Exécution de 2 processus

- 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.



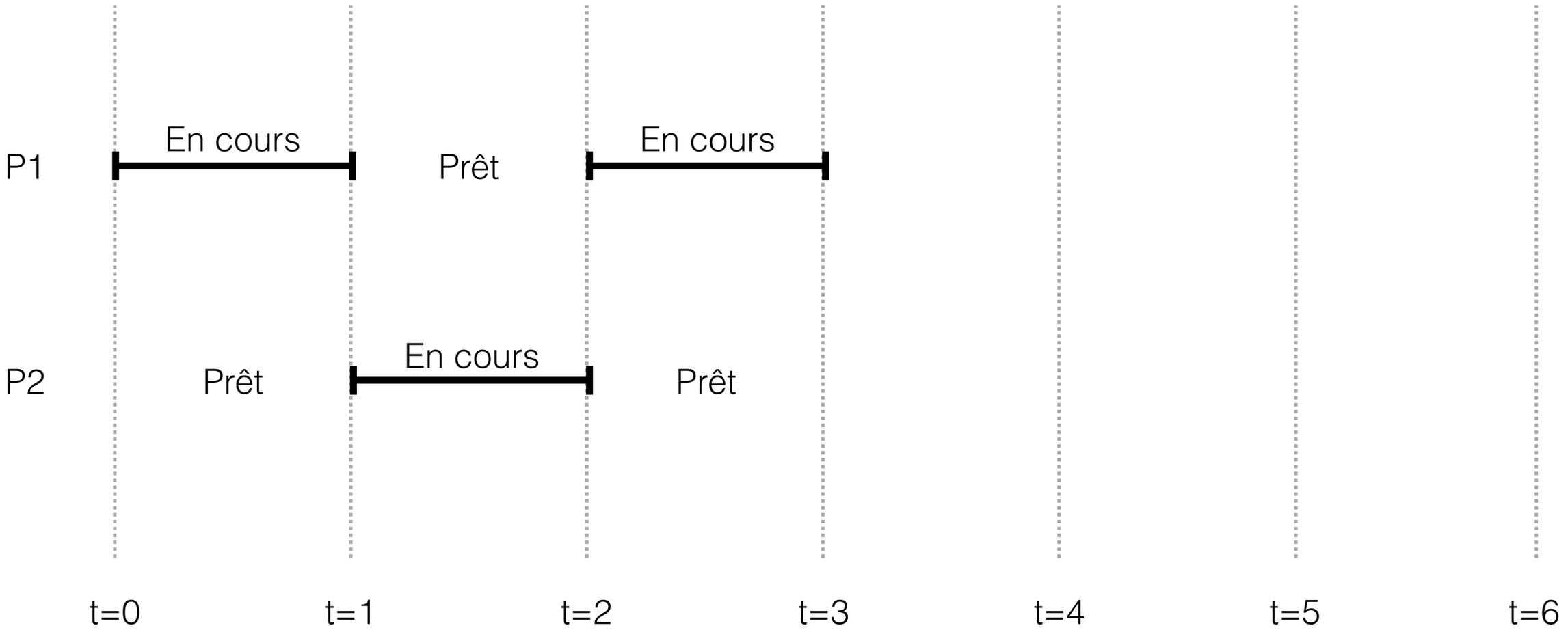
Exécution de 2 processus

- 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.



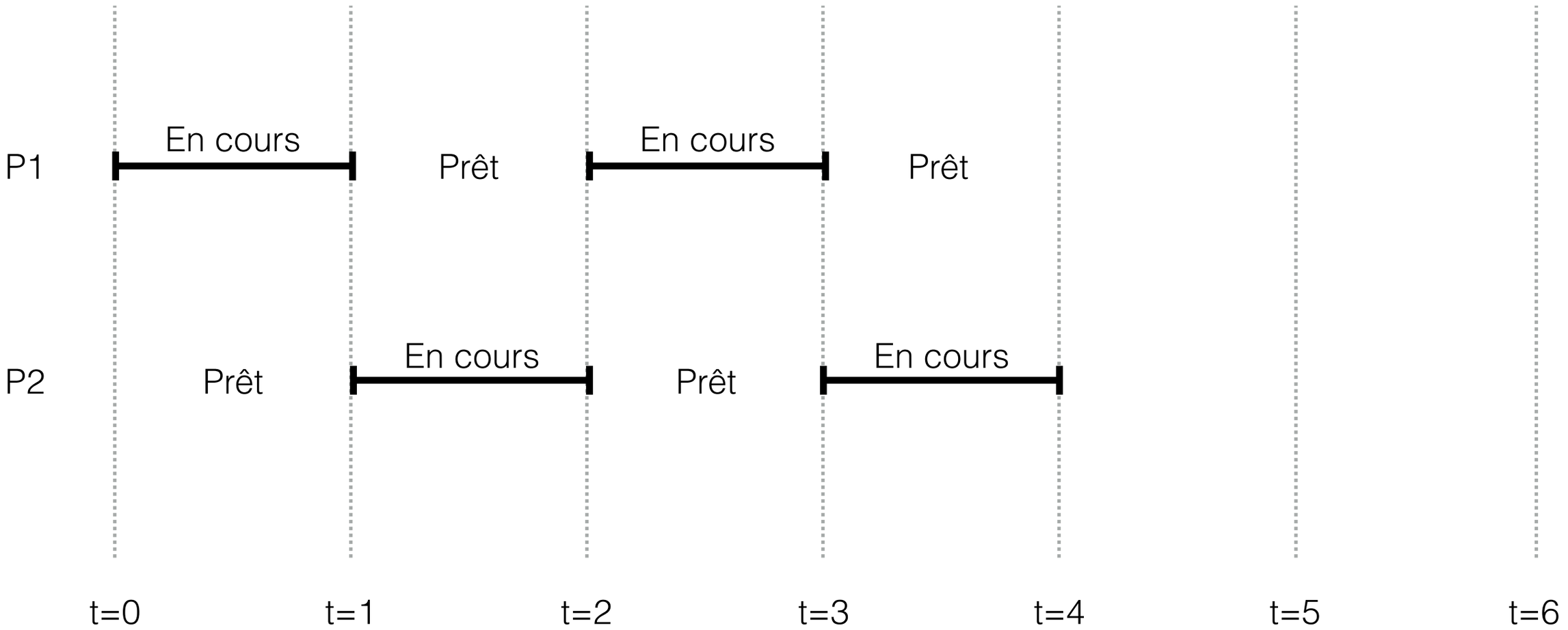
Exécution de 2 processus

- 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.



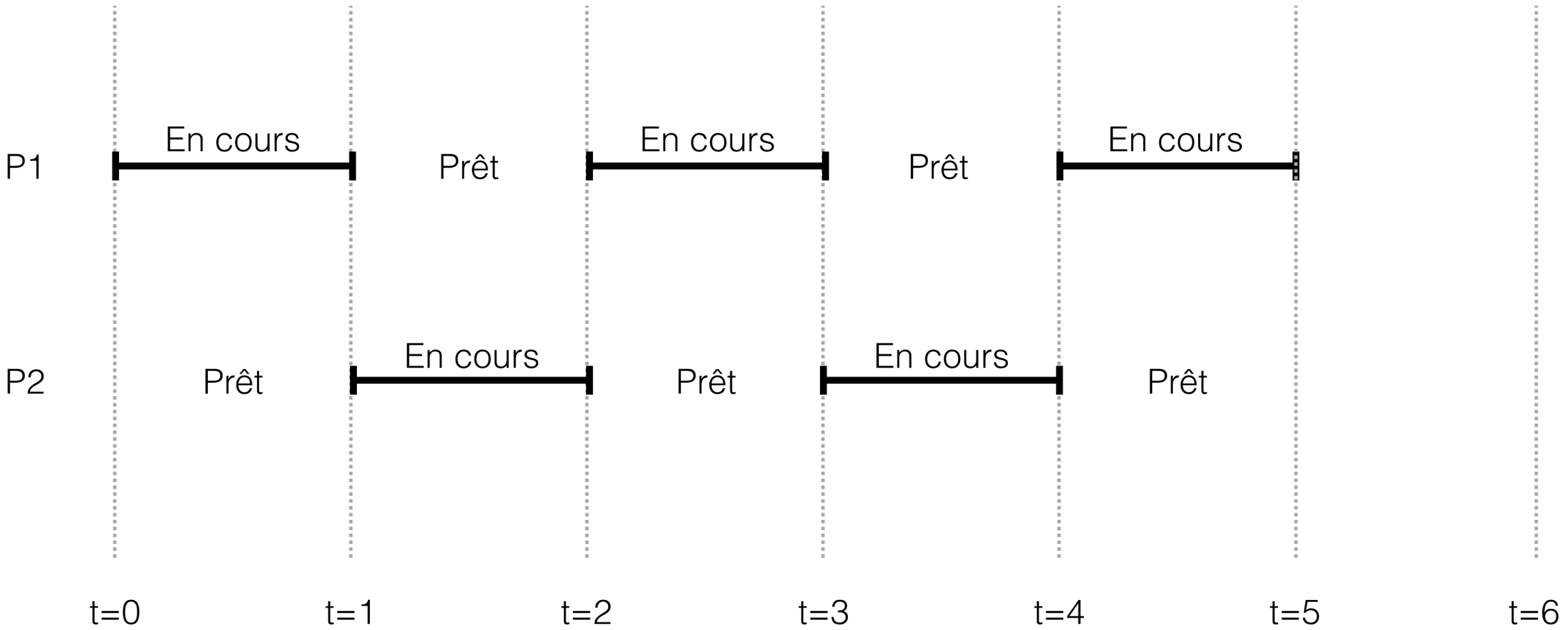
Exécution de 2 processus

- 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.



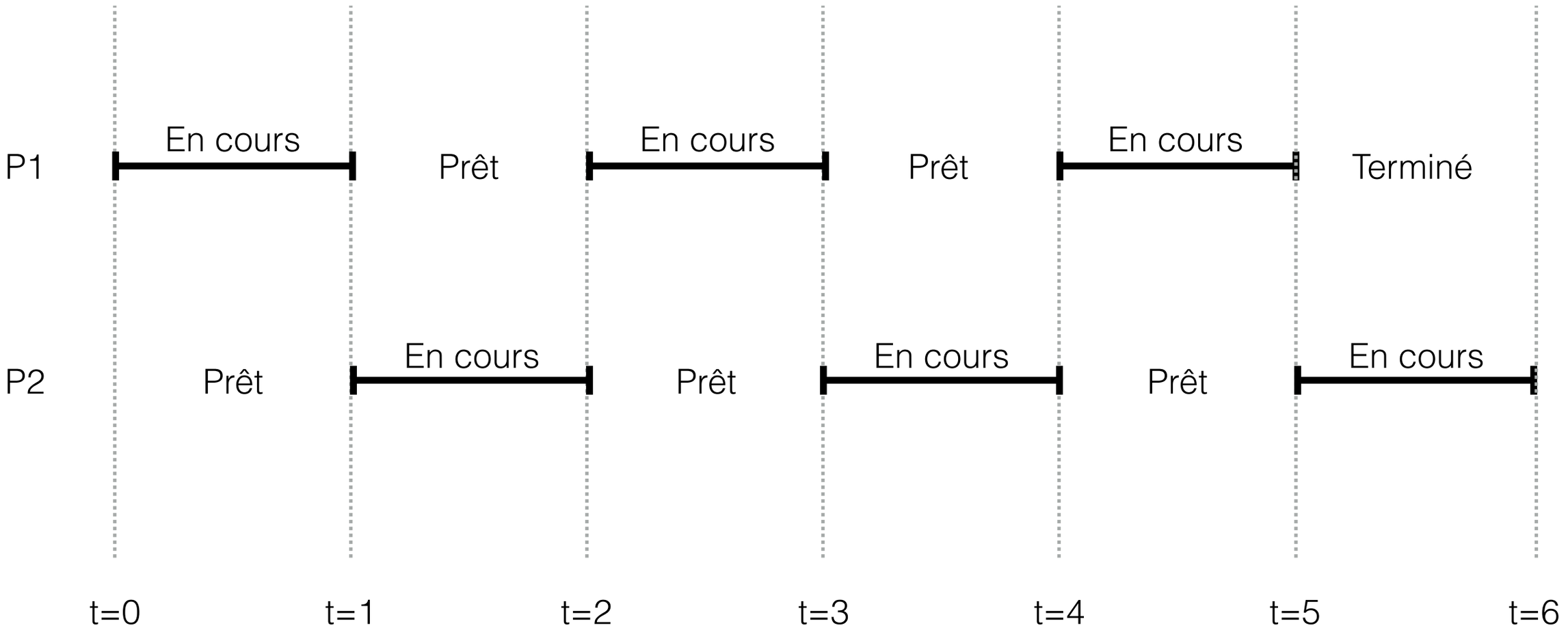
Exécution de 2 processus

- 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.



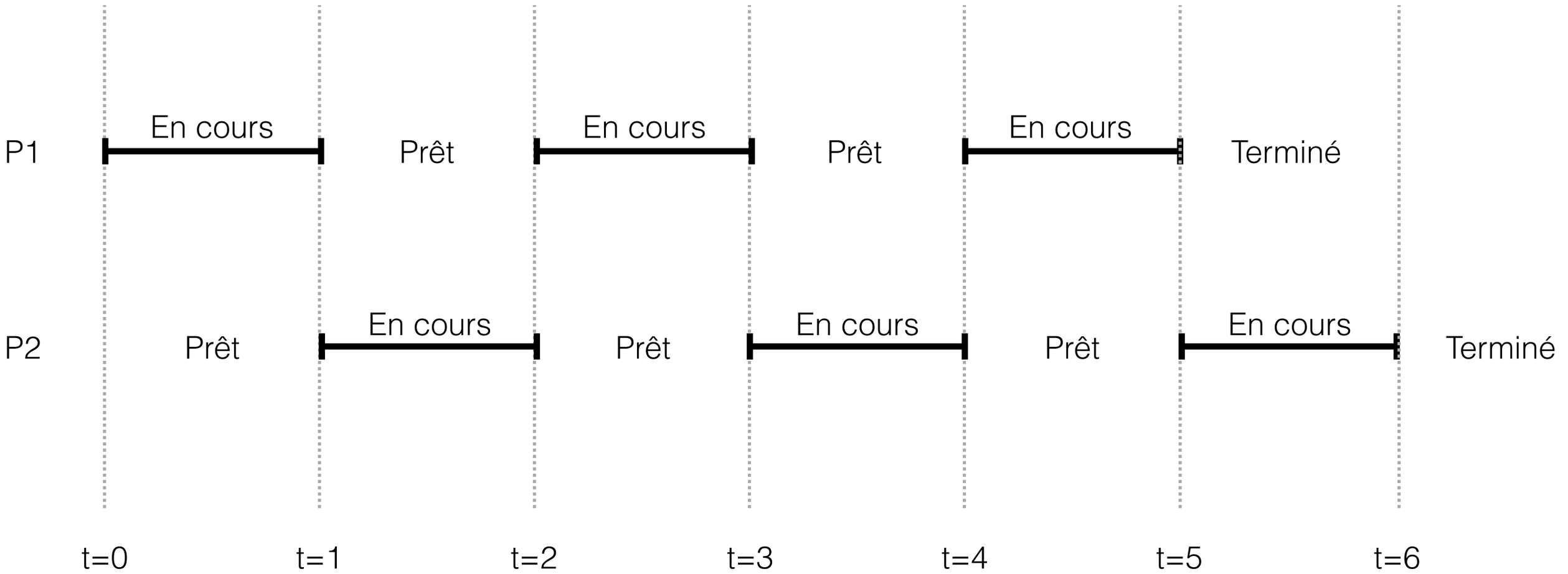
Exécution de 2 processus

- 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.

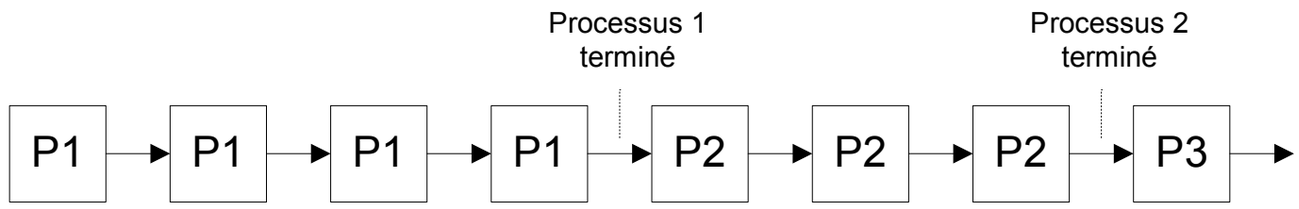
Analogie de la vie de tous les jours™

NETFLIX

Ordonnancement: algorithme général

1. S'il y a un nouveau processus:
    1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

Algorithmes d’ordonnancement

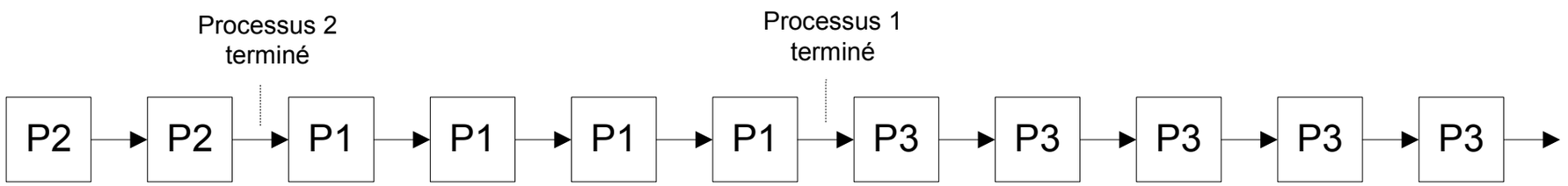


- **Premier arrivé, premier servi**
(exemple de **mauvais** algorithme)

  - Le premier processus admis est exécuté jusqu’à sa fin. Puis, on exécute le suivant.

  - Cet algorithme n’est jamais utilisé. Il détruirait l’illusion d’une exécution simultanée de plusieurs processus.

Algorithmes d’ordonnancement



- **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

Exercice #1

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

Nom	Durée
P1	5
P2	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

Exercice #1: premier arrivé premier servi

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
   1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

Solution #1: premier arrivé premier servi

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
   1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

P1—P1—P1—P1—P1—P2—P2—P2

Exercice #1: plus court d'abord

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
   1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

Solution #1: plus court d'abord

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
   1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

P2—P2—P2—P1—P1—P1—P1—P1

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

Exercice #2

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

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	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

Exercice #2: premier arrivé premier servi

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

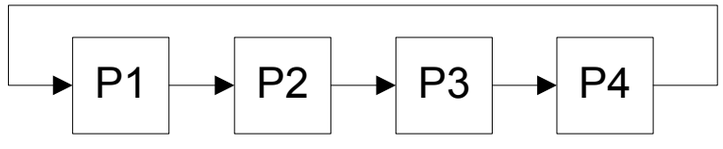
1. S'il y a un nouveau processus:
    1.1. L'admettre en mémoire;
2. Choisir le processus parmi ceux admis en mémoire selon l'algorithme d'ordonnancement;
3. Calculer la durée restante du processus choisi;
    3.1. Si le processus choisi est terminé, le retirer de la liste des processus en mémoire;
4. Passer au quanta suivant.

Exercice #2: plus court d'abord

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

1. S'il y a un nouveau processus:
   1.1. L'admettre en mémoire;
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;
4. Passer au quanta suivant.

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;

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;

4. Passer au quanta suivant.

Ordonnancement: algorithme

1. S'il y a un nouveau processus:
    1.1. L'admettre en mémoire;

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):

Nom	Durée
P1	5
P2	3

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

Exercice #1

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
    1.1. L'admettre en mémoire;
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.

Solution #1

Nom	Durée
P1	5
P2	3

1. S'il y a un nouveau processus:
    1.1. L'admettre en mémoire;
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.

P1—P2—P1—P2—P1—P2—P1—P1

Arrivée dynamique des processus

- En pratique, les processus arrivent à différents moments. Il faut donc recalculer quel processus sera exécuté en prochain.
- Pour le tourniquet:
  - On maintient une file d'attente.
  - Lorsqu'un nouveau processus arrive, on le place à la *fin* de la file d'attente.

Ordonnancement: algorithme

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
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.

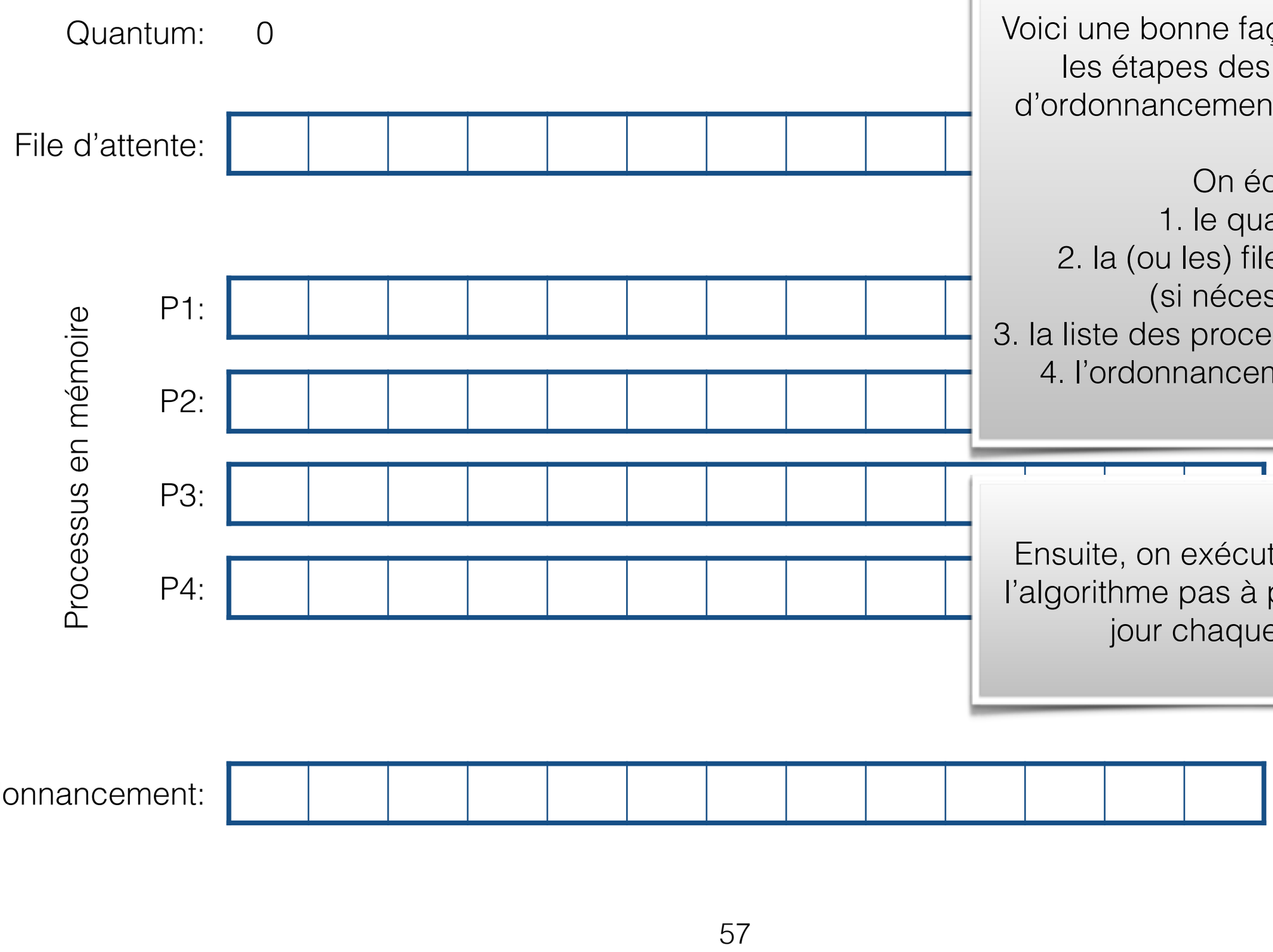
Exercice #2

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

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

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

Exercice #2



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

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 0

File d’attente:

Processus en mémoire

Ordonnancement:

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 0 (admission)

File d'attente:

P1 et P2 sont admis

Processus en mémoire

P1:

P2:

Ordonnancement:

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 0 (ordonnancement)

File d'attente:

~~P1~~	P2	P1										

Processus en mémoire

P1:	~~5~~	4											
P2:	4												

Ordonnancement:

P1												

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 1 (admission)

File d'attente:

~~P1~~	P2	P1										

Processus en mémoire

P1:

~~5~~	4											

P2:

4												

Ordonnancement:

P1												

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 1 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	P1	P2									

Processus en mémoire

P1:

~~5~~	4											

P2:

~~4~~	3											

Ordonnancement:

P1	P2											

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 2 (admission)

File d'attente:

~~P1~~	~~P2~~	P1	P2	P3								

P3 est admis

Processus en mémoire

P1:	~~5~~	4											
P2:	~~4~~	3											
P3:	1												

Ordonnancement:

P1	P2											

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 2 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	P2	P3	P1							

Processus en mémoire

P1:

~~5~~	~~4~~	3										

P2:

~~4~~	3											

P3:

1												

Ordonnancement:

P1	P2	P1										

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 3 (admission)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	P2	P3	P1							

Processus en mémoire

P1:

~~5~~	~~4~~	3										

P2:

~~4~~	3											

P3:

1												

Ordonnancement:

P1	P2	P1										

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 3 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	P3	P1	P2						

Processus en mémoire

P1:

~~5~~	~~4~~	3										

P2:

~~4~~	~~3~~	2										

P3:

1												

Ordonnancement:

P1	P2	P1	P2									

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 4 (admission)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	P3	P1	P2						

Processus en mémoire

P1:	~~5~~	~~4~~	3										
P2:	~~4~~	~~3~~	2										
P3:	1												

Ordonnancement:

P1	P2	P1	P2									

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 4 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	~~P3~~	P1	P2						

Processus en mémoire

P1:

~~5~~	~~4~~	3										

P2:

~~4~~	~~3~~	2										

~~P3:~~

~~1~~	0											

P3 se termine

Ordonnancement:

P1	P2	P1	P2	P3								

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 5 (admission)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	~~P3~~	P1	P2	P4					

P4 est admis

Processus en mémoire

P1:	~~5~~	~~4~~	3									
P2:	~~4~~	~~3~~	2									
P4:	3											

Ordonnancement:

P1	P2	P1	P2	P3								

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 5 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	~~P3~~	~~P1~~	P2	P4	P1				

Processus en mémoire

P1:

~~5~~	~~4~~	~~3~~	2									

P2:

~~4~~	~~3~~	2										

P4:

3												

Ordonnancement:

P1	P2	P1	P2	P3	P1							

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 6 (ordonnancement)

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									

P4:

3												

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2						

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 7 (ordonnancement)

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									

P4:

~~3~~	2											

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4					

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 8 (ordonnancement)

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									

P4:

~~3~~	2											

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1				

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 9 (ordonnancement)

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											

P2 se termine

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2			

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 10 (ordonnancement)

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								

P4:

~~3~~	~~2~~	1										

Ordonnancement:

P1	P2	P1	P2	P3	P1	P2	P4	P1	P2	P4		

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 11 (ordonnancement)

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							

P1 se termine

P4:

~~3~~	~~2~~	1										

Ordonnancement:

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

Exercice #2

Nom	Durée	Arrivée
P1	5	0
P2	4	0
P3	1	2
P4	3	5

Quantum: 12 (ordonnancement)

File d'attente:

~~P1~~	~~P2~~	~~P1~~	~~P2~~	~~P3~~	~~P1~~	~~P2~~	~~P4~~	~~P1~~	~~P2~~	~~P4~~	~~P1~~	~~P4~~

Processus en mémoire

~~P4:~~	~~3~~	~~2~~	~~1~~	0								

P4 se termine

Ordonnancement:

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

Algorithmes d’ordonnancement

- **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’ordonnancement

- **Tourniquet avec priorité**: On exécute tout d’abord les processus selon leur priorité. Si plusieurs processus ont la même priorité, on utilise le tourniquet pour les exécuter en alternance.
  - 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

Exercice #1

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

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

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

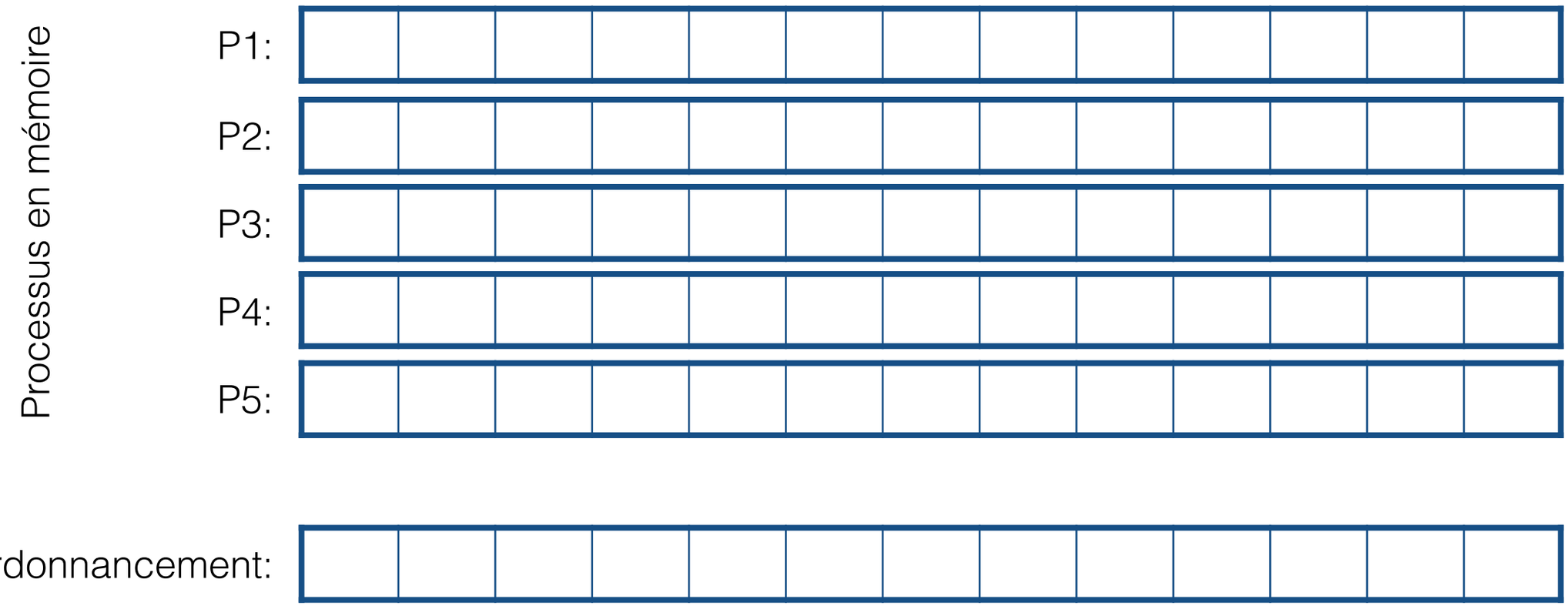
Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 0

Lorsque l'on doit tenir compte de la priorité, on maintient **une file d'attente par niveau de priorité**.

On choisit toujours le processus dans la file d'attente la plus prioritaire.

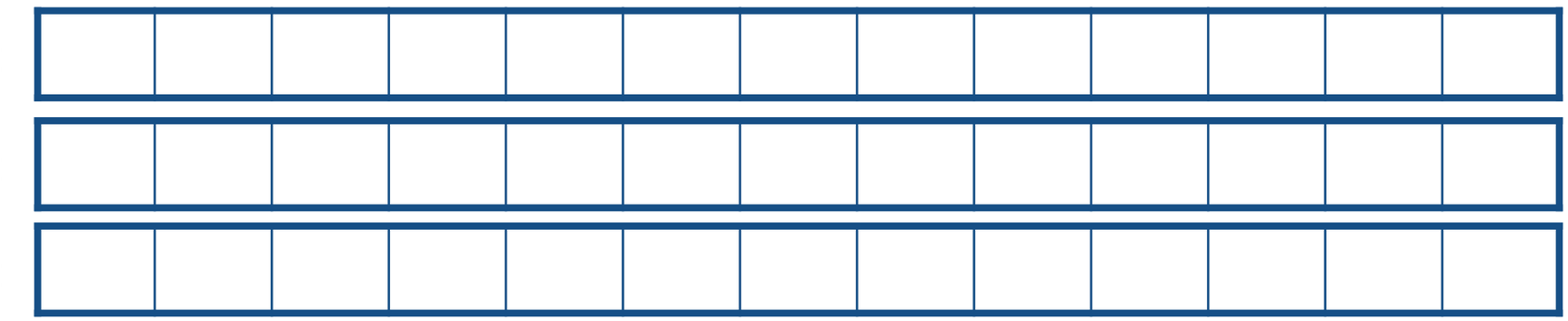


Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 0

Files d'attente

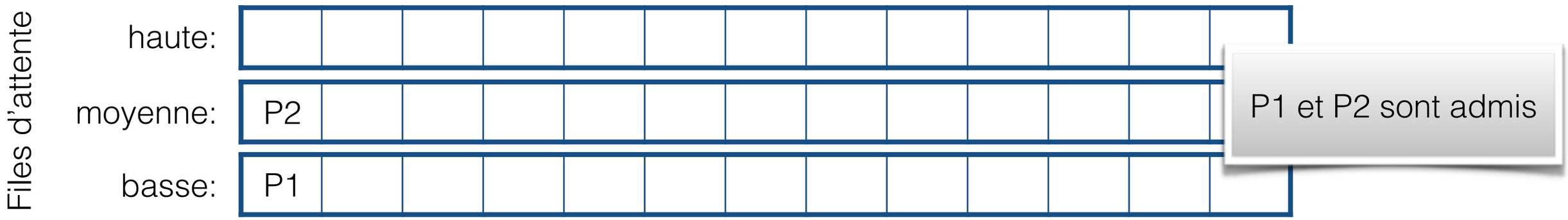


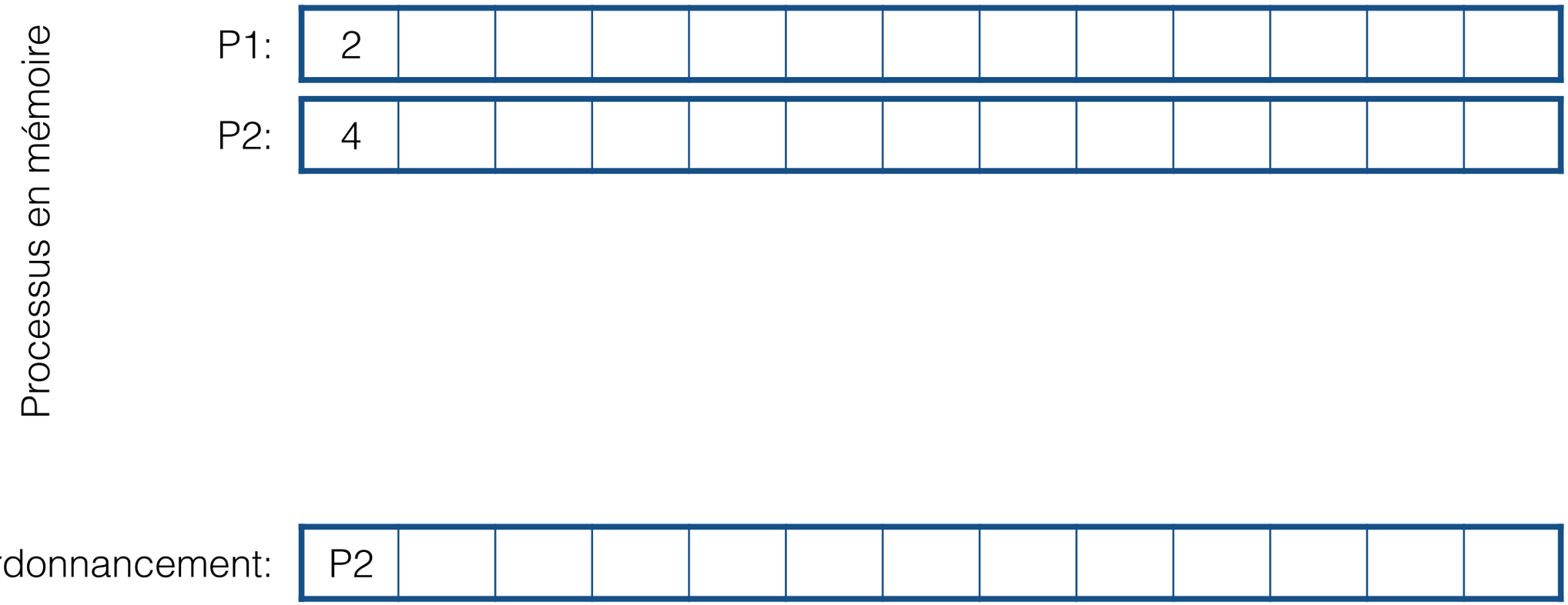
Processus en mémoire

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 0 (admission)





Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 0 (ordonnancement)

Files d'attente

haute:													
moyenne:	~~P2~~	P2											
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	3											

Ordonnancement:	P2												

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 1 (admission)

Files d'attente

haute:													
moyenne:	~~P2~~	P2											
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	3											

Ordonnancement:	P2												

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 1 (ordonnancement)

Files d'attente

haute:													
moyenne:	~~P2~~	~~P2~~	P2										
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	~~3~~	2										

Ordonnancement:	P2	P2											

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 2 (admission)

Files d'attente

haute:	P3	P4										
moyenne:	~~P2~~	~~P2~~	P2									
basse:	P1											

P3 et P4 sont admis

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P3:	2											
P4:	3											

Ordonnancement:	P2	P2										

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 2 (ordonnancement)

Files d'attente

haute:	~~P3~~	P4	P3									
moyenne:	~~P2~~	~~P2~~	P2									
basse:	P1											

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P3:	~~2~~	1										
P4:	3											

Ordonnancement:	P2	P2	P3									

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 3 (admission)

Files d'attente

haute:	~~P3~~	P4	P3									
moyenne:	~~P2~~	~~P2~~	P2									
basse:	P1											

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P3:	~~2~~	1										
P4:	3											

Ordonnancement:	P2	P2	P3									

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 3 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	P3	P4								
moyenne:	~~P2~~	~~P2~~	P2									
basse:	P1											

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P3:	~~2~~	1										
P4:	~~3~~	2										

Ordonnancement:	P2	P2	P3	P4								

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 4 (admission)

Files d'attente

haute:	~~P3~~	~~P4~~	P3	P4								
moyenne:	~~P2~~	~~P2~~	P2									
basse:	P1											

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P3:	~~2~~	1										
P4:	~~3~~	2										

Ordonnancement:	P2	P2	P3	P4								

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 4 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	P4									
moyenne:	~~P2~~	~~P2~~	P2										
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	~~3~~	2										
~~P3:~~	~~2~~	~~1~~	0										
P4:	~~3~~	2											

P3 se termine

Ordonnancement:	P2	P2	P3	P4	P3								

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 5 (admission)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	P4								
moyenne:	~~P2~~	~~P2~~	P2	P5								
basse:	P1											

P5 est admis

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P4:	~~3~~	2										
P5:	2											

Ordonnancement:	P2	P2	P3	P4	P3							

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 5 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	P4							
moyenne:	~~P2~~	~~P2~~	P2	P5								
basse:	P1											

Processus en mémoire

P1:	2											
P2:	~~4~~	~~3~~	2									
P4:	~~3~~	~~2~~	1									
P5:	2											

Ordonnancement:	P2	P2	P3	P4	P3	P4						

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 6 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	P2	P5									
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	~~3~~	2										
~~P4:~~	~~3~~	~~2~~	~~1~~	0									
P5:	2												

P4 se termine

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4						

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 7 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	P5	P2								
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	~~3~~	~~2~~	1									
P5:	2												

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2					

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 8 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	~~P5~~	P2	P5							
basse:	P1												

Processus en mémoire

P1:	2												
P2:	~~4~~	~~3~~	~~2~~	1									
P5:	~~2~~	1											

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2	P5				

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 9 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	~~P5~~	~~P2~~	P5							
basse:	P1												

Processus en mémoire

P1:	2												
~~P2:~~	~~4~~	~~3~~	~~2~~	~~1~~	0								
P5:	~~2~~	1											

P2 se termine

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2	P5	P2			

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 10 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	~~P5~~	~~P2~~	~~P5~~							
basse:	P1												

Processus en mémoire

P1:	2												
~~P5:~~	~~2~~	~~1~~	0										

P5 se termine

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2	P5	P2	P5		

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 1 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	~~P5~~	~~P2~~	~~P5~~							
basse:	~~P1~~	P1											

Processus en mémoire

P1:	~~2~~	1											

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2	P5	P2	P5	P1	

Exercice #1

Nom	Durée	Arrivée	Priorité
P1	2	0	basse
P2	4	0	moyenne
P3	2	2	haute
P4	3	2	haute
P5	2	5	moyenne

Quantum: 12 (ordonnancement)

Files d'attente

haute:	~~P3~~	~~P4~~	~~P3~~	~~P4~~	~~P4~~								
moyenne:	~~P2~~	~~P2~~	~~P2~~	~~P5~~	~~P2~~	~~P5~~							
basse:	~~P1~~	~~P1~~											

Processus en mémoire

~~P1:~~	~~2~~	~~1~~	0									

P1 se termine

Ordonnancement:	P2	P2	P3	P4	P3	P4	P4	P2	P5	P2	P5	P1	P1

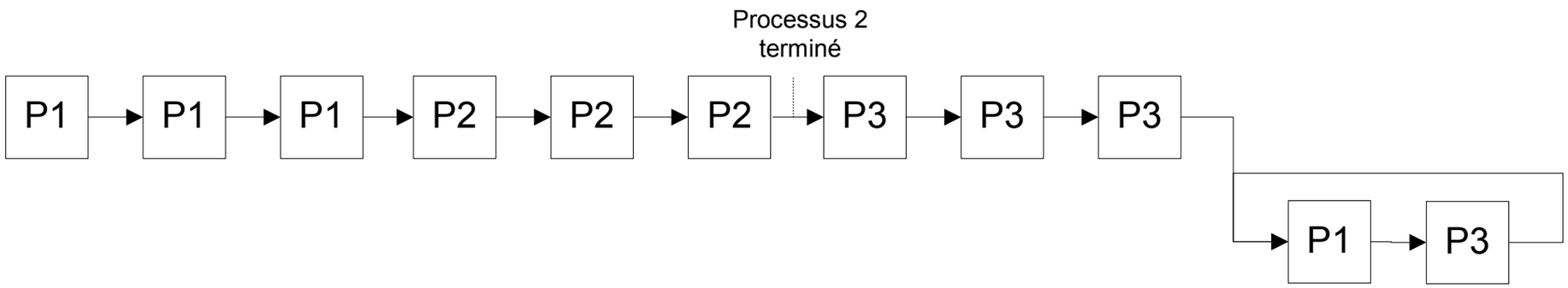
Algorithmes d'ordonnancement

- **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

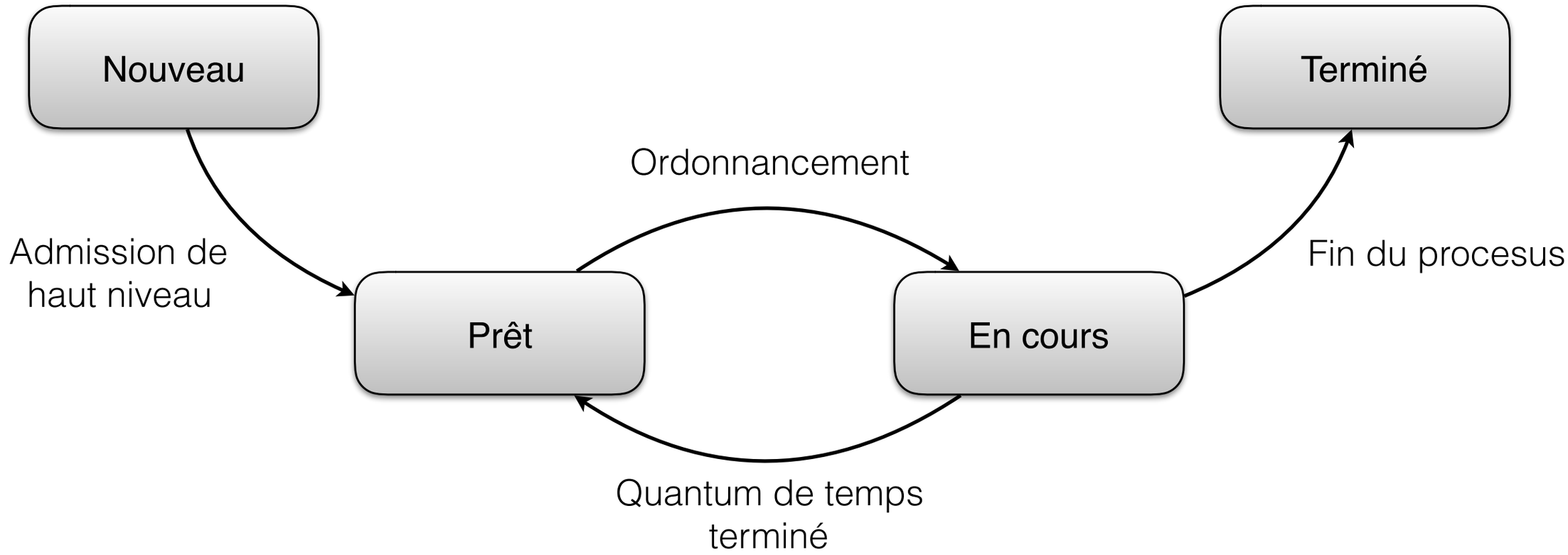


- **File avec tourniquet (round-robin)**: On exécute plusieurs fois les processus nouvellement admis, puis les processus non terminés sont mis dans un tourniquet.
  - Avantages: Tous les processus ont du temps de CPU; très équitable.
  - Désavantages: Même que tourniquet, mais avec un impact moindre

Algorithmes d’ordonnancement

- **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 et entrées-sorties (*I/Os*)

- Lorsqu'il est **en cours**, un processus peut parfois avoir besoin d'accéder à un périphérique. Exemples:
  - demander à accéder un fichier stocké sur le disque dur;
  - imprimer une page grâce à l'imprimante;
  - envoyer des données sur le web (téléverser une photo sur Instagram) via l'interface Ethernet sans-fil.
- Les périphériques sont **beaucoup** plus lents que le micro-processeur, alors le processus doit attendre (longtemps) que la requête soit terminée avant de poursuivre l'exécution.
- Pendant ce temps, le micro-processeur ne fait rien!

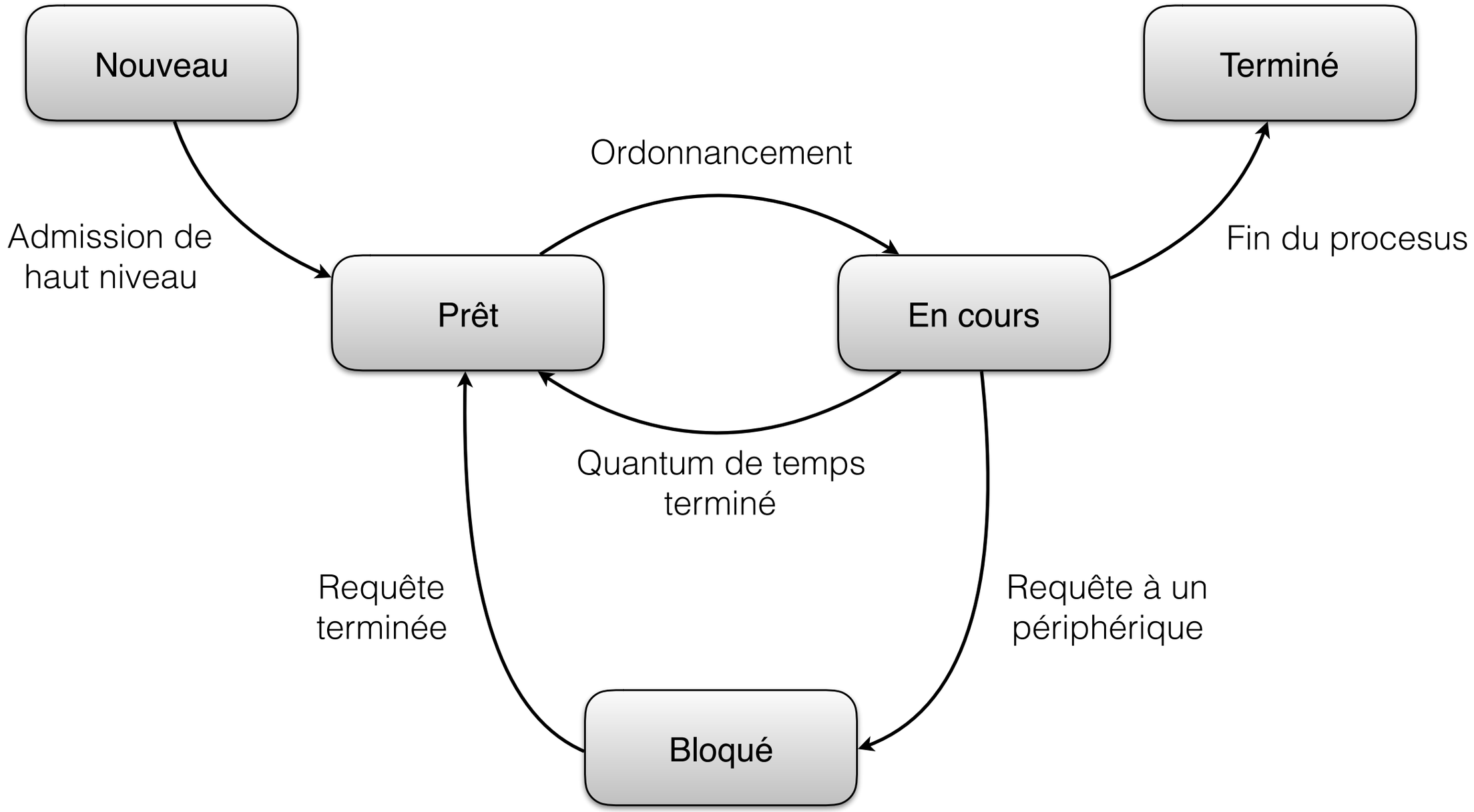
Processus: état « en cours »

- 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**

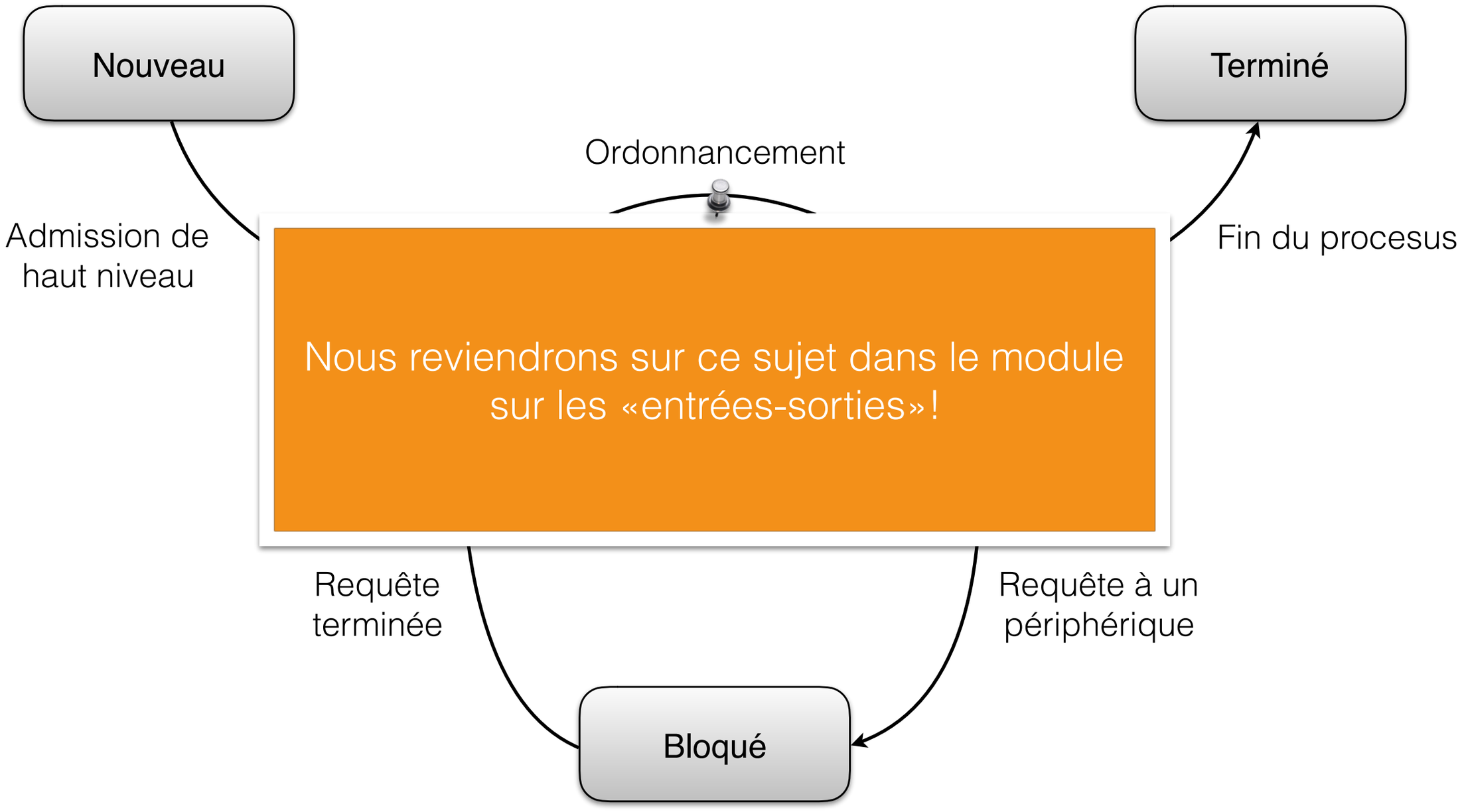
Processus: état « bloqué »

- Le processus est en attente d'un périphérique (entrée-sortie)
- Lorsque le périphérique a terminé la requête,
  - il soulève une interruption pour indiquer au système d'exploitation qu'il a terminé la requête.
  - le système d'exploitation remet alors le processus 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 ordonnées (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.