

# Systemes d'exploitation

## ENSISA 1A

**Maxime Devanne**

[maxime.devanne@uha.fr](mailto:maxime.devanne@uha.fr)

Bureau 3.37

1A IR

1<sup>er</sup> semestre

# Systemes d'exploitation

ENSISA 1A

## Chapitre 2

### Processus

## Objectifs pédagogiques du chapitre

- Connaître les **caractéristiques** des **processus**
- Appréhender leur **programmation** en langage C
- Comprendre les différentes méthode **d'ordonnancement** des processus
- Connaître les **caractéristiques** des **threads** et appréhender ce qui les diffère des processus
- Comprendre la **synchronisation** entre processus

## Objectifs pédagogiques du chapitre

- Connaître les **caractéristiques** des **processus**
- Appréhender leur **programmation** en langage C
- Comprendre les différents méthode **d'ordonnancement** des processus
- Connaître les **caractéristiques** des **threads** et appréhender ce qui les diffère des processus
- Comprendre la **synchronisation** entre processus



Introduction

Les processus

Notions de programmation des  
processus

**L'ordonnancement**

Les threads

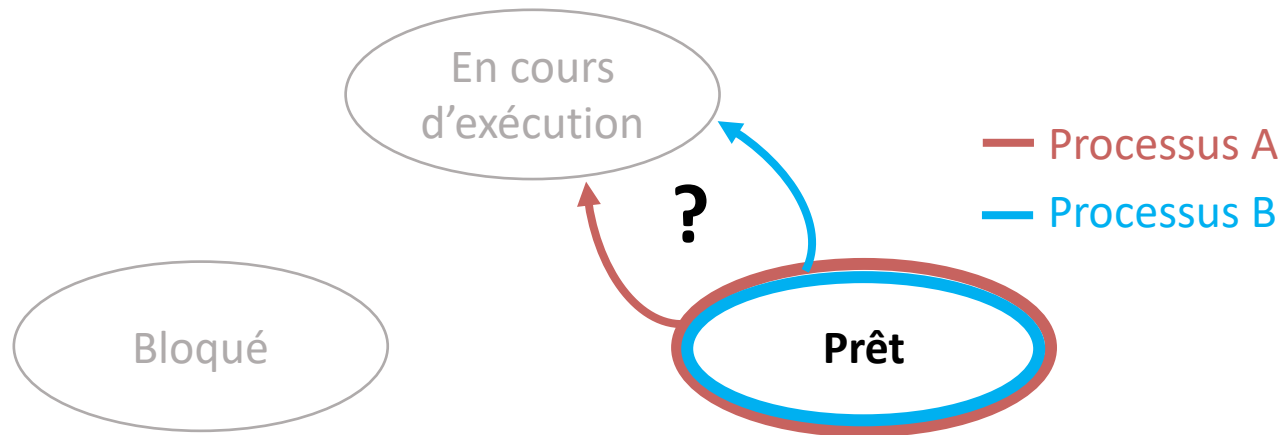
Synchronisation

Conclusion

# L'ordonnancement

## Introduction

- Un ordinateur multiprogrammé possède plusieurs processus en **concurrence** pour l'obtention du **temps processeur**
  - Deux ou plusieurs processus sont en état **prêt** en même temps
  - Un processeur ne peut exécuter **qu'un seul** processus à la fois
    - Un choix doit être fait : quel processus exécuter ?



- C'est le rôle de **l'ordonnanceur** (*scheduler*)
- Il utilise des **algorithmes d'ordonnancement**

## Problématiques

- Comment sélectionner le bon processus à un instant  $t$ ?
  - Chaque processus occupe du temps processeur
  - Comment limiter le temps d'attente perçu par l'utilisateur ?
- La problématique est moins vraie aujourd'hui pour la plupart des systèmes
  - Il n'y a souvent qu'un seul processus « actif »
  - Le temps processeur n'est plus une ressource rare
    - La plupart des programmes sont limités par la vitesse à laquelle l'utilisateur interagit (clavier, souris) et non par celle du processeur
- Cas des serveurs en réseau
  - Plusieurs processus (clients) se disputent du temps processeur

## Problématiques

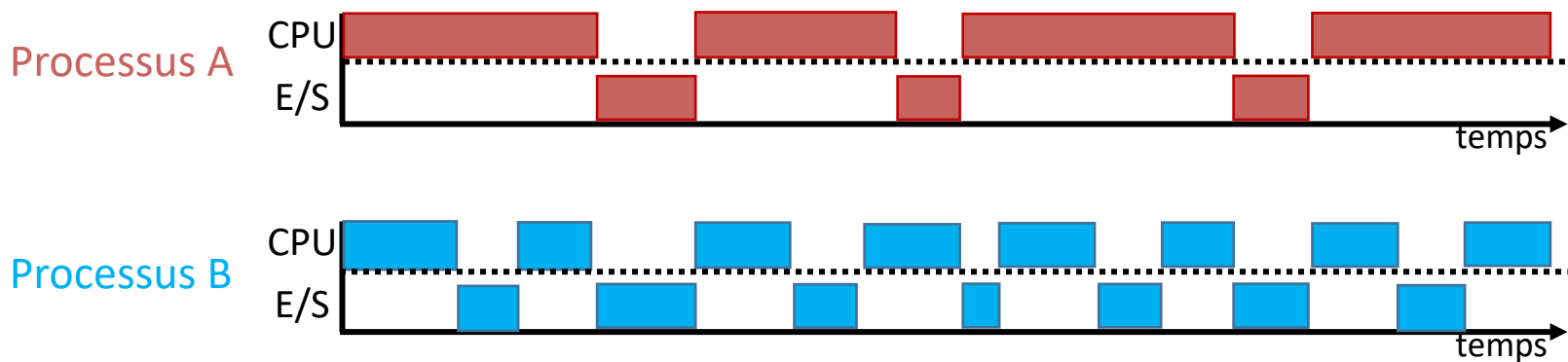
- Comment faire un usage efficace de l'UC ?
  - Le passage d'un processus à un autre est coûteux
    - Basculement du mode utilisateur vers le mode noyau
    - L'état du processus en cours est enregistré (stockage du registre)
    - Chargement mémoire du nouveau processus
    - Démarrage du nouveau processus
- Si les changements de processus sont trop nombreux, ils peuvent consommer un temps processeur notable



# L'ordonnancement

## Le comportement des processus

- Les processus alternent :
  - Les phase de traitement CPU
  - Les phases d'attente d'E/S

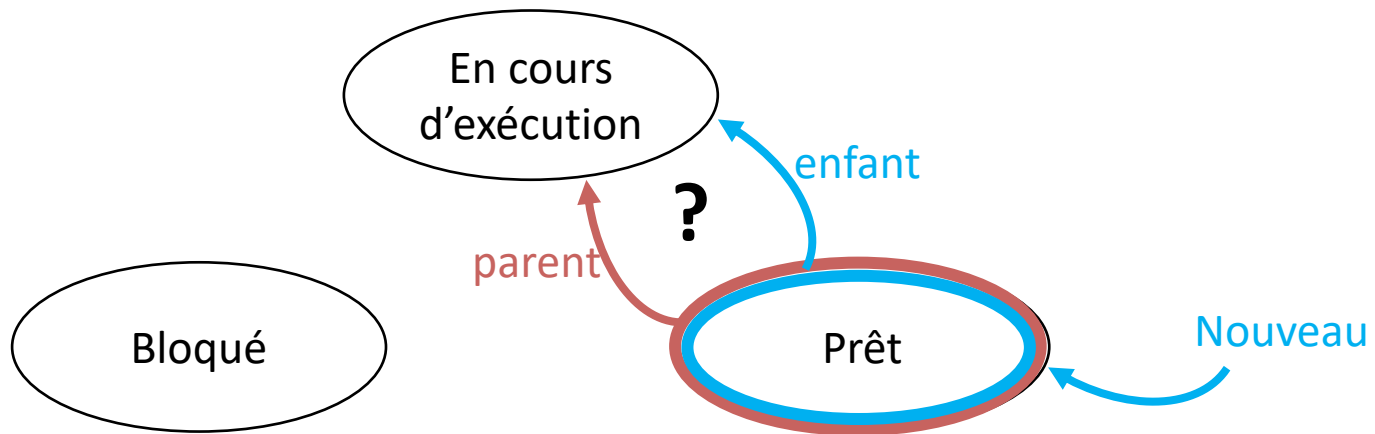


- On distingue deux types de comportements :
  - **Processus de traitement :**
    - Longues phases de traitement et rares attentes d'E/S
  - **Processus d'E/S**
    - Brèves phases de traitement et fréquentes attentes d'E/S

# L'ordonnancement

## Quand ordonnancer ?

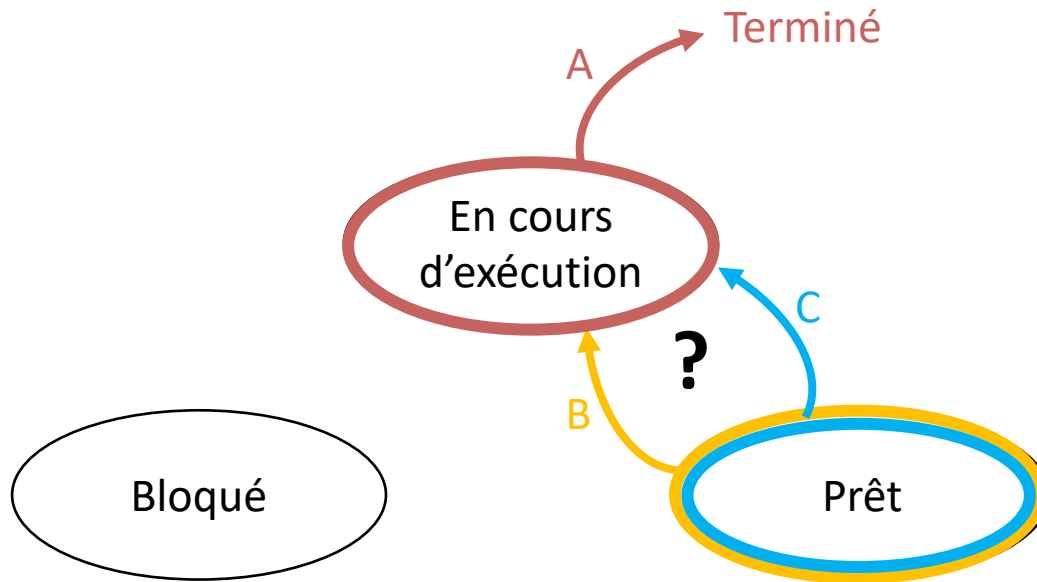
- Plusieurs situations nécessitent de l'ordonnancement :
  - 1. Lorsqu'un nouveau processus **enfant** est créé
    - Les deux processus sont en état prêt
    - Il faut décider quel processus exécuter (**parent** ou **enfant**)



# L'ordonnancement

## Quand ordonnancer ?

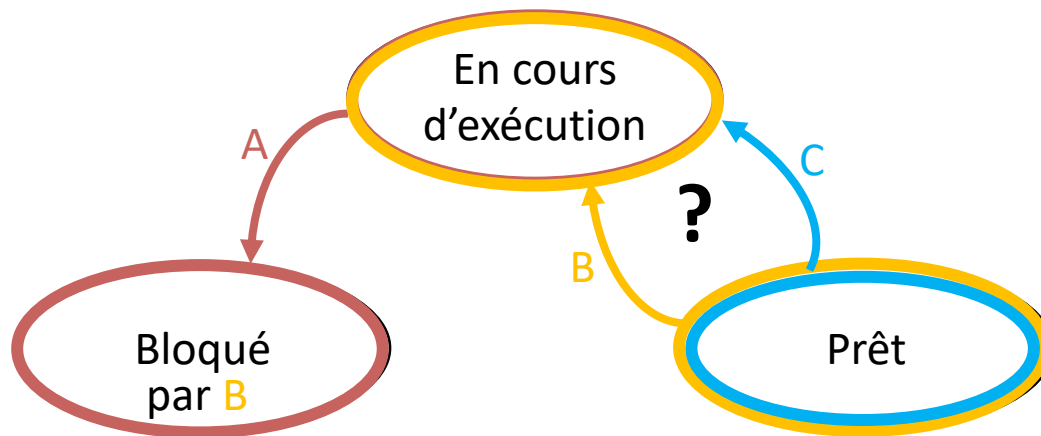
- Plusieurs situations nécessitent de l'ordonnancement :
  - 2. Lorsqu'un processus **A** se termine
    - Il faut décider quel processus exécuter à la suite parmi le jeu de processus prêts (**B** ou **C**)



# L'ordonnancement

## Quand ordonnancer ?

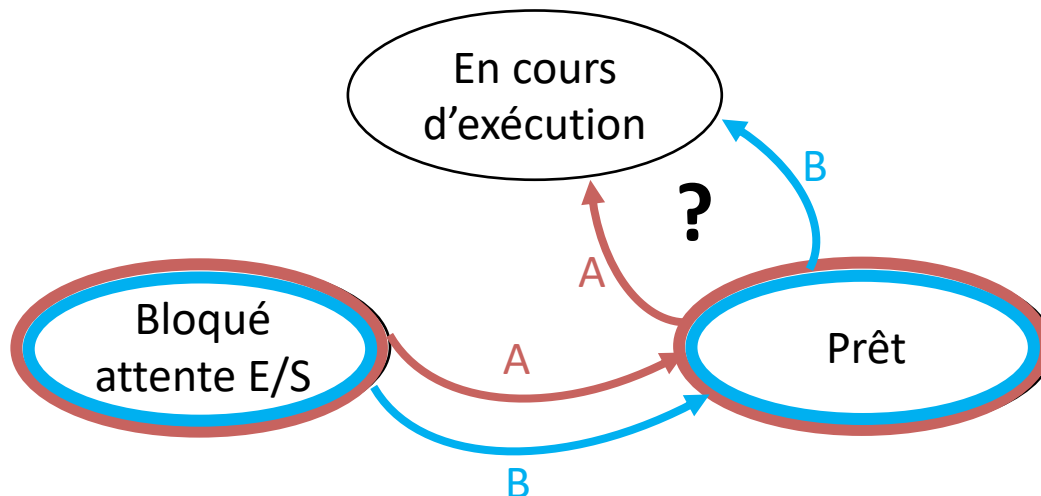
- Plusieurs situations nécessitent de l'ordonnancement :
  - 3. Lorsqu'un processus **A** bloque sur des E/S ou un autre processus
    - Il faut décider quel processus exécuter pendant cette attente (**B** ou **C**)
    - La raison du blocage peut jouer un rôle (si **A** attend **B**, alors préférer **B**)



# L'ordonnancement

## Quand ordonnancer ?

- Plusieurs situations nécessitent de l'ordonnancement :
  - 4. Lorsqu'une interruption d'E/S se produit
    - Tout processus qui attendait cette interruption est éligible (A et B)
    - Il faut décider lequel choisir (A ou B)



## Algorithmes d'ordonnancement

- Deux principales catégories
  - Ordonnance coopératif (ou non préemptif)
    - L'algorithme sélectionne un processus puis le laisse s'exécuter jusqu'à ce qu'il bloque (attente d'E/S ou un autre processus)
    - Une fois le traitement de l'interruption terminé le même processus est relancé
  - Ordonnancement préemptif
    - L'algorithme sélectionne un processus puis le laisse s'exécuter pendant un certain délai déterminé
    - Si le processus est toujours en cours à la fin de ce délai, il est suspendu
    - Cela nécessite une interruption à la fin du délai (horloge matérielle qui fournit des interruptions périodiques)
- En l'absence d'horloge l'ordonnancement non préemptif est la seule solution

## Algorithmes d'ordonnancement

- Différents types d'environnements ou d'usages
  - Traitement par lots
    - Enchaînement automatique de commandes sans opérateur humain
    - Ex: Etats de stocks, traitement de dossiers d'assurances
    - Les algorithmes non préemptifs ou préemptifs avec de long délais sont souvent utilisés
  - Interactifs
    - Utilisateurs interactifs
    - Les algorithmes préemptifs sont préférés
  - Temps réel
    - Contraintes temps réel
    - La préemption est curieusement parfois inutile
    - Les processus savent qu'ils doivent s'exécuter rapidement

## Algorithmes d'ordonnancement

- Objectifs de l'algorithme d'ordonnancement
  - Important à définir avant de concevoir un algorithme
    - Certains sont souhaitables dans tous les cas
    - D'autres sont dépendants des environnements
  - Les objectifs peuvent servir pour évaluer les performances des algorithmes
- Tous les environnements
  - **Équité** : attribuer à chaque processus un temps processeur équitable
  - **Application de la politique** : faire en sorte que la politique définie soit bien appliquée
  - **Équilibre** : faire en sorte que toutes les parties du système soient occupées



## Algorithmes d'ordonnancement

- Objectifs de l'algorithme d'ordonnancement
  - Traitement par lots
    - **Capacité de traitement** : optimiser le nombre de jobs à l'heure
    - **Délai de rotation** : réduire le délai entre la soumission et l'achèvement
    - **Utilisation de l'UC** : faire en sorte que le processeur soit occupé en permanence
  - Interactifs
    - **Temps de réponse** : répondre rapidement aux requêtes
    - **Proportionnalité** : répondre aux attentes des utilisateurs
  - Temps réel
    - **Respecter les délais** : éviter de perdre les données
    - **Prévisibilité** : éviter la dégradation de la qualité dans les systèmes multimédias

## Algorithmes d'ordonnancement

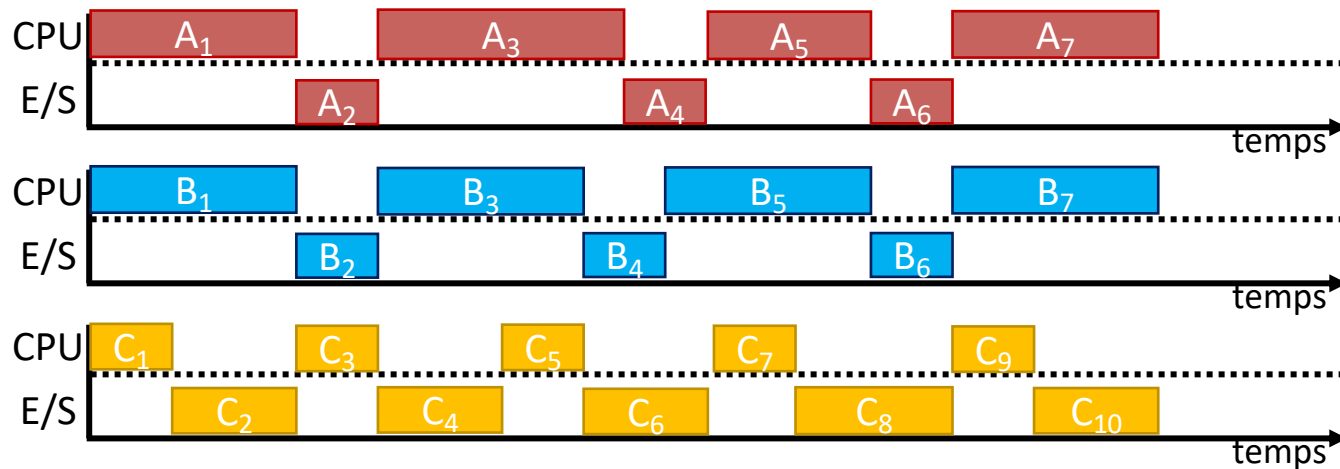
- **1.** Premier arrivé, premier servi
  - Environnements de traitement par lots
- **2.** Exécution du job le plus court en premier
  - Environnements de traitement par lots (et interactifs)
- **3.** Exécution du temps restant suivant le plus court
  - Environnements de traitements par lots
- **4.** Exécution du ratio de réponse suivant le plus élevé
  - Environnements de traitements par lots
- **5.** Ordonnancement de type tourniquet (round robin)
  - Environnements interactifs
- **6.** Ordonnancement par priorité
  - Environnements interactifs
- **7.** Ordonnancement équitable
  - Environnements interactifs
- **8.** Ordonnancement RMS (Rate Monotonic Scheduling)
  - Environnements temps réel

# L'ordonnancement

## Algorithmes d'ordonnancement

- Exemple de situation

- Vision omnisciente



- Vision de l'ordonnanceur à l'instant **t=0**

- La file d'attente (ready queue) contient A<sub>1</sub>, B<sub>1</sub> et C<sub>1</sub>
    - Le CPU est libre

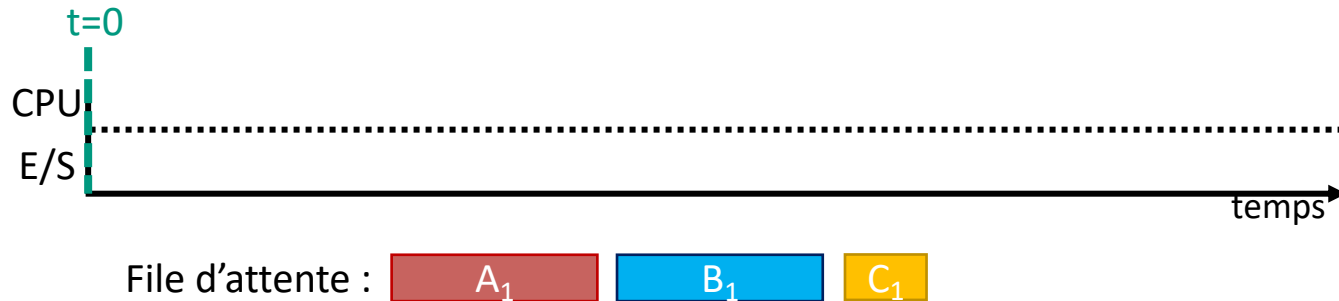
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



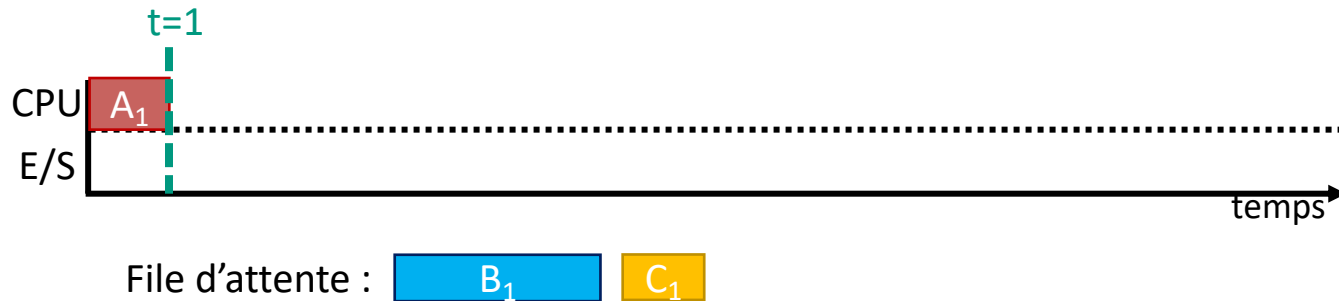
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



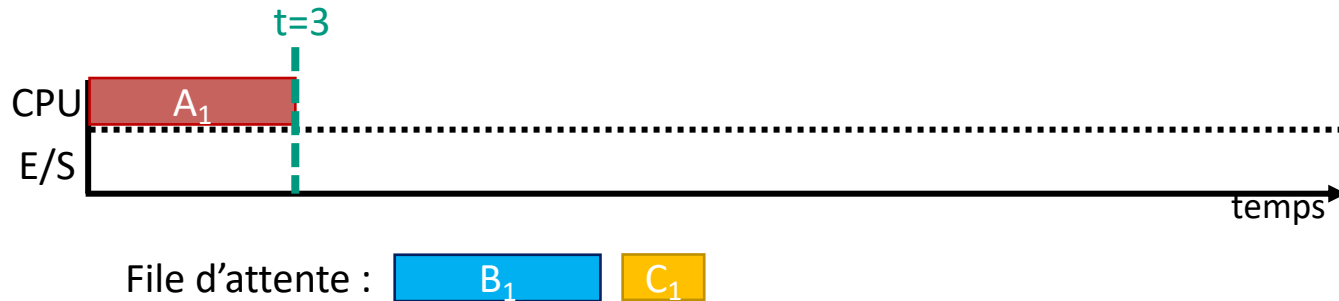
# L'ordonnement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



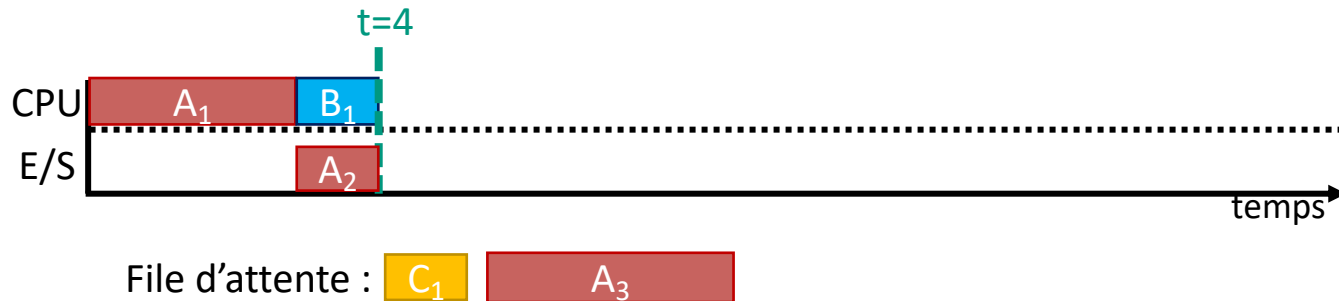
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



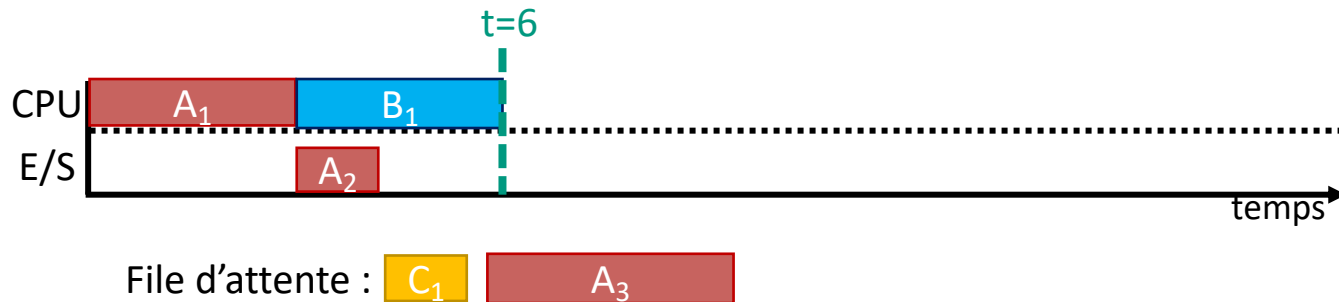
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente





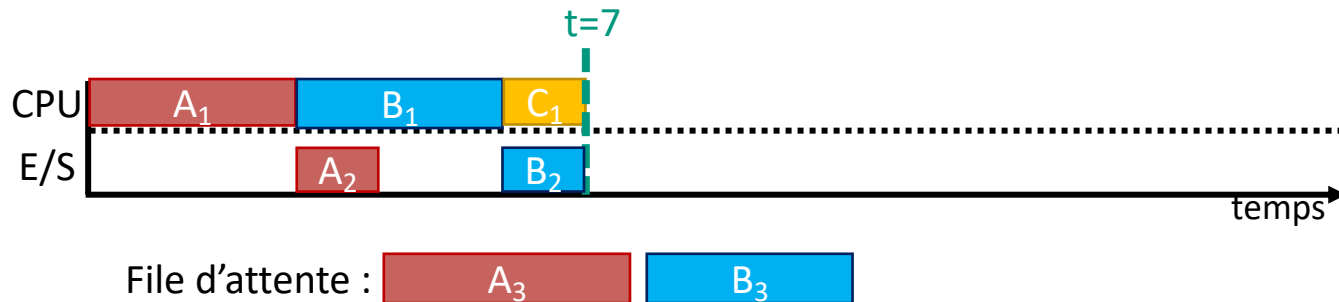
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



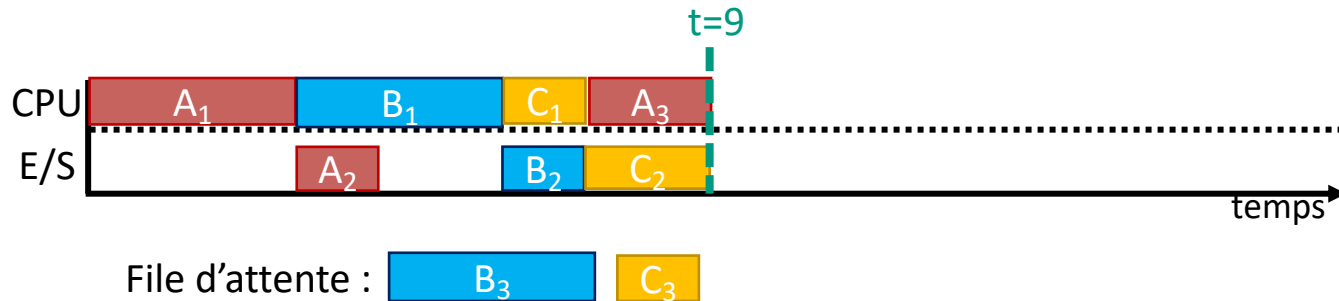
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



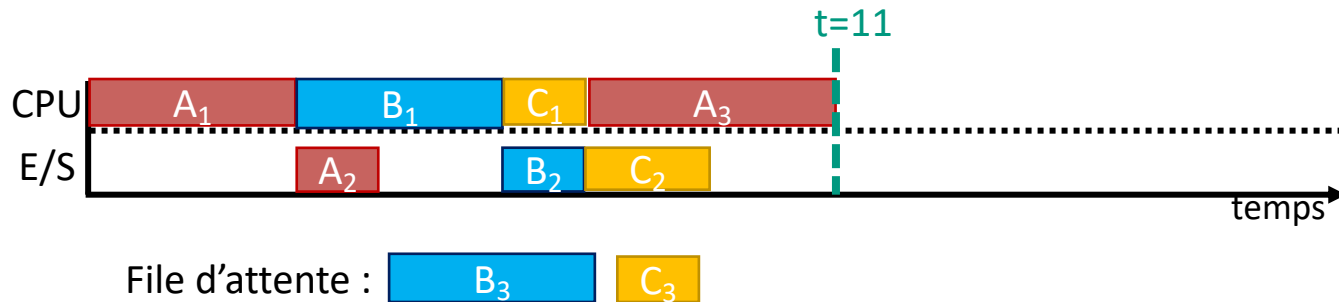
# L'ordonnement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



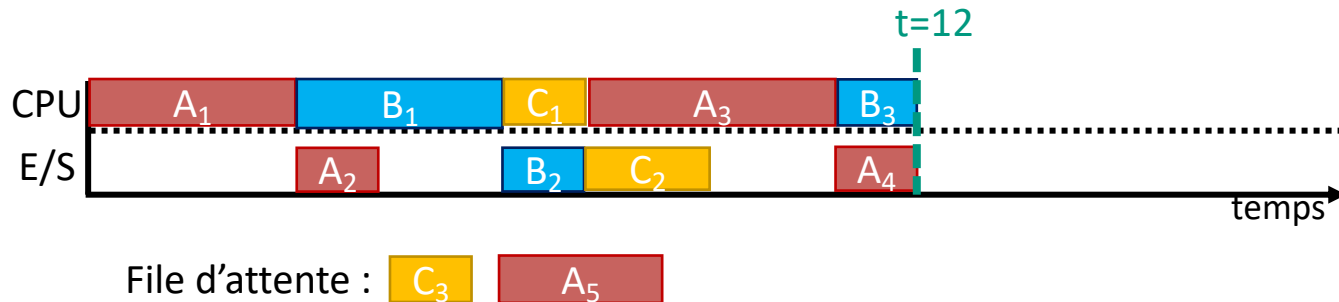
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



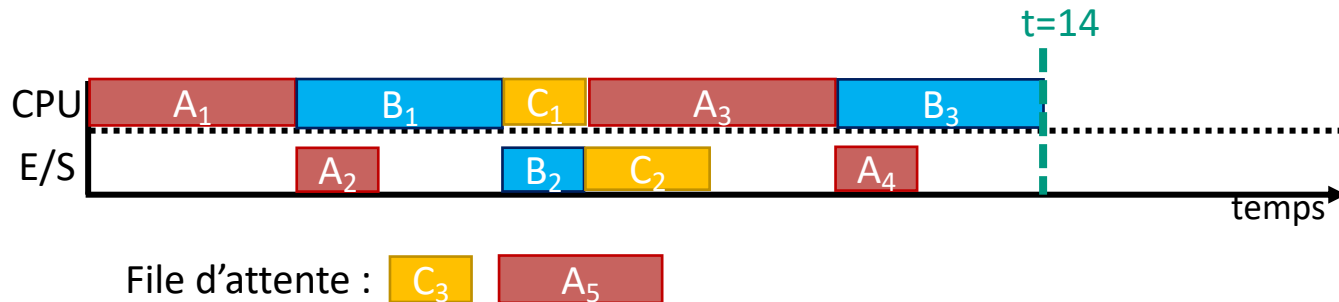
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



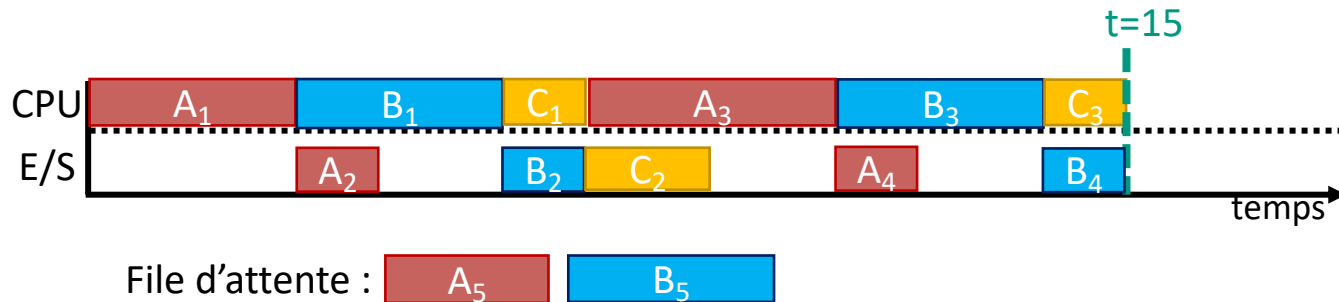
# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



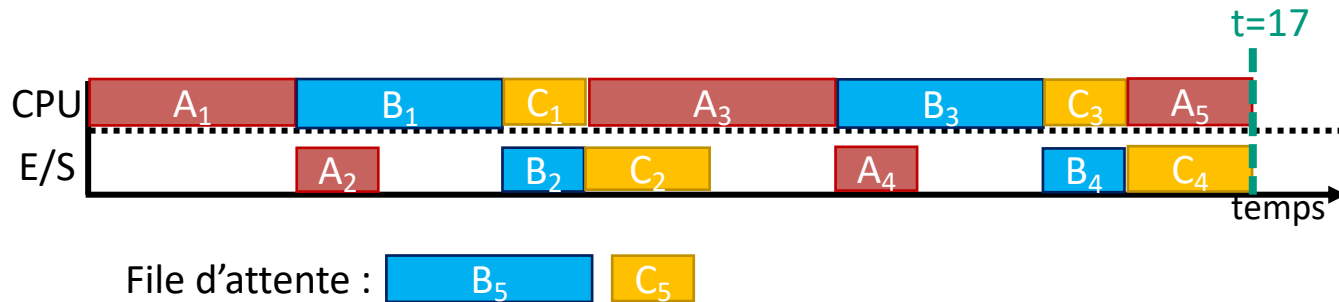
# L'ordonnement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



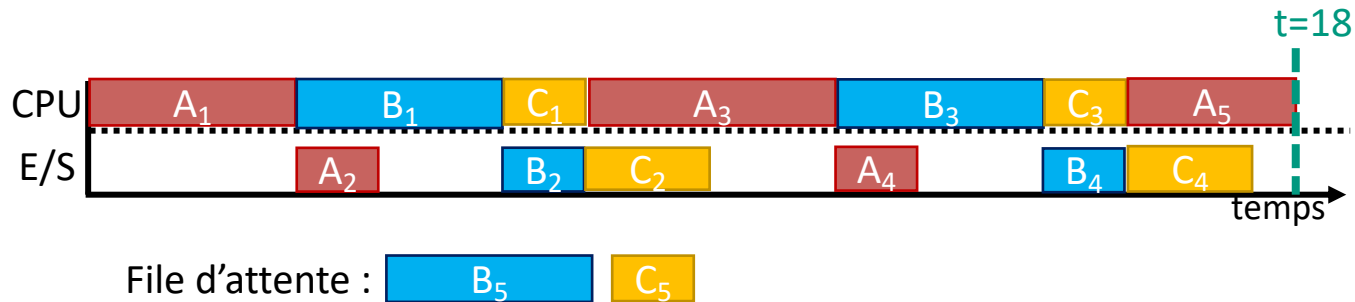
# L'ordonnement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- Politique :

- Lancement des tâches **par ordre d'arrivée** dans la file d'attente



- Non préemptif
- ✓ Simple et relativement équitable
- ✗ Le temps de réponse dépend du processus qui a la main
- ✗ Pénalise les processus courts (ratio attente/exécution)

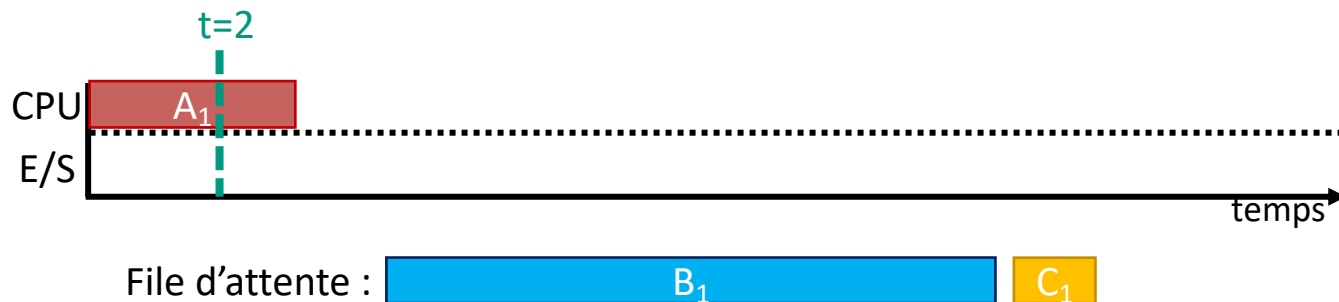


# L'ordonnement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

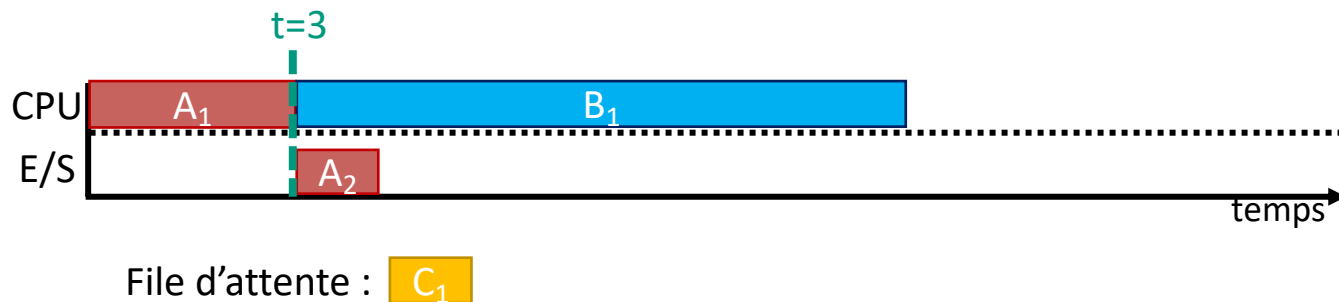
- **×** Le temps de réponse dépend du processus qui a la main
- **×** **Pénalise les processus courts** (ratio attente/exécution)
- Exemple:
  - Le processus **C<sub>1</sub>** arrive dans la file d'attente à **t=2**
  - Le processus **B<sub>1</sub>** est arrivé à **t=1**, il est donc prioritaire



# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)
  - **×** Le temps de réponse dépend du processus qui à la main
  - **×** **Pénalise les processus courts** (ratio attente/exécution)
  - Exemple:
    - Le processus **C<sub>1</sub>** arrive dans la file d'attente à **t=2**
    - Le processus **B<sub>1</sub>** est arrivé à **t=1**, il est donc prioritaire
    - A **t=3**, l'ordonnanceur choisit **B<sub>1</sub>**



# L'ordonnancement

## Premier arrivé, premier servi

- First Come First Served (FCFS)

- **✗** Le temps de réponse dépend du processus qui à la main

- **✗** Pénalise les processus courts (ratio attente/exécution)

- Exemple:

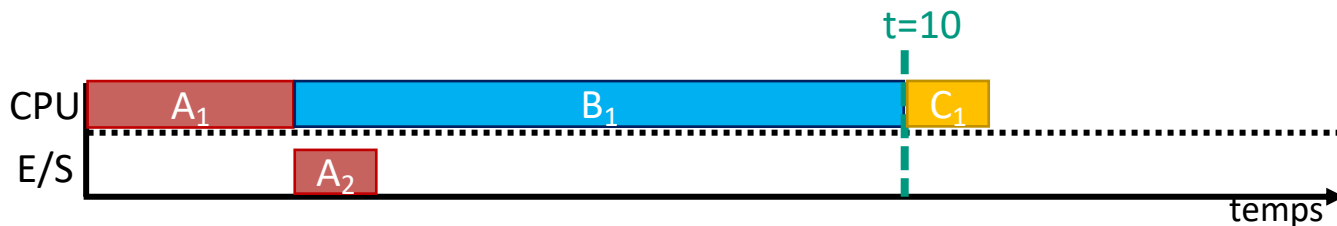
- Le processus  $C_1$  arrive dans la file d'attente à  $t=2$

- Le processus  $B_1$  est arrivé à  $t=1$ , il est donc prioritaire

- A  $t=3$ , l'ordonnanceur choisit  $B_1$

- $C_1$  doit attendre  $t=10$  pour pouvoir être exécuté

- $C_1$  attend **8 unités de temps**, 8 fois plus que son temps d'exécution !



File d'attente :

# L'ordonnancement

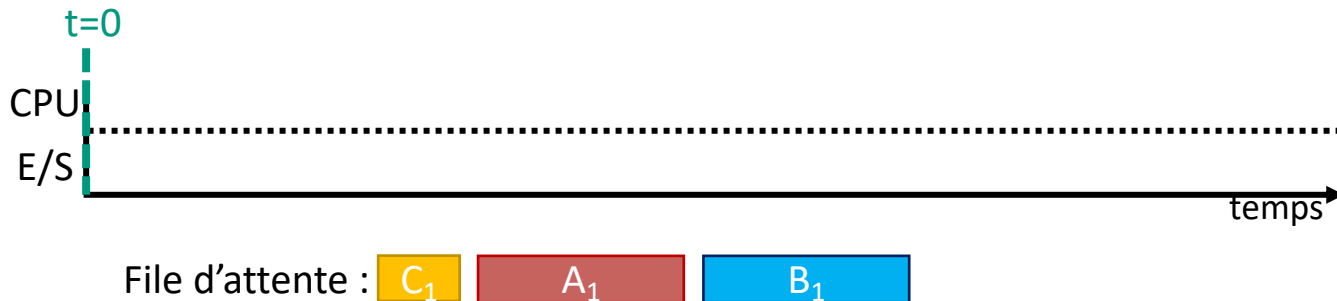
## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
    - Réévalué à chaque transition d'état **bloqué** -> **prêt**

⚠ On suppose la connaissance du temps d'exécution



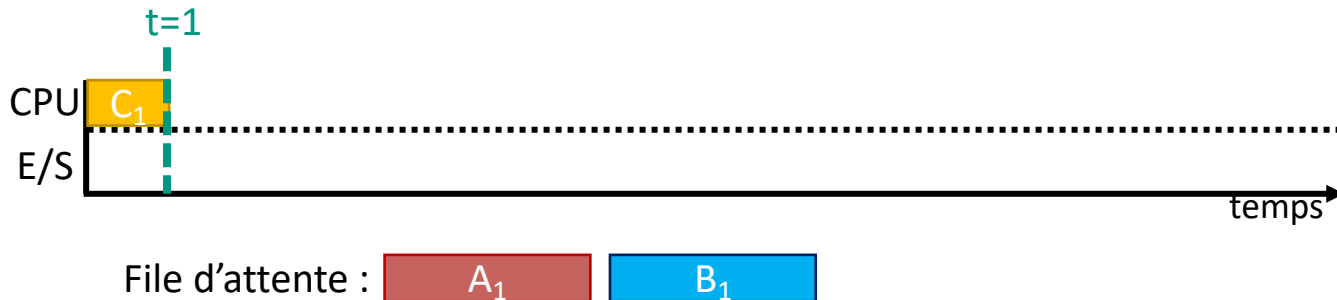
# L'ordonnement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
    - Réévalué à chaque transition d'état **bloqué** -> **prêt**



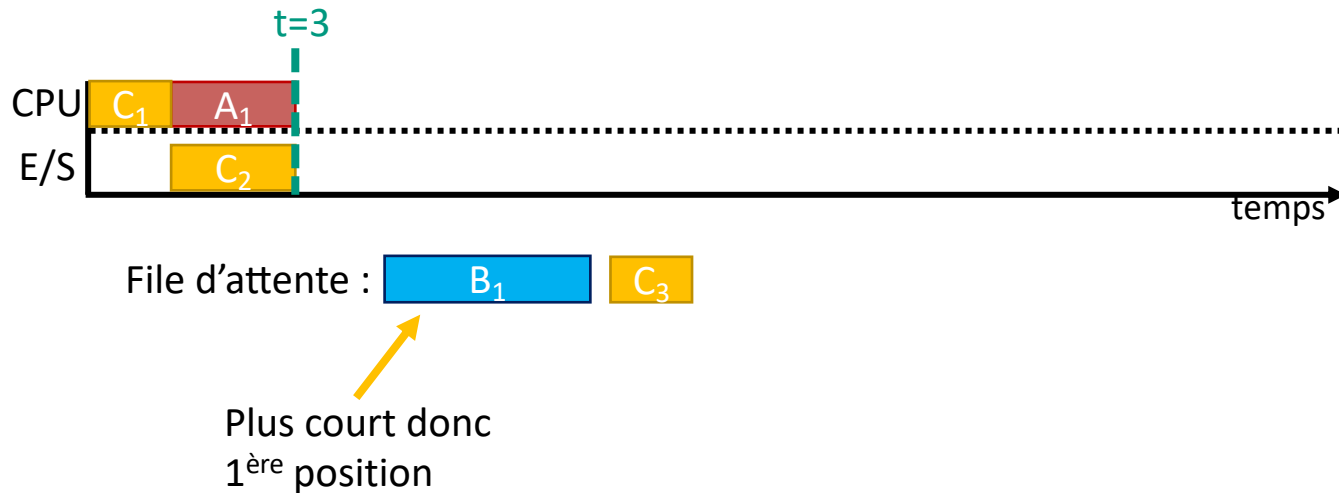
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



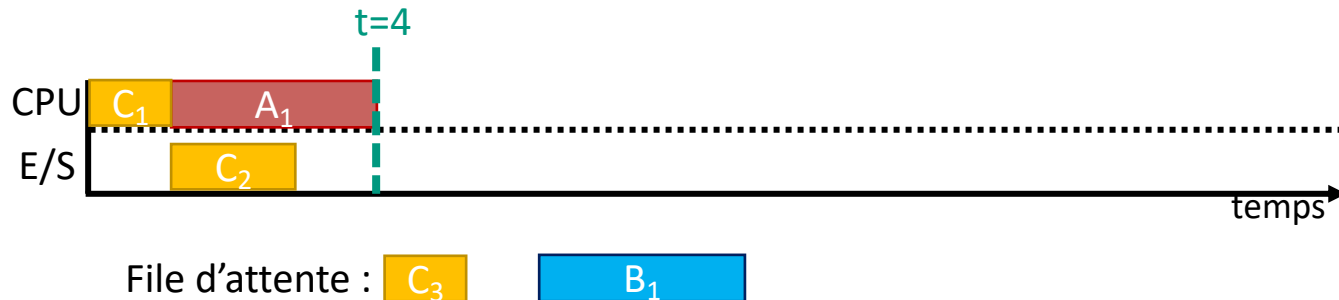
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



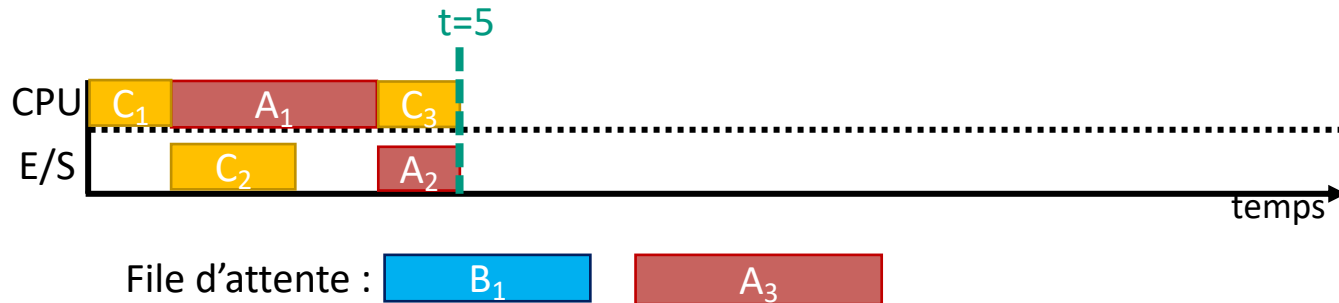
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**





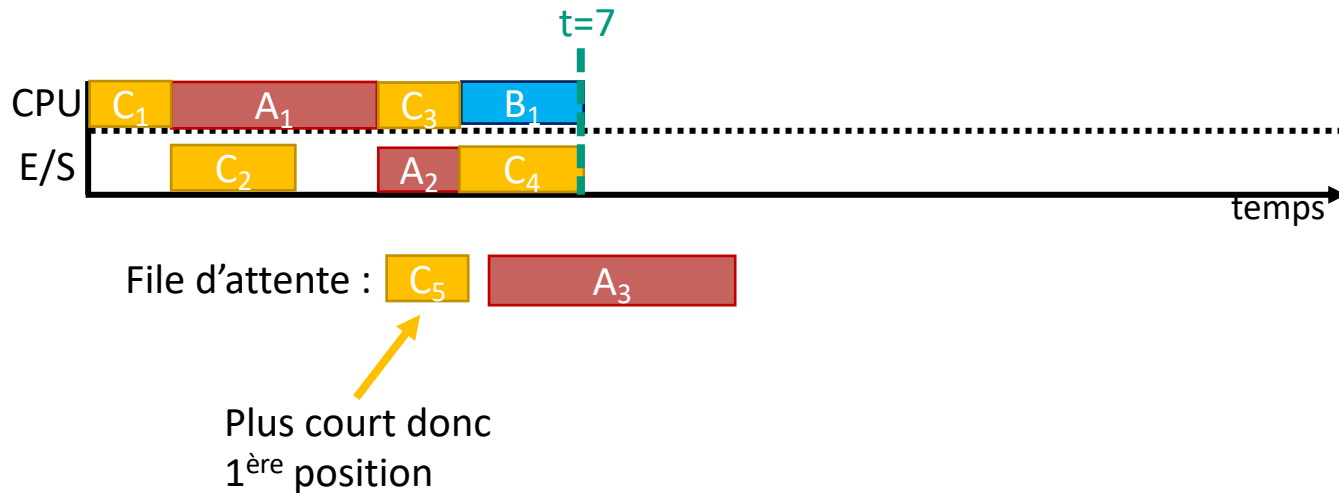
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



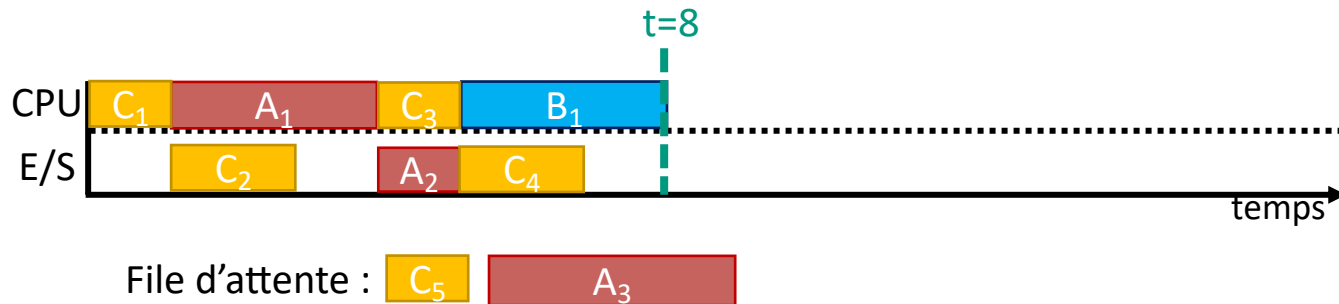
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



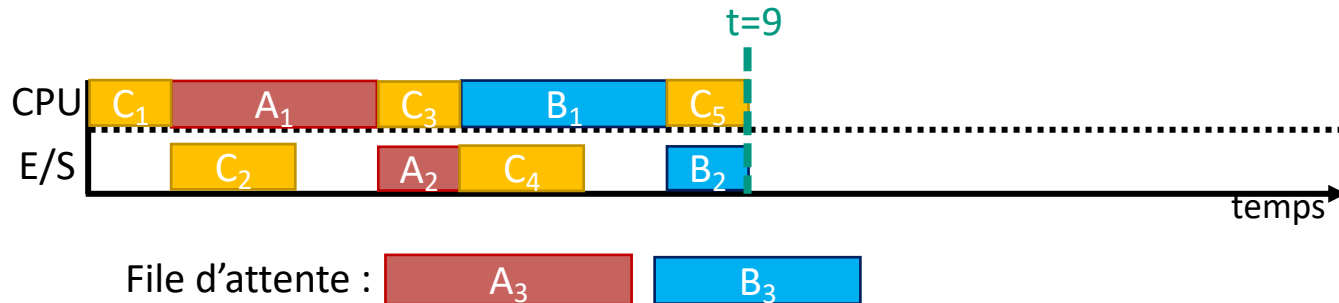
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



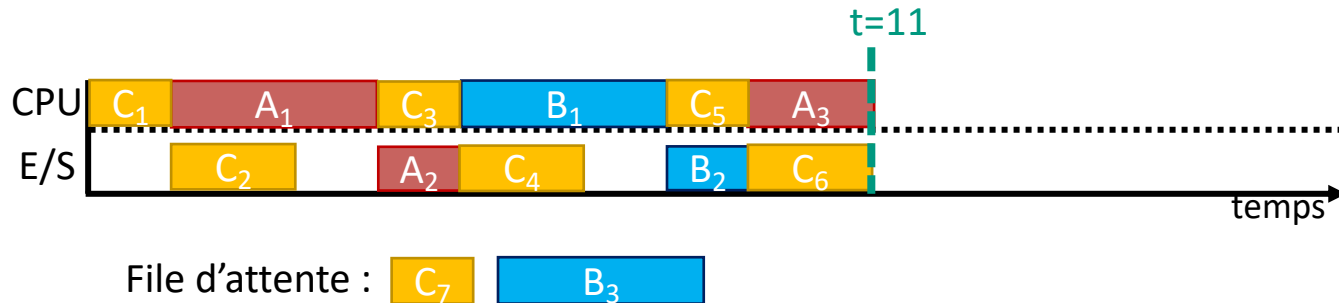
# L'ordonnement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



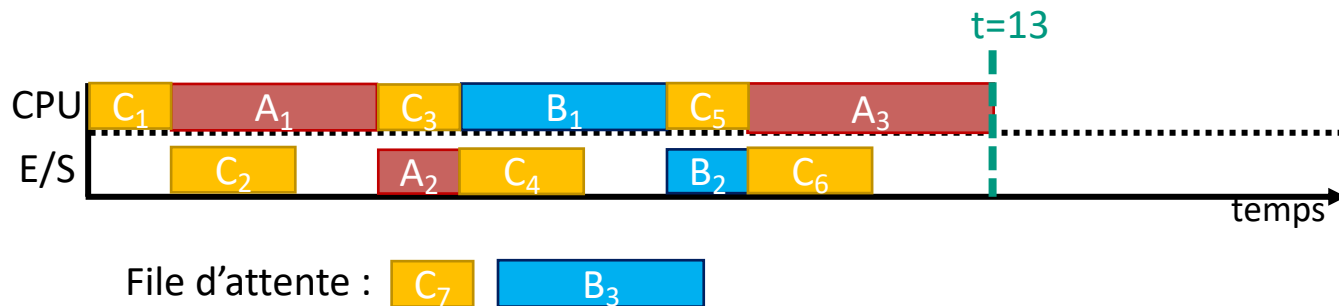
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



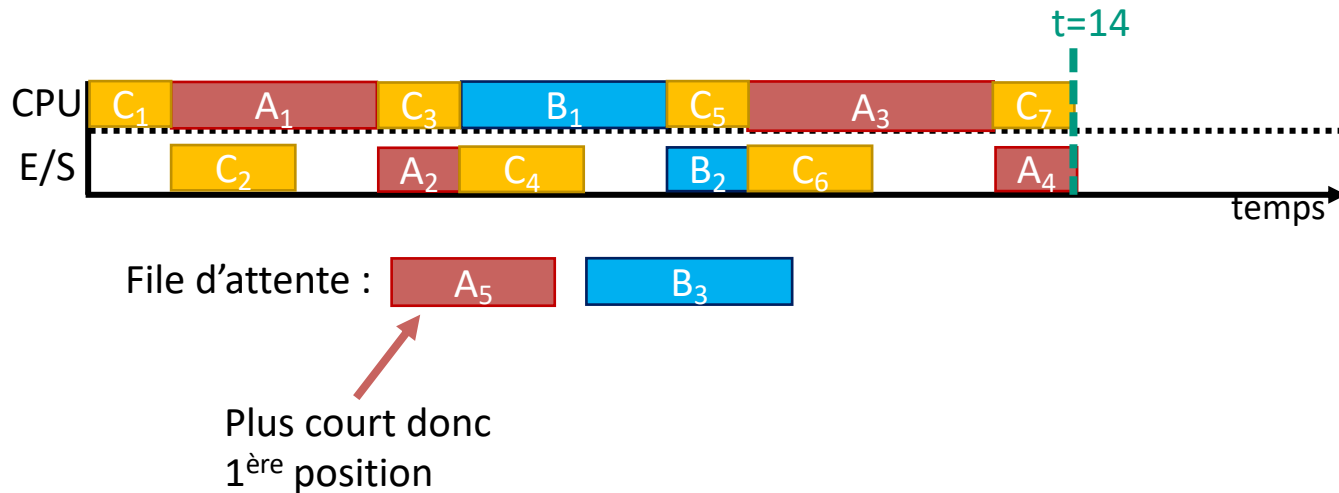
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué -> prêt**



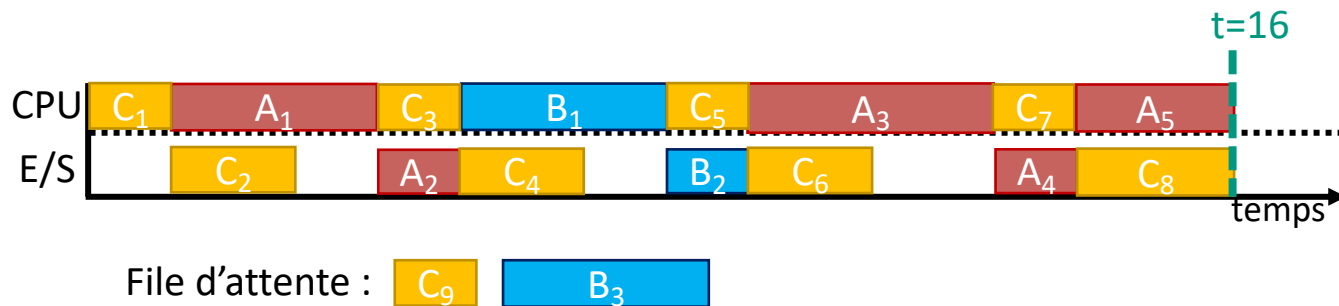
# L'ordonnement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



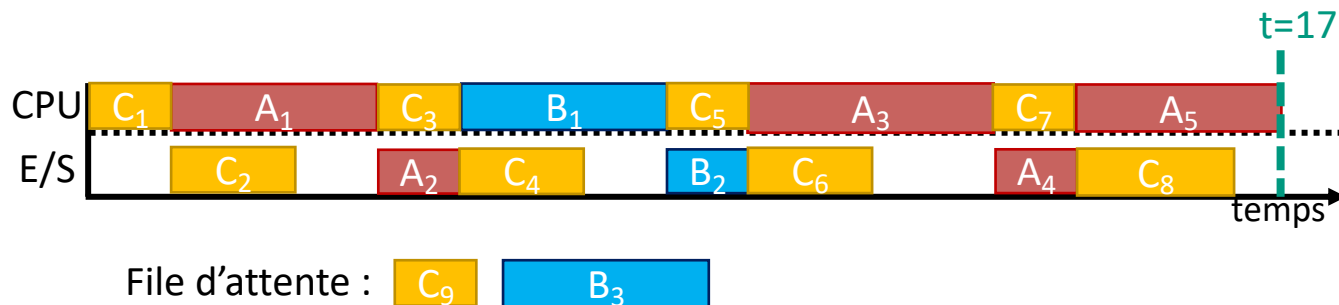
# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- Politique :

- Lancement de la **tâche la plus courte** en premier
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



- Non préemptif
- ✓ Avantageux pour les processus d'E/S sans être trop pénalisant pour les processus de traitement
- ✗ Risque de monopolisation du processeur



# L'ordonnancement

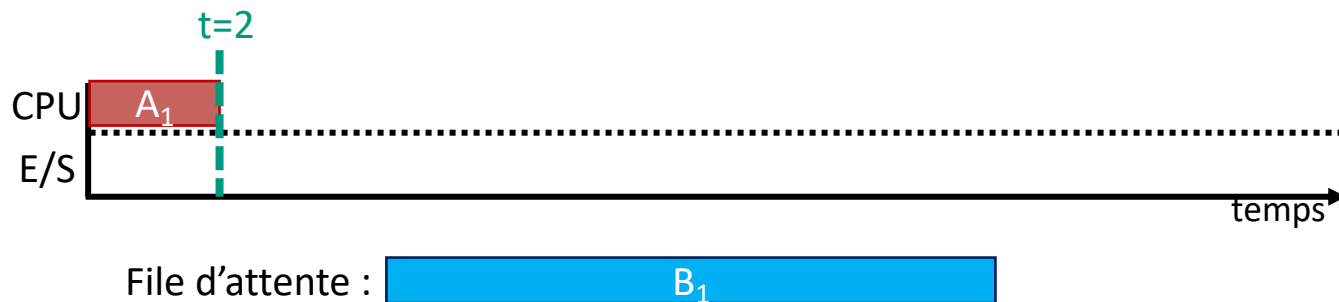
## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- **×** Risque de **monopolisation** du processeur

- Exemple:

- À  $t=2$ , la file d'attente ne contient uniquement  $B_1$  qui est donc exécuté



# L'ordonnancement

## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- **×** Risque de **monopolisation** du processeur

- Exemple:

- À  $t=2$ , la file d'attente ne contient uniquement  $B_1$  qui est donc exécuté



File d'attente :

# L'ordonnancement

## Exécution du job le plus court en premier

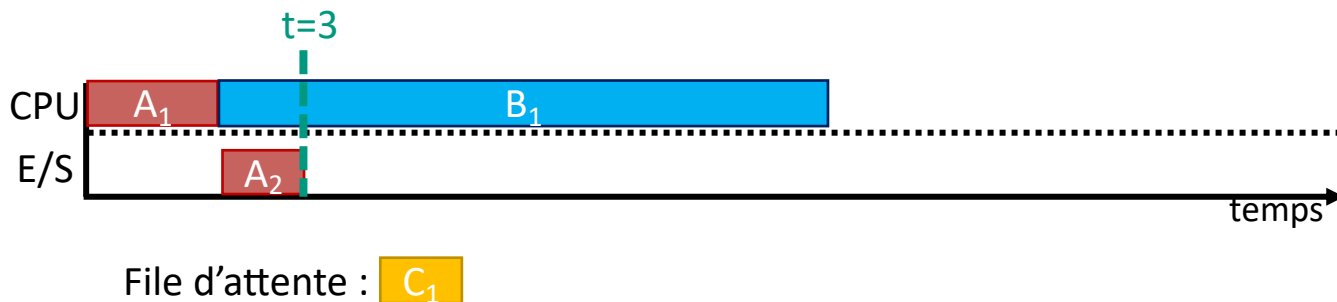
- Shortest Job First (SJF)

- **×** Risque de **monopolisation** du processeur

- Exemple:

- À  $t=2$ , la file d'attente ne contient uniquement  $B_1$  qui est donc exécuté

- À  $t=3$ , le processus  $C_1$  arrive dans la file d'attente



# L'ordonnancement

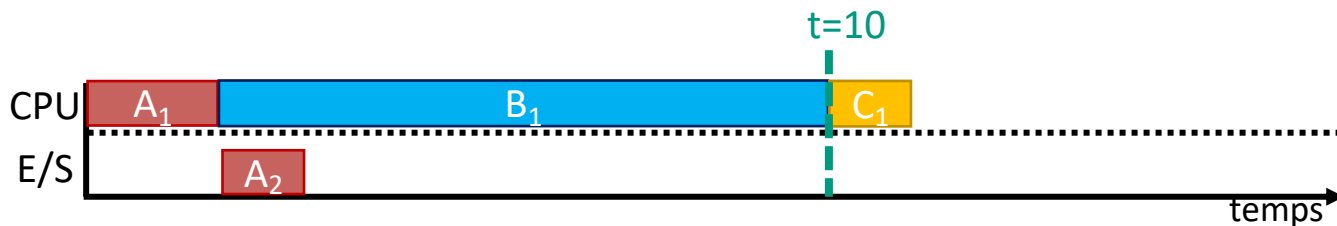
## Exécution du job le plus court en premier

- Shortest Job First (SJF)

- **×** Risque de **monopolisation** du processeur

- Exemple:

- À  $t=2$ , la file d'attente ne contient uniquement  $B_1$  qui est donc exécuté
- À  $t=3$ , le processus  $C_1$  arrive dans la file d'attente
- $C_1$  doit attendre  $t=10$  pour pouvoir être exécuté
- $C_1$  attend **7 unités de temps**, 7 fois plus que son temps d'exécution !



File d'attente :

# L'ordonnancement

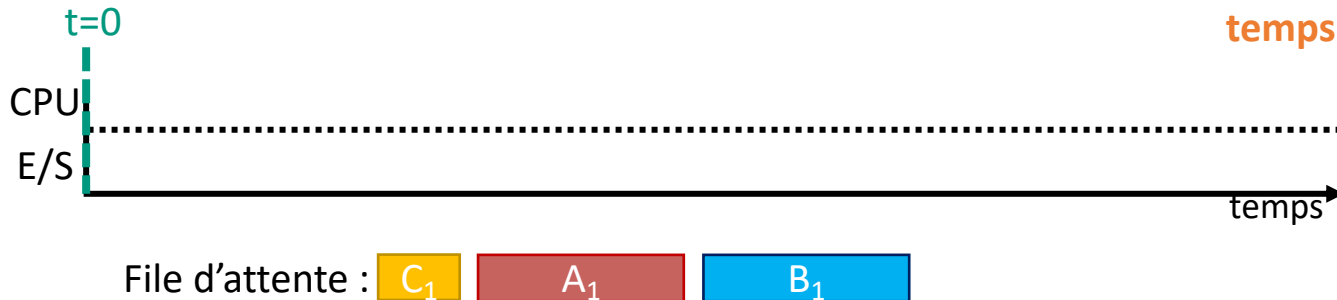
## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**

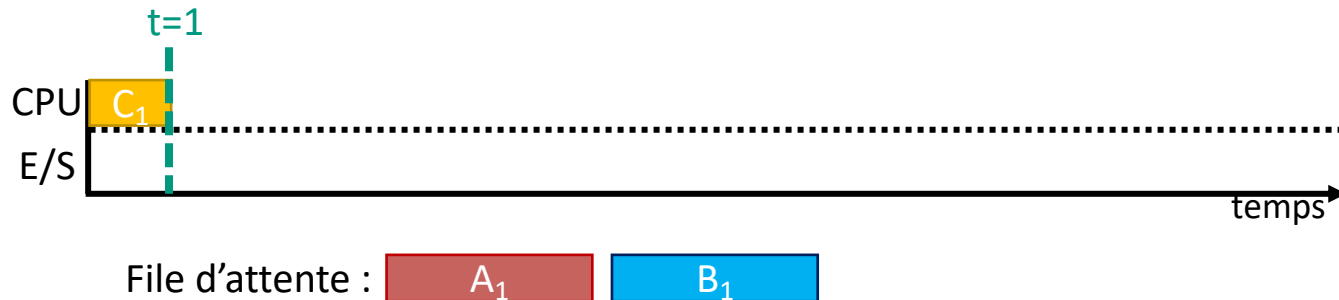
⚠ On suppose la connaissance du temps d'exécution



# L'ordonnement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)
  - Politique :
    - Lancement de la tâche avec le **temps restant le plus court**
    - Réévalué à chaque transition d'état **bloqué** -> **prêt**



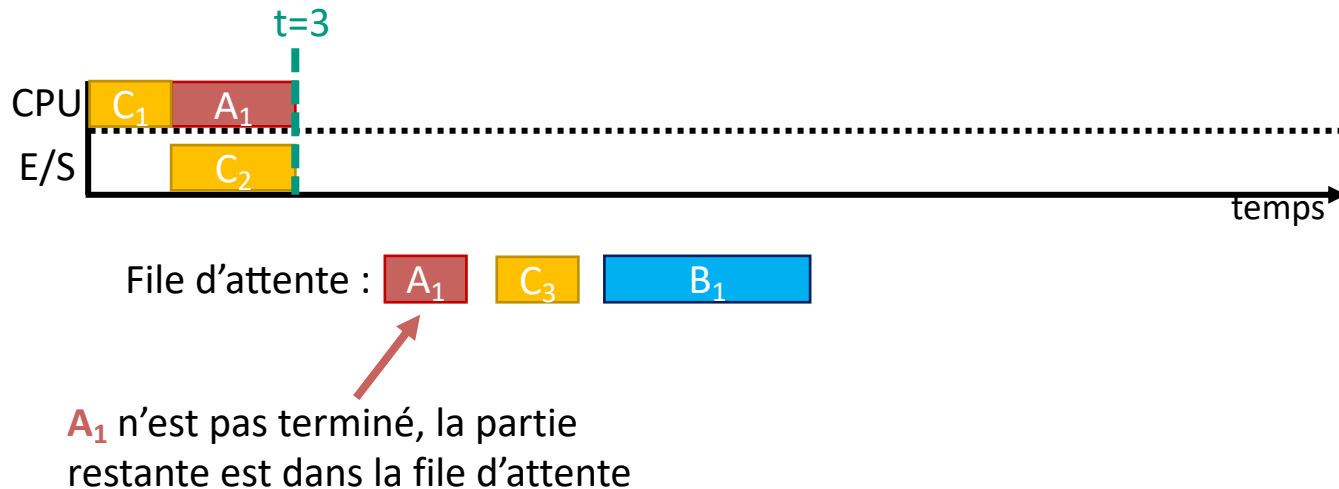
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



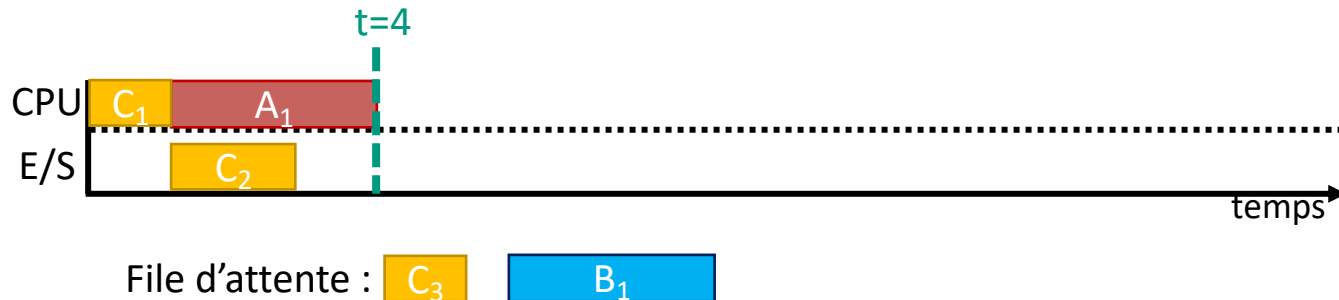
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**





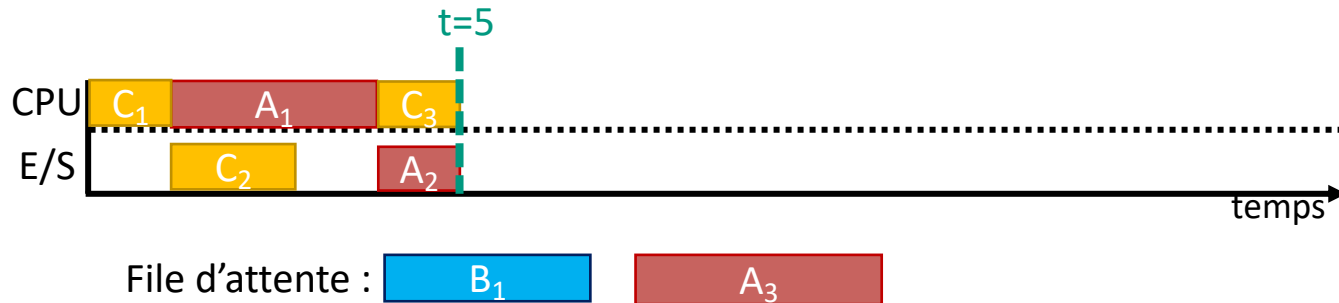
# L'ordonnement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



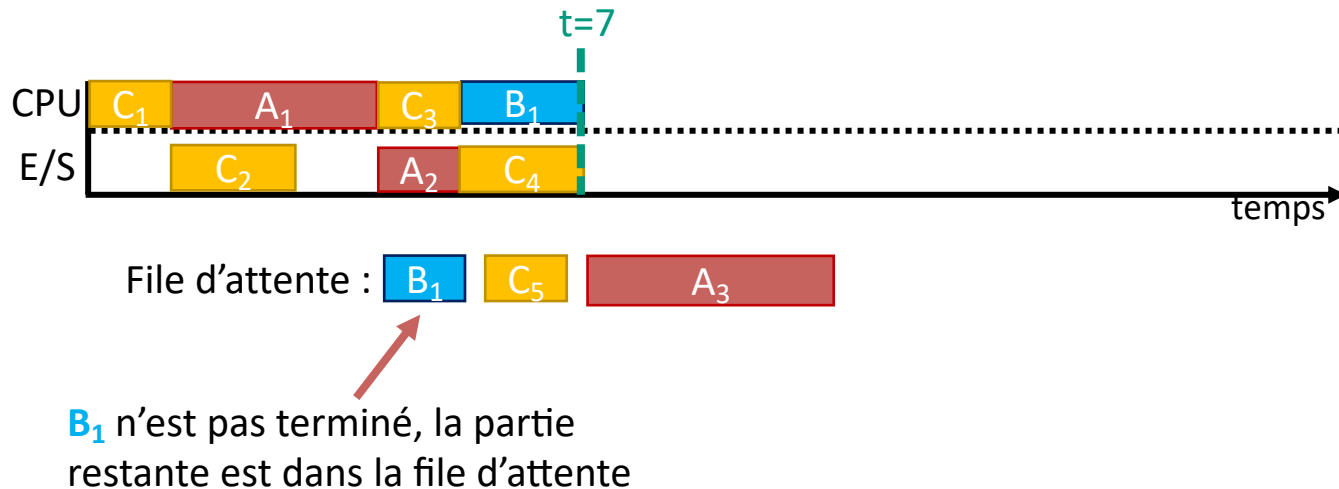
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



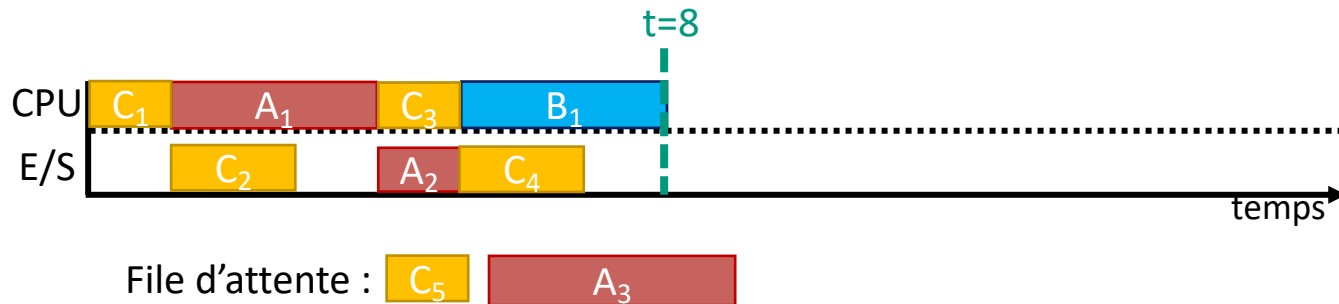
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



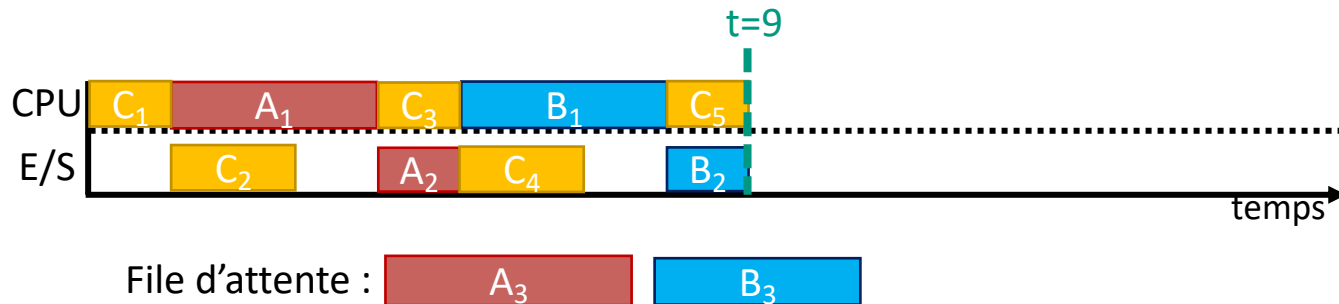
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



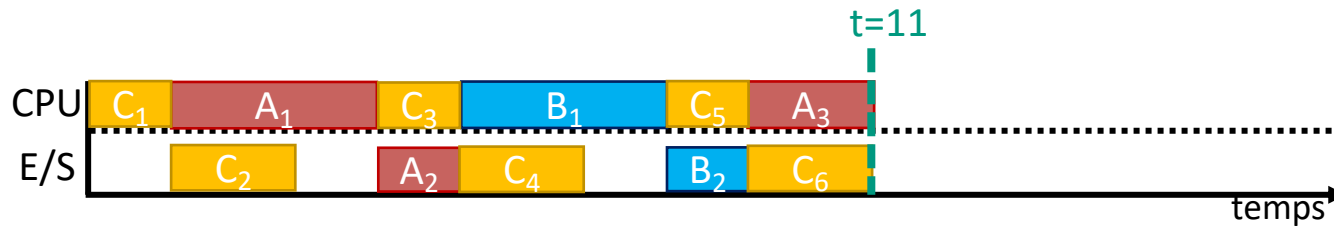
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué -> prêt**



File d'attente : C<sub>7</sub> A<sub>3</sub> B<sub>3</sub>

A<sub>3</sub> n'est pas terminé, la partie restante est dans la file d'attente en 2<sup>ème</sup> position

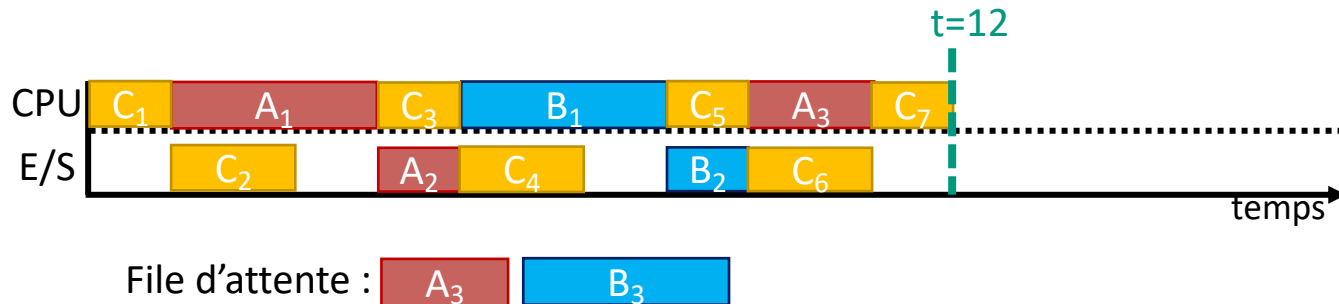
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



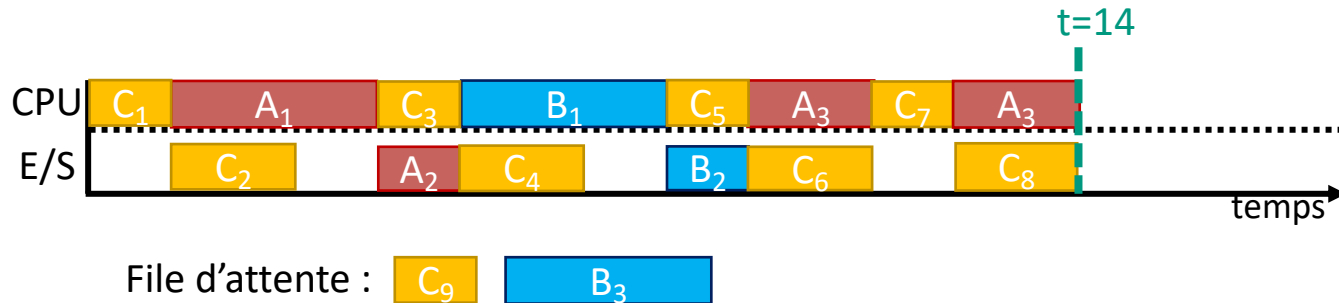
# L'ordonnement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**



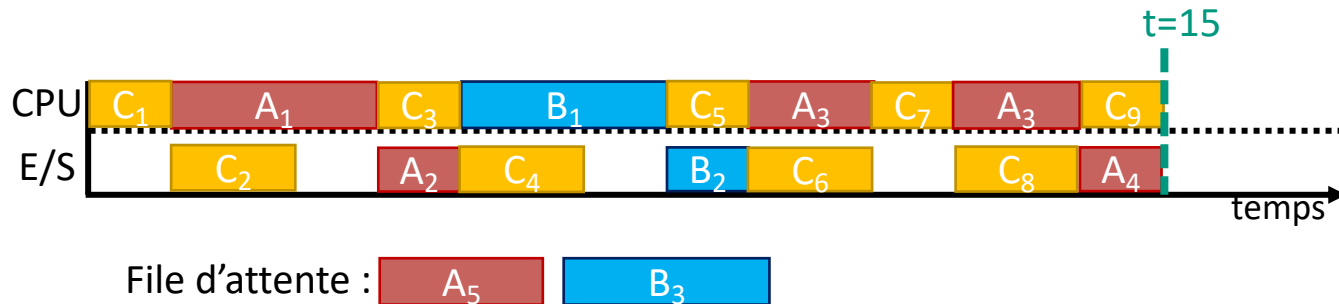
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué** -> **prêt**





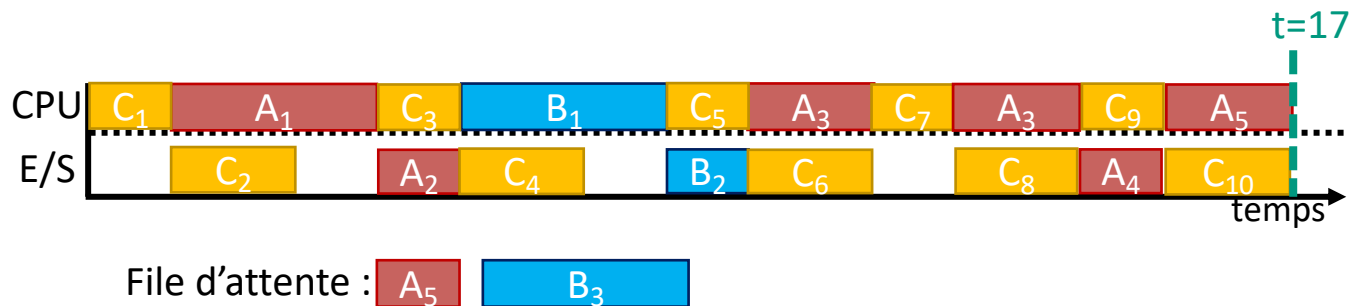
# L'ordonnancement

## Exécution du temps restant suivant le plus court

- Shortest Remaining Time First (SRRF)

- Politique :

- Lancement de la tâche avec le **temps restant le plus court**
- Réévalué à chaque transition d'état **bloqué -> prêt**



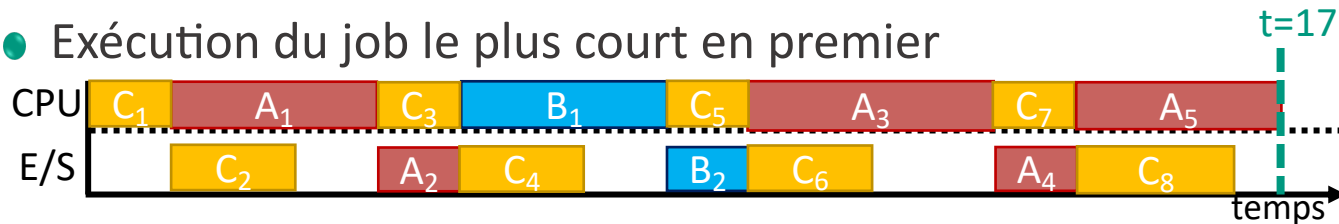
- **Préemptif**

- ✓ Avantageux pour les processus d'E/S sans être trop pénalisant pour les processus de traitement
- ✓ Moins de risque de monopoliser le CPU (grâce à la préemption)

# L'ordonnancement

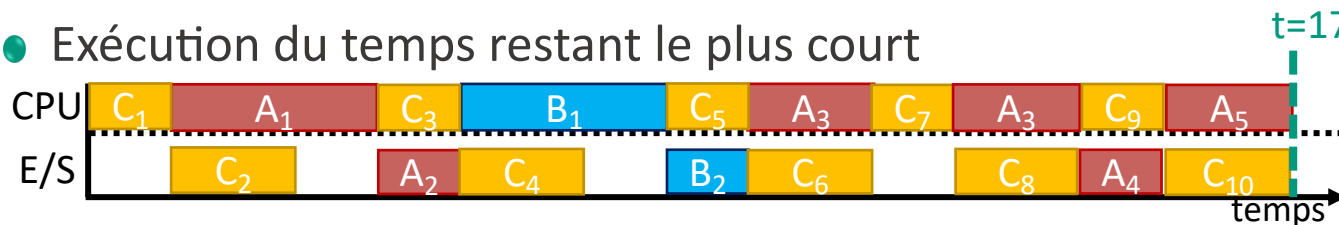
## Un instant...

- Exécution du job le plus court en premier



File d'attente : C<sub>9</sub> B<sub>3</sub>

- Exécution du temps restant le plus court



File d'attente : A<sub>5</sub> B<sub>3</sub>

- Que remarquez-vous ?

- Le processus B n'est presque jamais exécuté
  - B<sub>3</sub> n'est jamais exécuté

## Problématiques

- Problème de **non équité**

- Situation dans laquelle un processus n'est que **très peu** exécuté par rapport aux autres

- Problème de **famine**

- Situation dans laquelle une tâche, bien que prête, se retrouve à attendre **indéfiniment** avant de pouvoir s'exécuter

- Comment remédier à ces problématiques ?

- Une idée ?

# L'ordonnancement

## Exécution du ration de réponse suivant le plus élevé

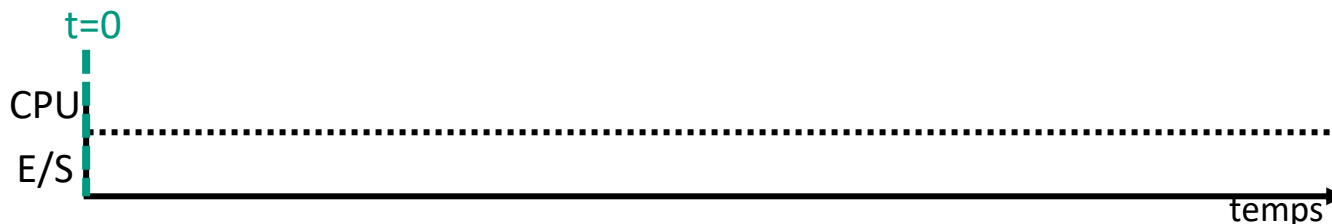
### ● Highest Response Ratio Next (HRRN)

#### ● Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

⚠ On suppose la connaissance du temps d'exécution

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : C<sub>1</sub> A<sub>1</sub> B<sub>1</sub>

	A	B	C
R	1	1	1

# L'ordonnancement

## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ration Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : A<sub>1</sub> B<sub>1</sub>

	A	B	C
R	1,33	1,33	1

# L'ordonnancement

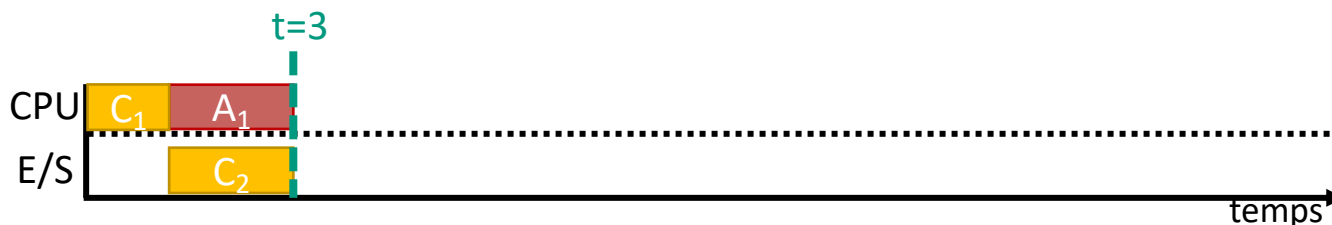
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ration Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : B<sub>1</sub> C<sub>3</sub>

	A	B	C
R	1	2	1

# L'ordonnancement

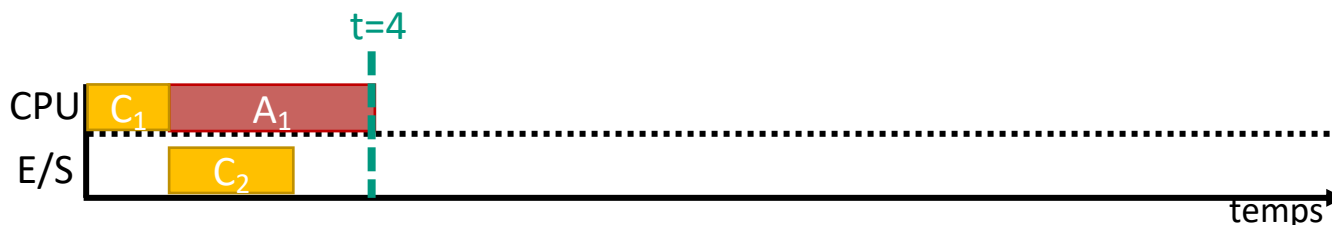
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : B<sub>1</sub> C<sub>3</sub>

	A	B	C
R	1	2,33	2

# L'ordonnancement

## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : C<sub>3</sub> A<sub>3</sub>

	A	B	C
R	1	1	3



# L'ordonnement

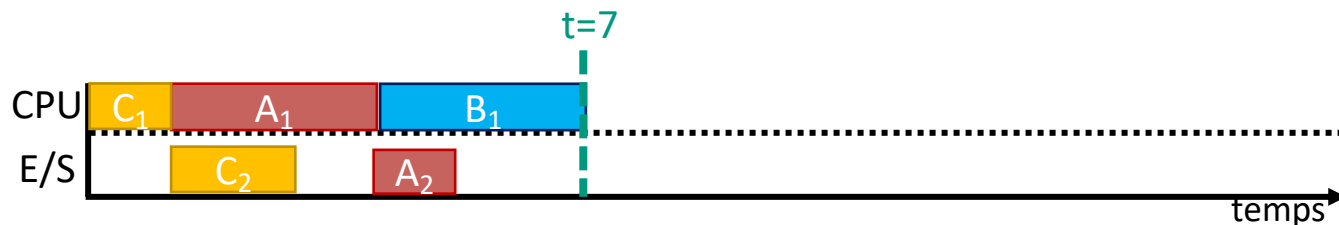
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ration Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : C<sub>3</sub> A<sub>3</sub>

	A	B	C
R	1,5	1	5

# L'ordonnancement

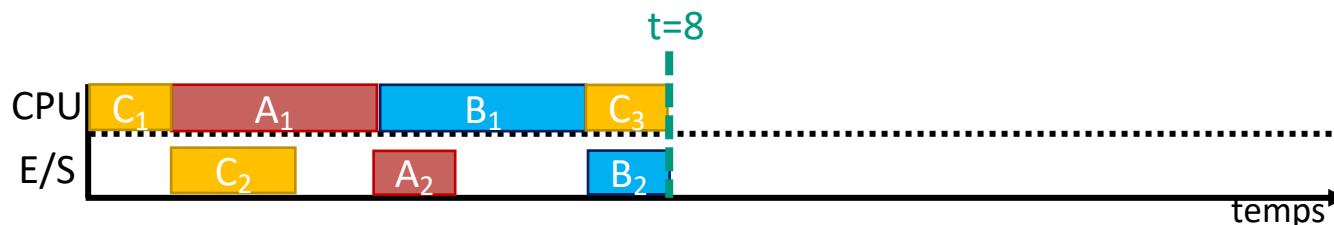
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : A<sub>3</sub> B<sub>3</sub>

	A	B	C
R	1,75	1	1

# L'ordonnancement

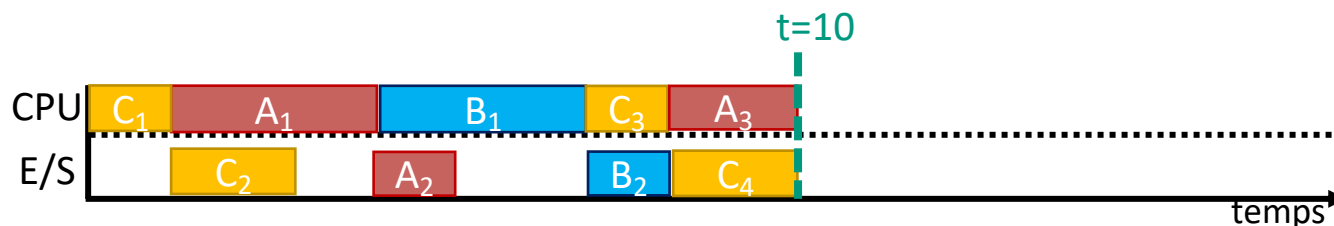
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : B<sub>3</sub> C<sub>5</sub>

	A	B	C
R	1	1,66	1

# L'ordonnancement

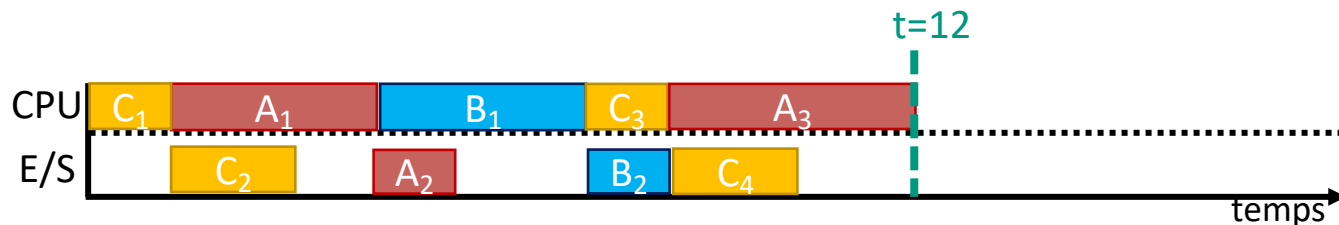
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : B<sub>3</sub> C<sub>5</sub>

	A	B	C
R	1	2,33	3

# L'ordonnancement

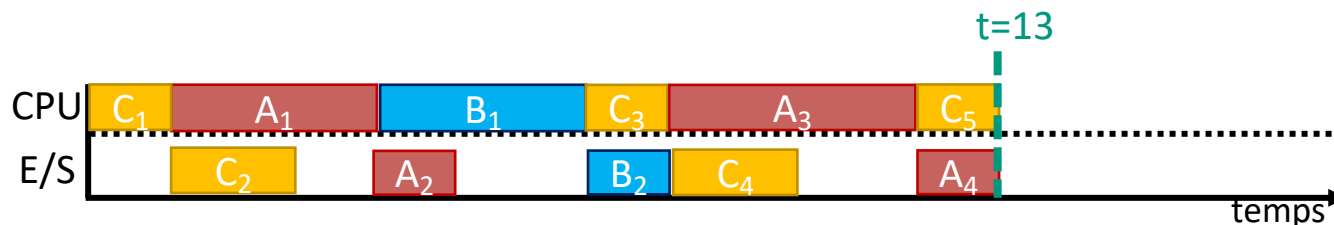
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : B<sub>3</sub> A<sub>5</sub>

	A	B	C
R	1	2,66	1

# L'ordonnancement

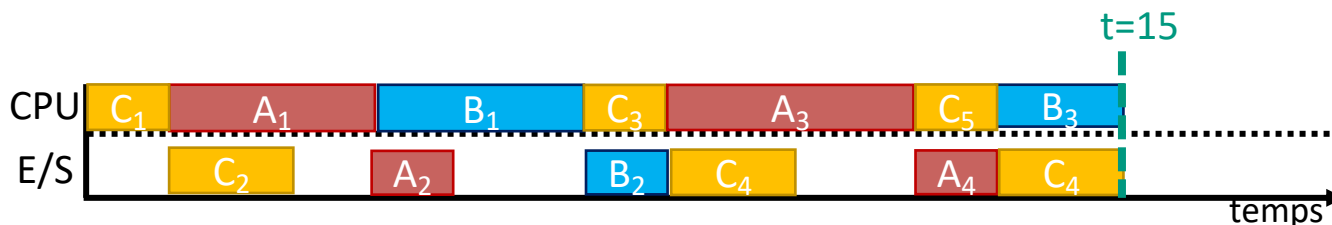
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ration Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : A<sub>5</sub> (red) C<sub>5</sub> (yellow)

	A	B	C
R	1,66	1	1

# L'ordonnancement

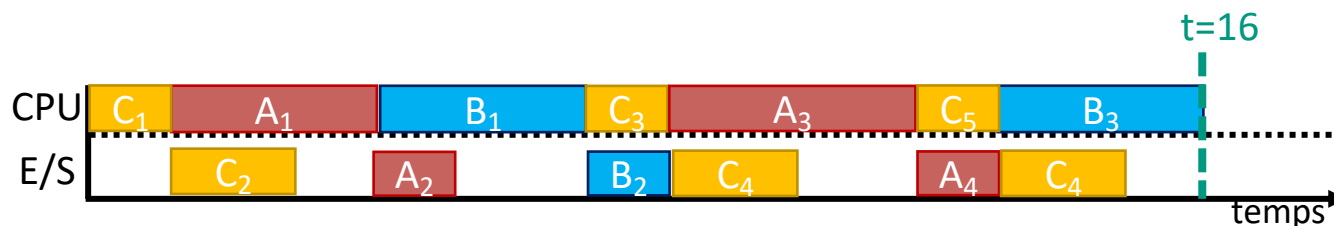
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ration Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : A<sub>5</sub> (red) C<sub>5</sub> (yellow)

	A	B	C
R	2	1	2

# L'ordonnancement

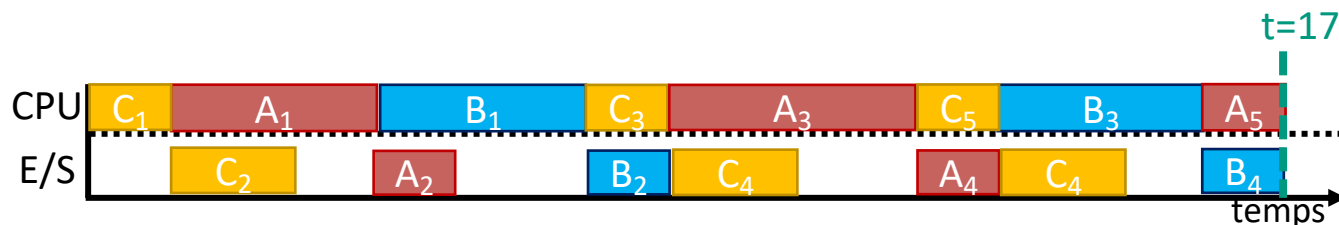
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



File d'attente : C<sub>5</sub> B<sub>5</sub>

	A	B	C
R	1	1	3



# L'ordonnancement

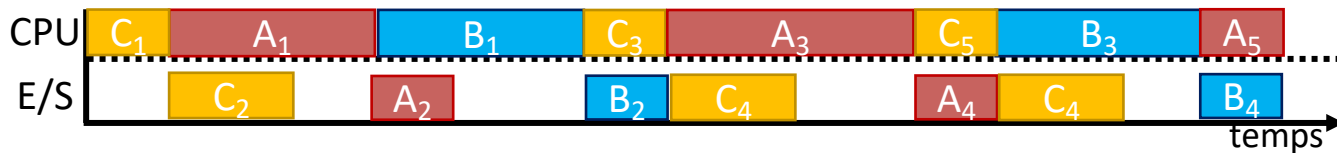
## Exécution du ration de réponse suivant le plus élevé

- Highest Response Ratio Next (HRRN)

- Politique :

- Lancement de la tâche avec le **ration de réponse le plus élevé**

$$R = \frac{\text{Temps d'attente} + \text{Temps d'exécution}}{\text{Temps d'exécution}} = 1 + \frac{\text{Temps d'attente}}{\text{Temps d'exécution}}$$



- ✓ Semble plus équitable
  - Privilégies les processus court sans trop pénaliser les processus longs
- ✓ Supprime le problème de famine
  - Plus un processus attend, plus il augmente ses chances d'être choisi
  - Sorte de priorité...
- ✗ Risque de monopolisation du processeur (non préemptif)

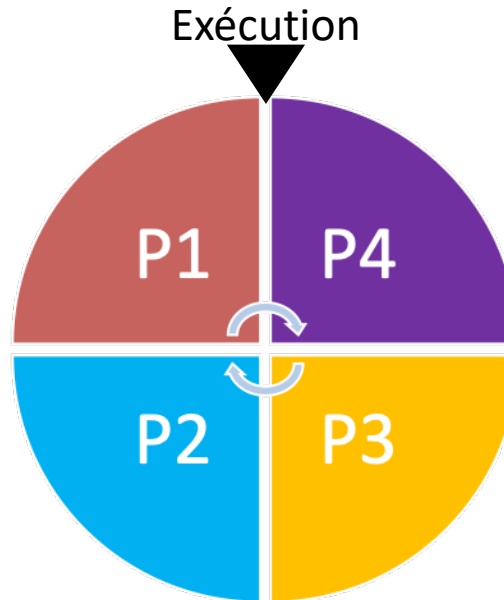
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



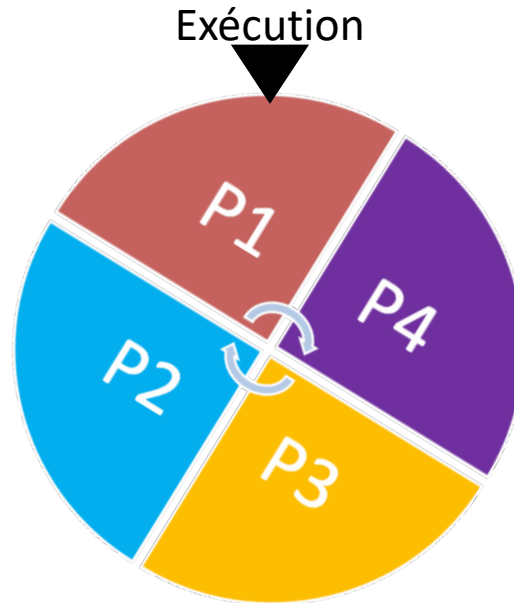
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



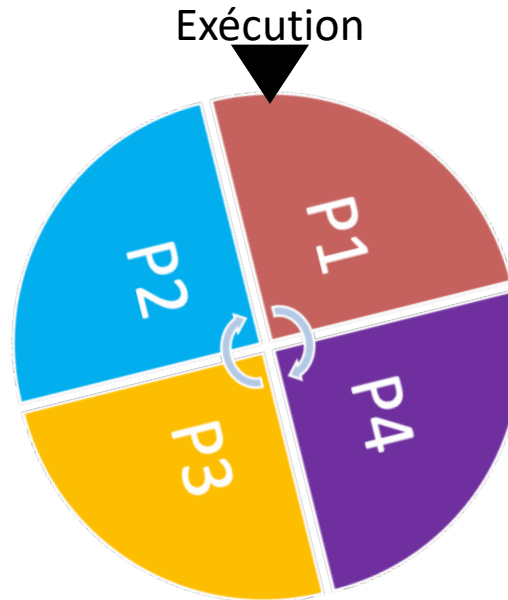
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



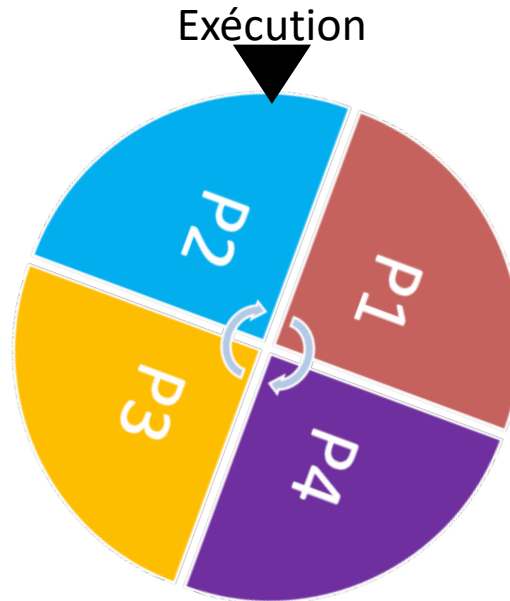
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



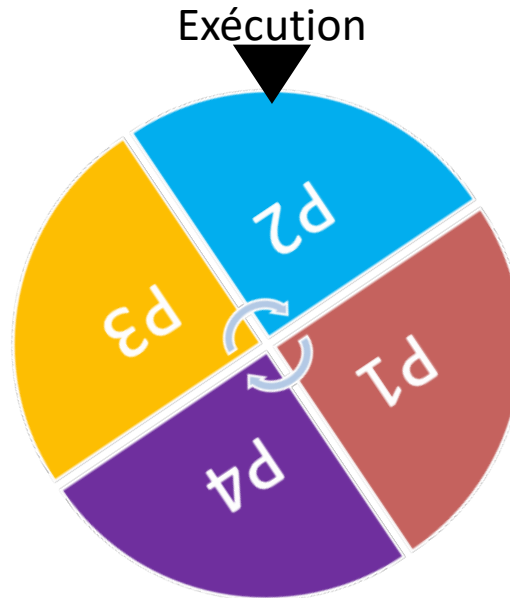
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



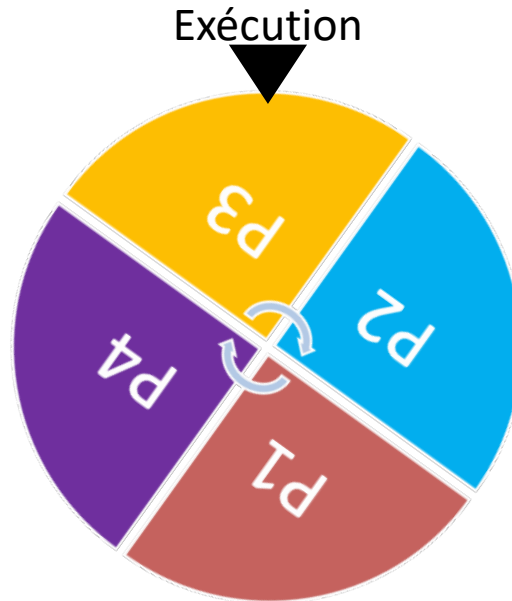
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum



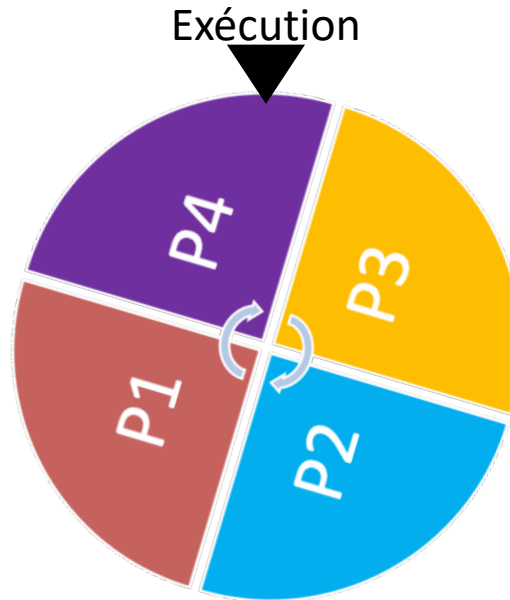
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum





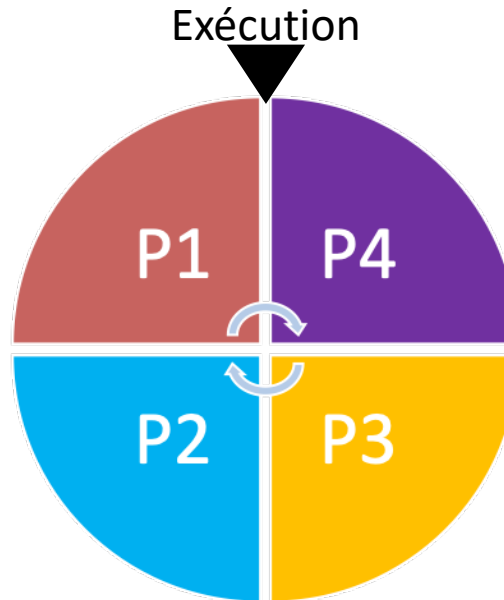
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale q** (time quantum)
    - **Préemption** si une tâche dépasse du quantum



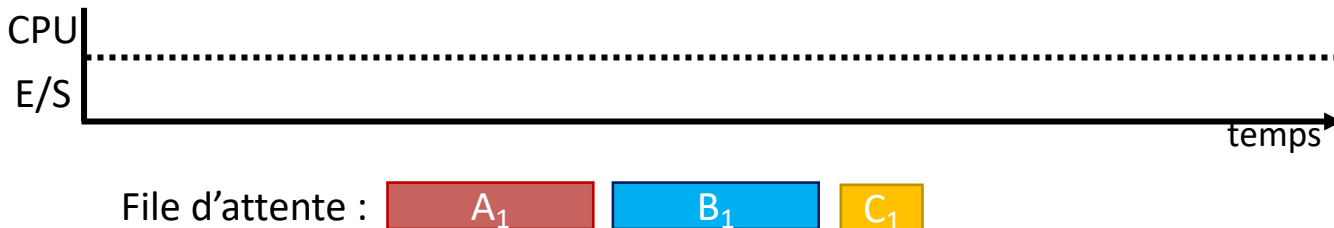
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum
    - **L'horloge matériel** est souvent utilisée pour définir  $q$
    - L'horloge fournit des interruptions périodiques (à chaque top)
    - Exemple :  **$q=2$  tops** d'horloge



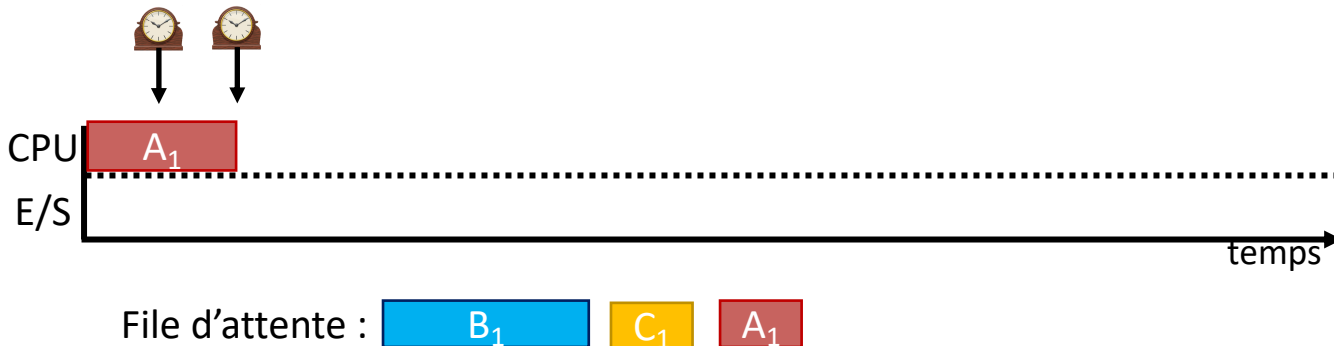
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum
    - **L'horloge matériel** est souvent utilisée pour définir  $q$
    - L'horloge fournit des interruptions périodiques (à chaque top)
    - Exemple :  **$q=2$  tops d'horloge**



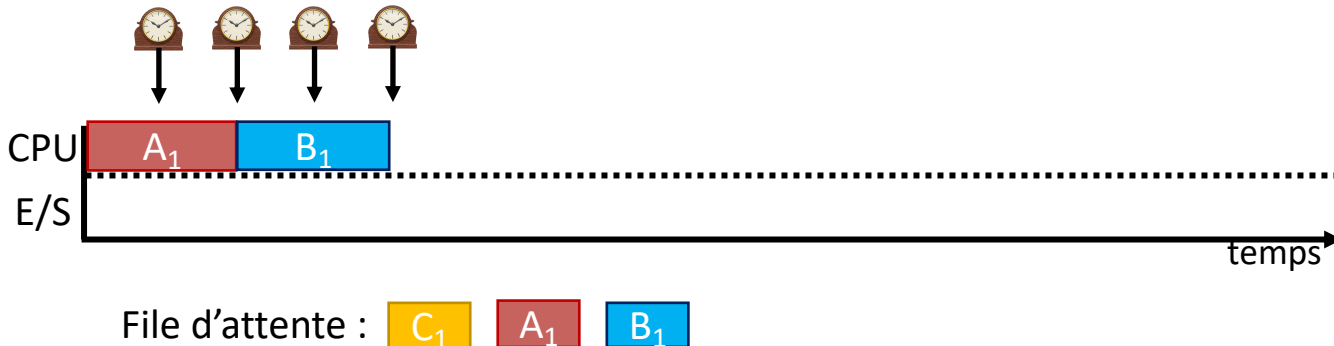
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum
    - **L'horloge matériel** est souvent utilisée pour définir  $q$
    - L'horloge fournit des interruptions périodiques (à chaque top)
    - Exemple :  **$q=2$  tops d'horloge**



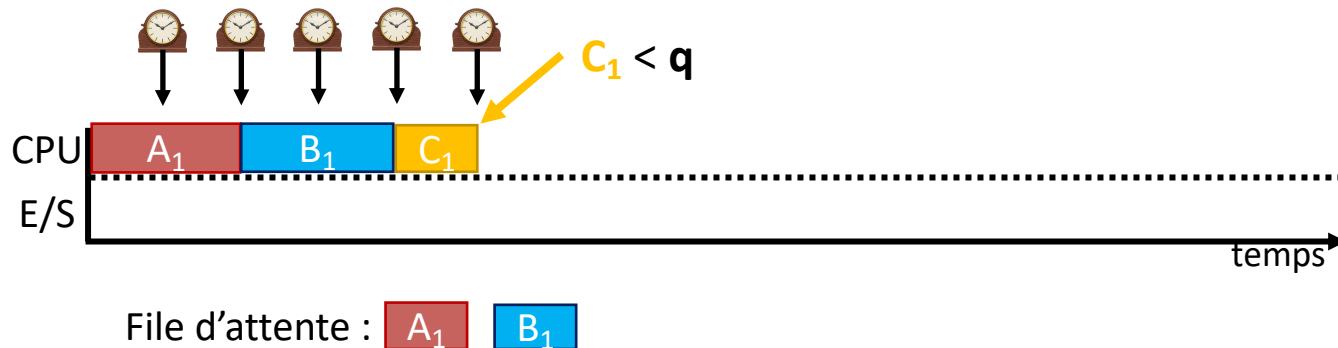
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**



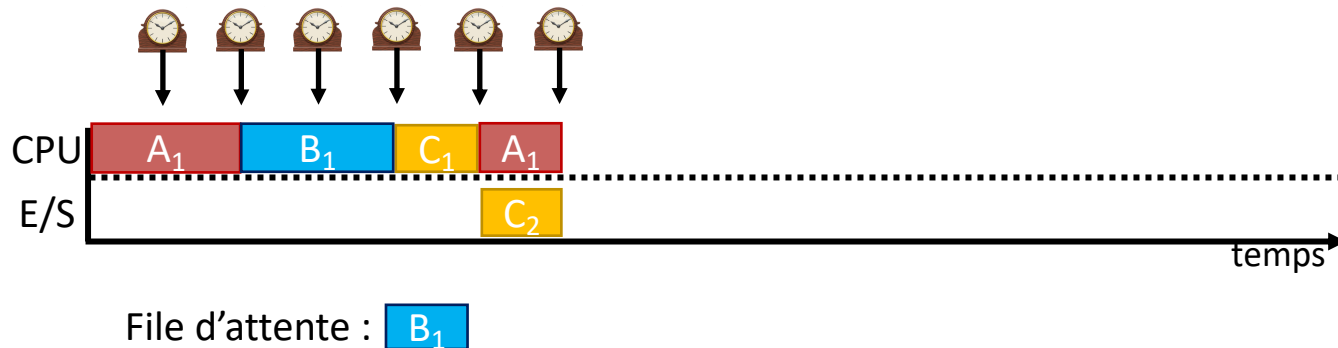
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur à **tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum
    - **L'horloge matériel** est souvent utilisée pour définir  $q$
    - L'horloge fournit des interruptions périodiques (à chaque top)
    - Exemple :  **$q=2$  tops d'horloge**



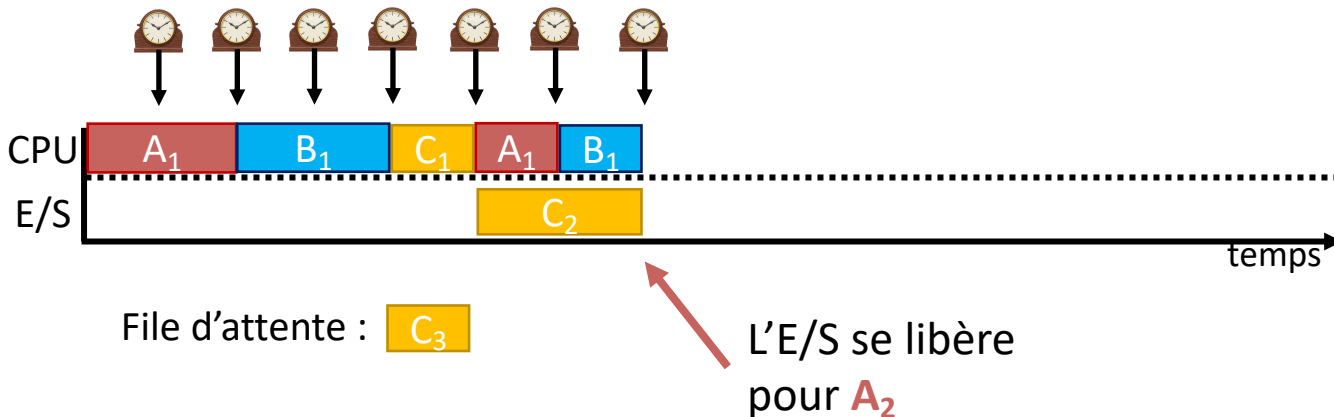
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**



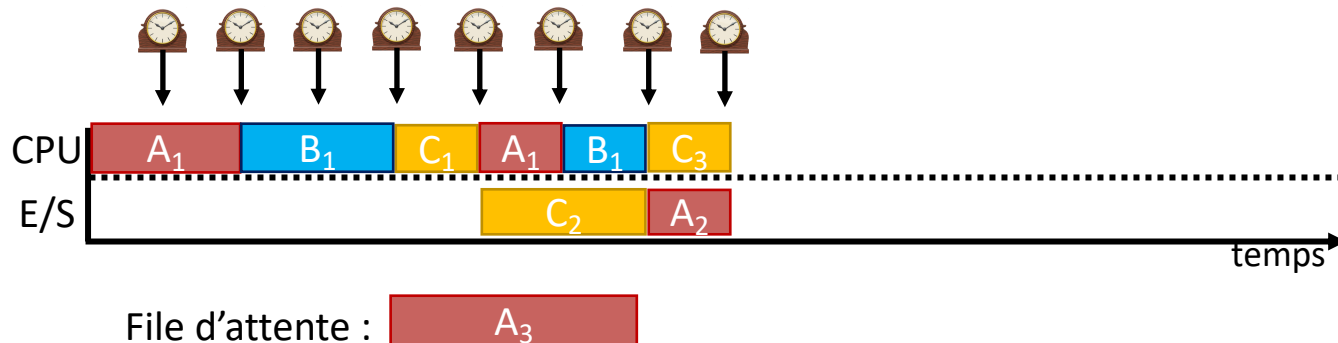
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
    - Pour une **durée maximale  $q$**  (time quantum)
    - **Préemption** si une tâche dépasse du quantum
    - **L'horloge matériel** est souvent utilisée pour définir  $q$
    - L'horloge fournit des interruptions périodiques (à chaque top)
    - Exemple :  **$q=2$  tops d'horloge**





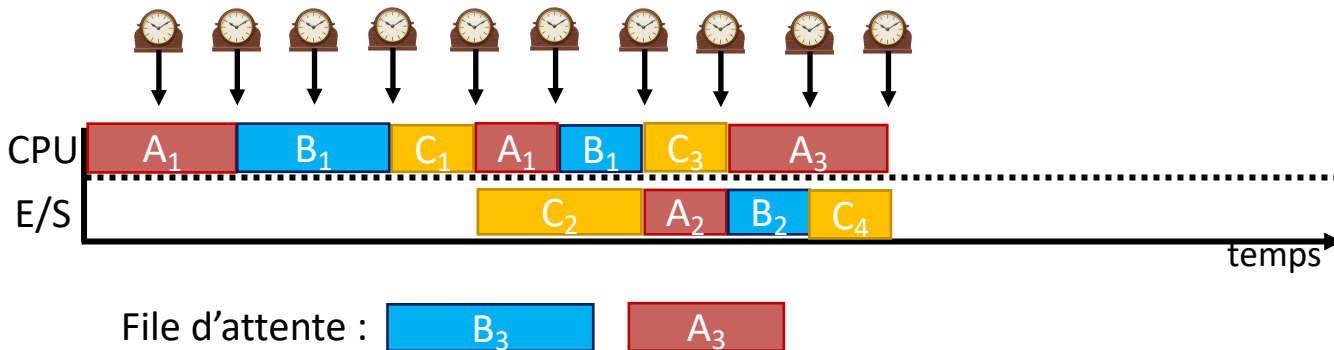
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**



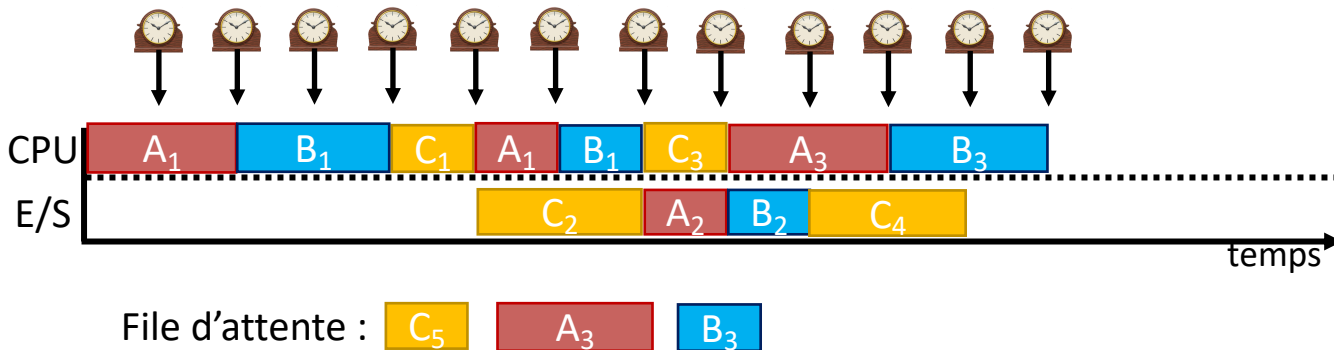
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**



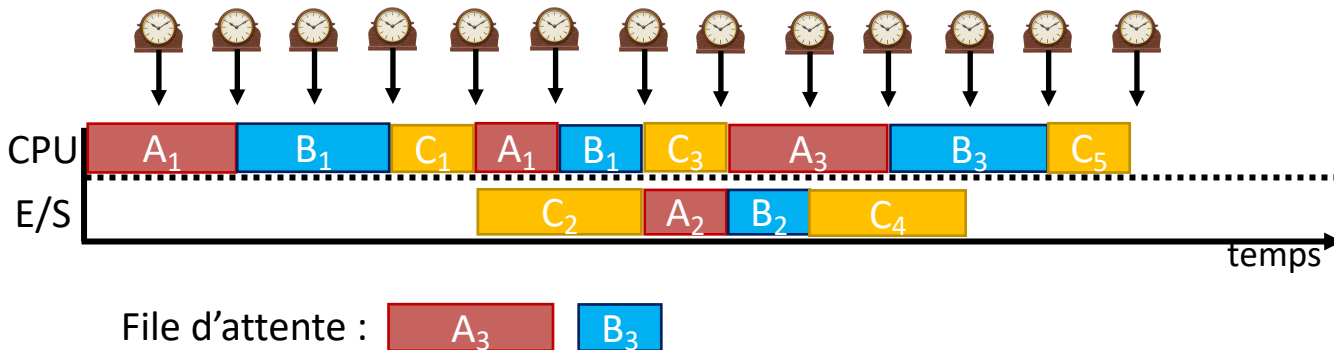
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)

- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**



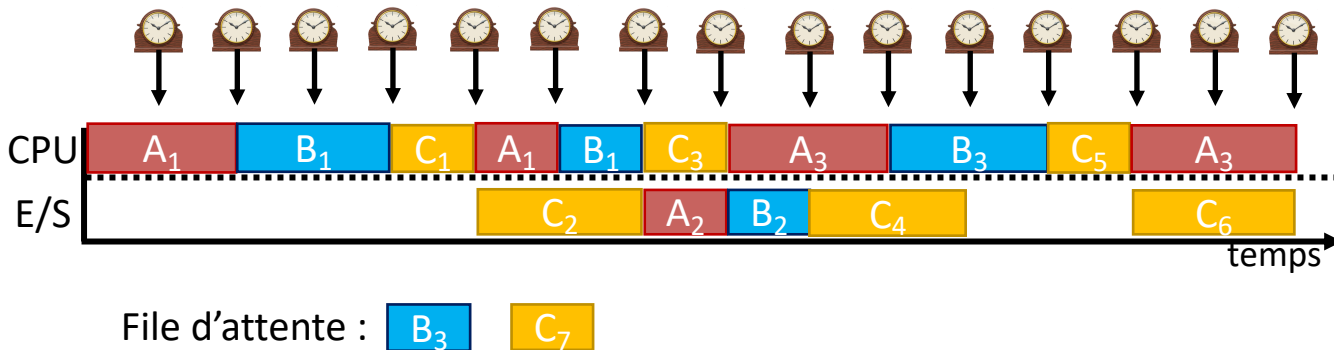
# L'ordonnancement

## Ordonnancement de type tourniquet

- Round Robin (RR)




- Politique :

- Donner du temps processeur **à tour de rôle** à chaque processus
- Pour une **durée maximale  $q$**  (time quantum)
- **Préemption** si une tâche dépasse du quantum
- **L'horloge matériel** est souvent utilisée pour définir  $q$
- L'horloge fournit des interruptions périodiques (à chaque top)
- Exemple :  **$q=2$  tops d'horloge**






## Ordonnancement de type tourniquet

- Round Robin (RR)

-  Equitable
-  Pas de famine
-  Difficile de définir **q** optimal
- Exemple :
  - Supposons que changer de processus dure **1ms**
  - Si **q=4ms**, après **4ms** le processeur doit gaspiller **1ms** pour changer
  - Cela revient à gaspiller **20%**, c'est beaucoup !
  
  - Pour améliorer l'UC, on peut définir **q=100ms**
  - Le gaspillage est alors de **1%**
  - Mais que se passe t-il si **50** requête arrivent presque en même temps ?
  - La **50<sup>ème</sup>** tâche devra attendre **5 secondes** (50x100)

## Ordonnancement de type tourniquet

- Round Robin (RR)

-  Equitable
-  Pas de famine
-  Difficile de définir **q** optimal
- Conclusion :
  - Si le quantum est trop court, il y a beaucoup de changements de processus et **l'efficacité** de l'UC **chute**
  - Si le quantum est trop long, les **temps de réponses** aux requêtes interactives deviennent **trop longs**
  - Il faut trouver un **compromis** raisonnable (pour l'exemple : 20-50 ms)

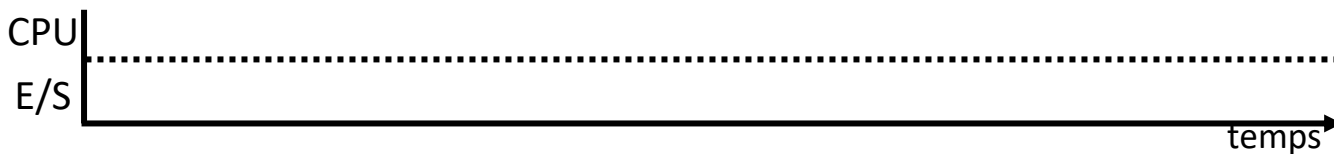
## Ordonnancement par priorité

- Jusqu'à présent on a supposé que tous les processus ont une importance égale
  - C'est rarement le cas dans des environnements interactifs
- L'ordonnancement par priorité
  - Principe
    - Chaque processus détient un **niveau de priorité**
    - L'ordonnanceur choisit le processus avec la **priorité** la plus **élevée**
  - Exemple : un environnement Moodle multi-utilisateurs à l'UHA
    - 1. L'enseignant du cours
    - 2. Les étudiants inscrits au cours
    - 3. Les autres étudiants de l'UHA
    - 4. Les autres enseignants
    - 5. Les chefs des composantes d'enseignement

# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité
  - Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
  - Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge



	A	B	C
Priorité	3	2	1



# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité
  - Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
  - Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge



	A	B	C
Priorité	2	2	1

# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité

- Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
- Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge



	A	B	C
Priorité	1	2	1

# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité

- Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
- Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge



	A	B	C
Priorité	1	1	1

# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité

- Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
- Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge



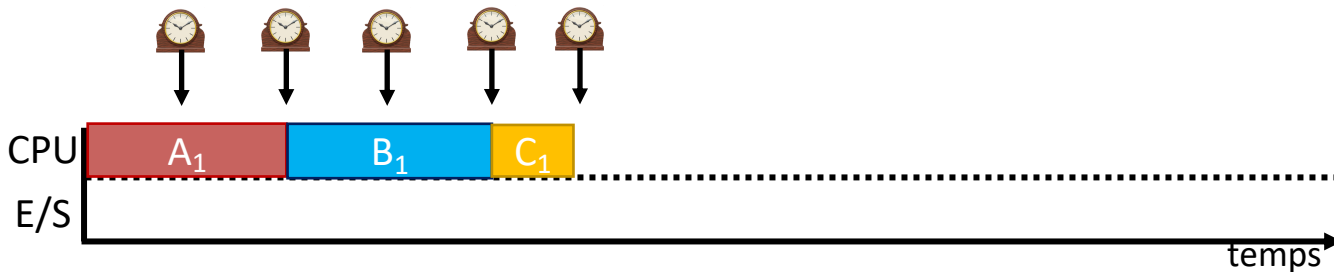
	A	B	C
Priorité	1	0	1

# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité

- Mais si un enseignant monopolise le système, comment faire pour qu'un étudiant puisse quand même l'utiliser ?
- Afin d'éviter qu'un processus prioritaire ne s'exécute indéfiniment, une idée est de réduire sa priorité à chaque top d'horloge

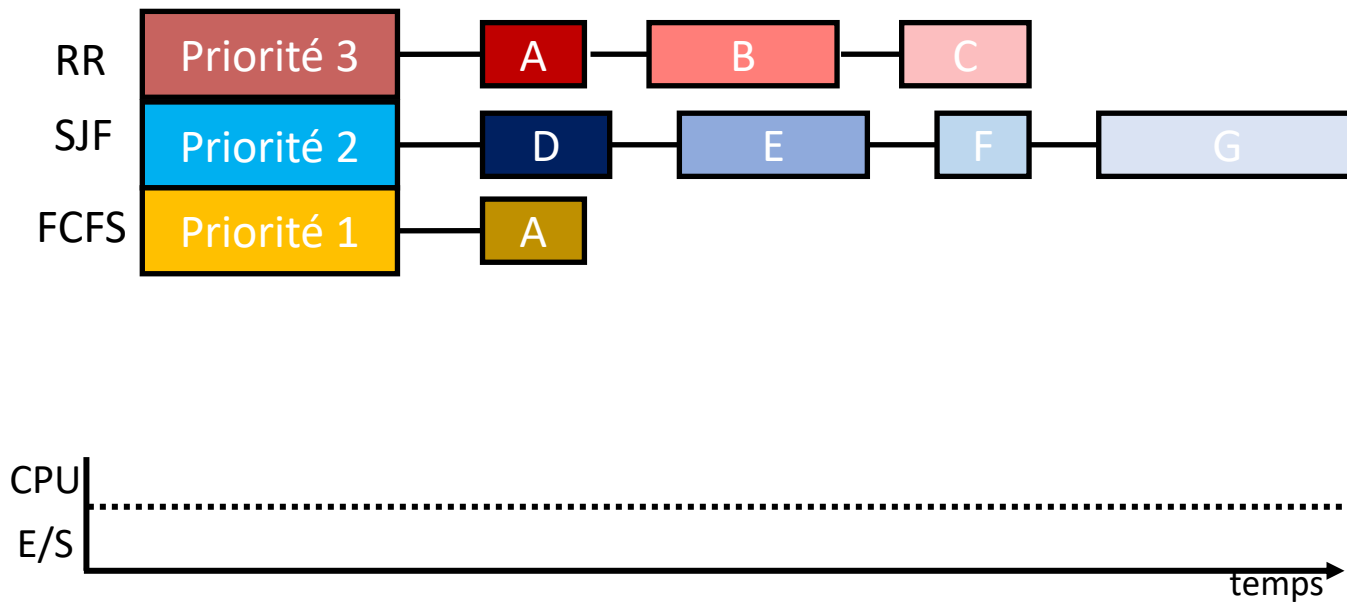


	A	B	C
Priorité	1	0	0

# L'ordonnancement

## Ordonnancement par priorité

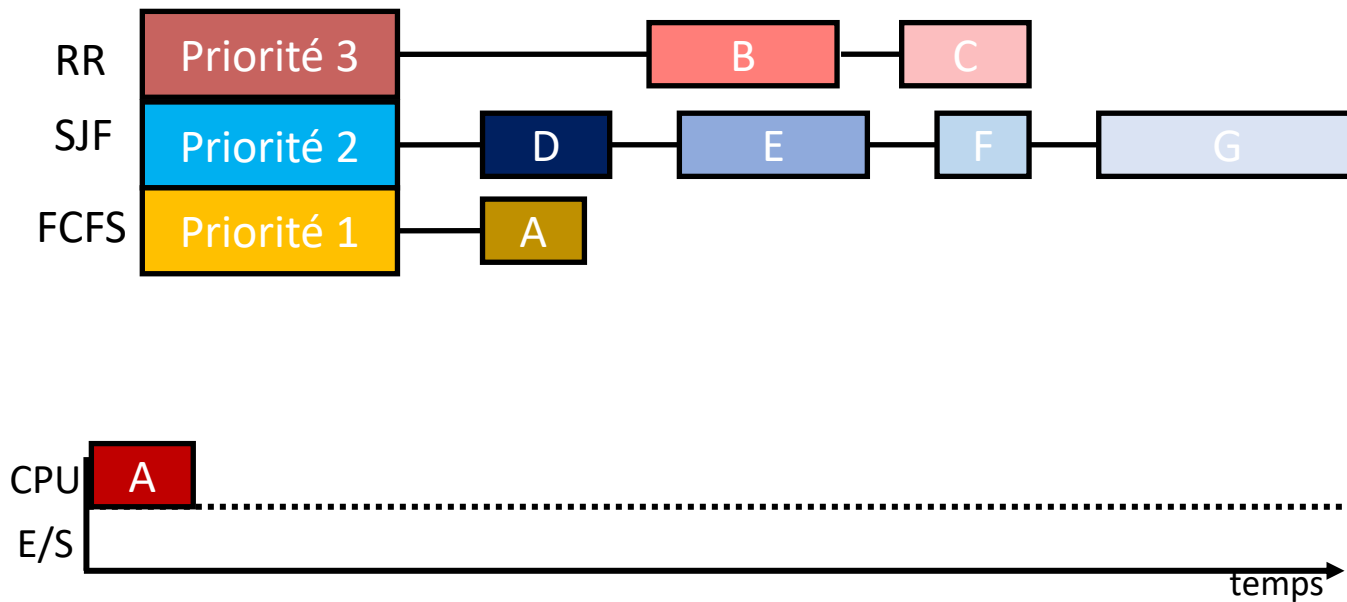
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

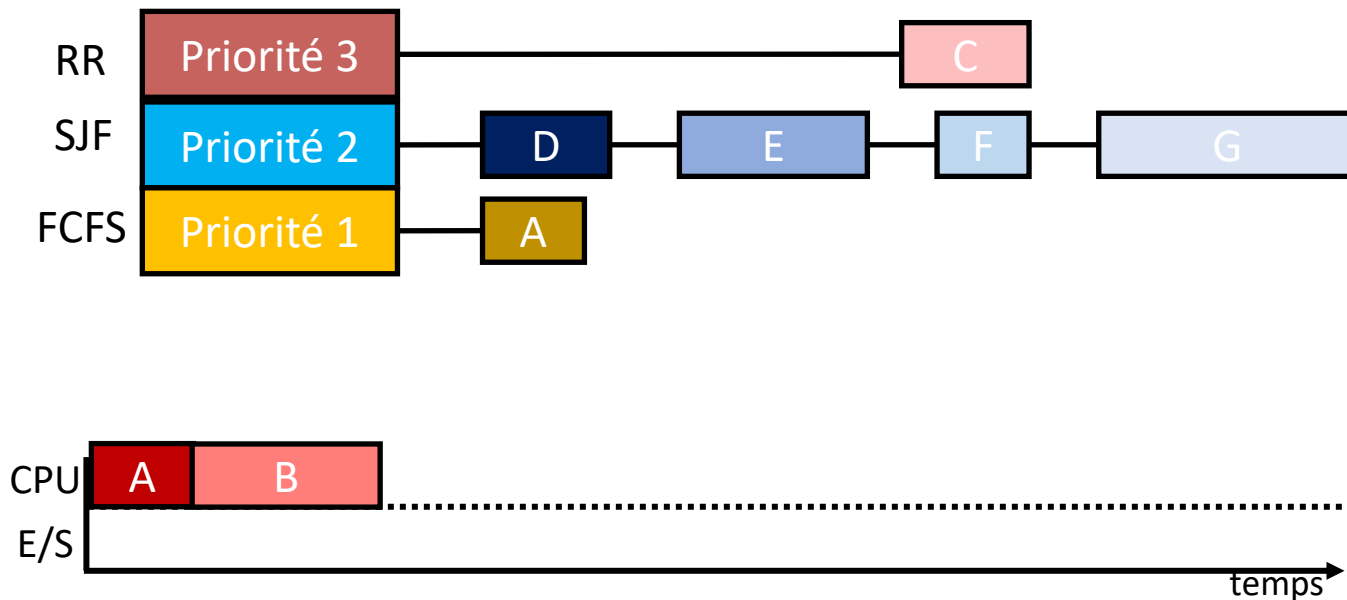
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés

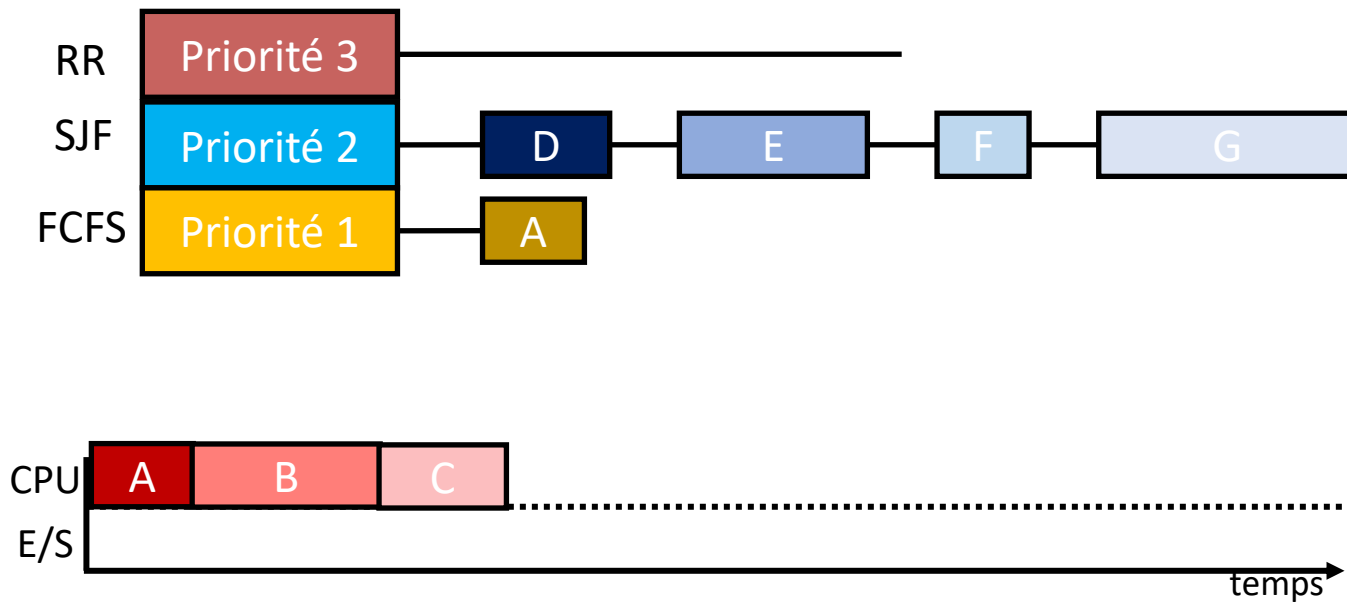




# L'ordonnancement

## Ordonnancement par priorité

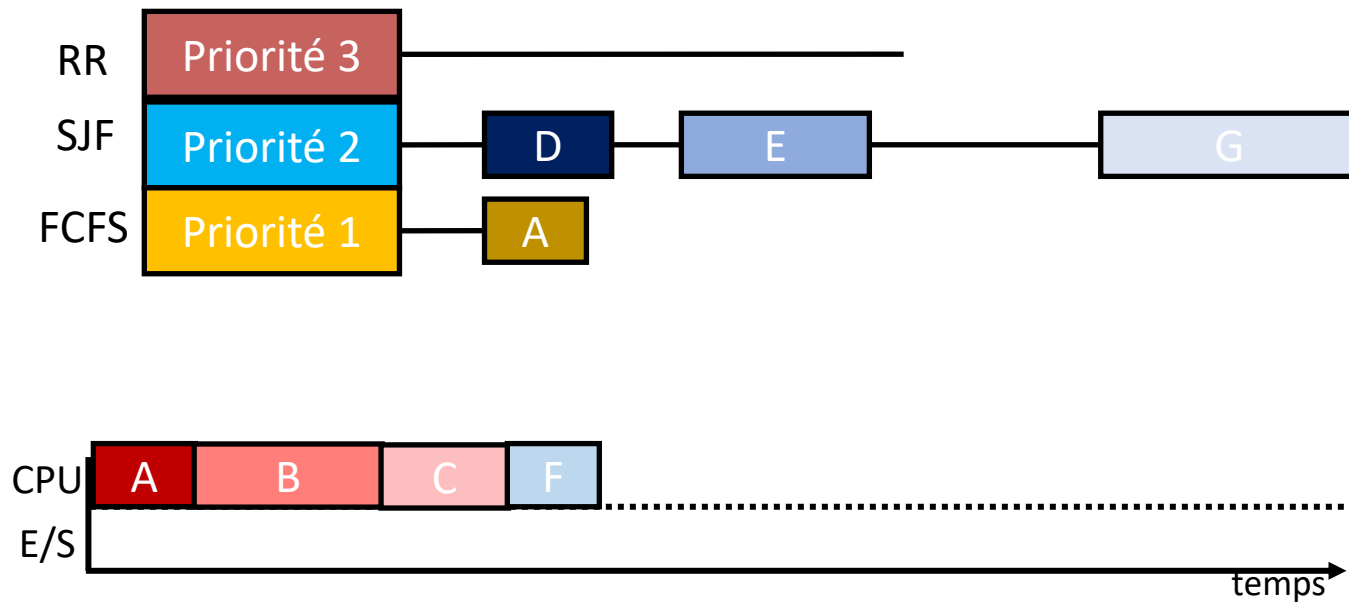
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

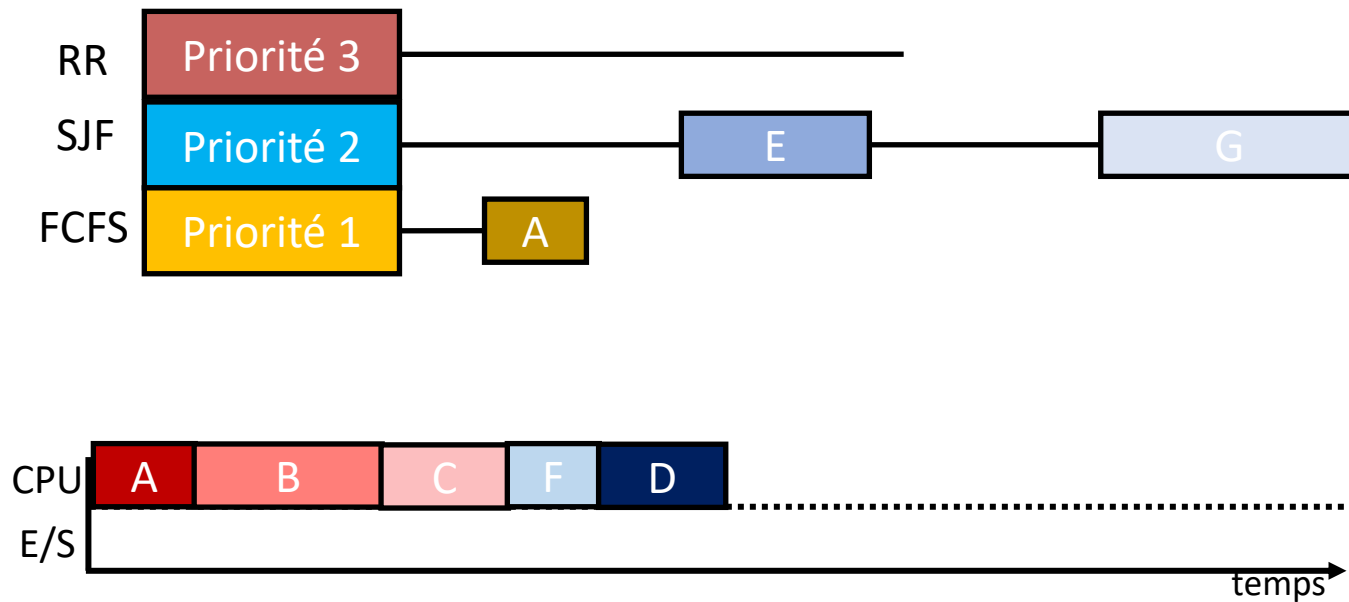
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

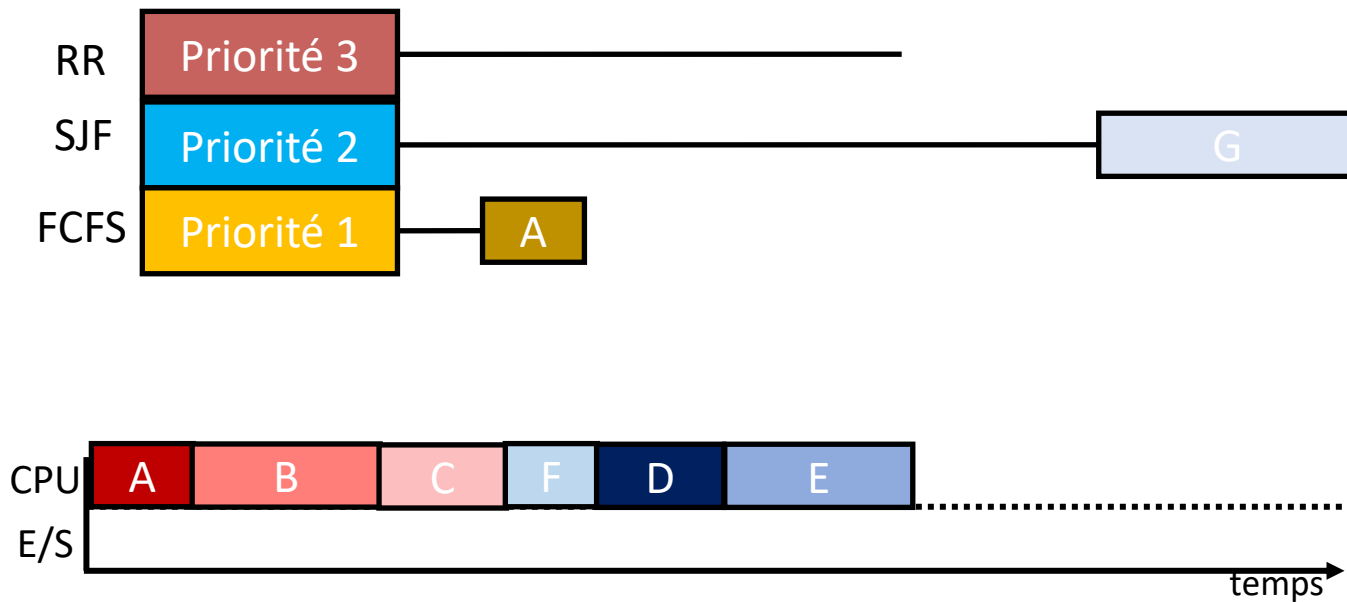
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

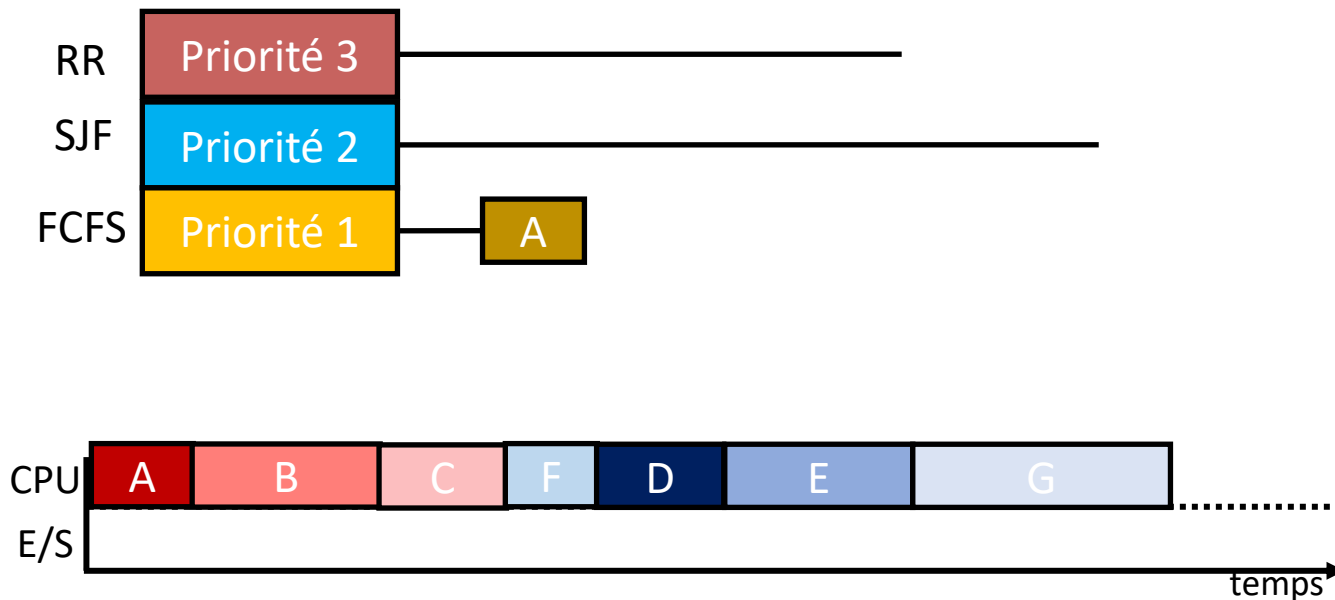
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

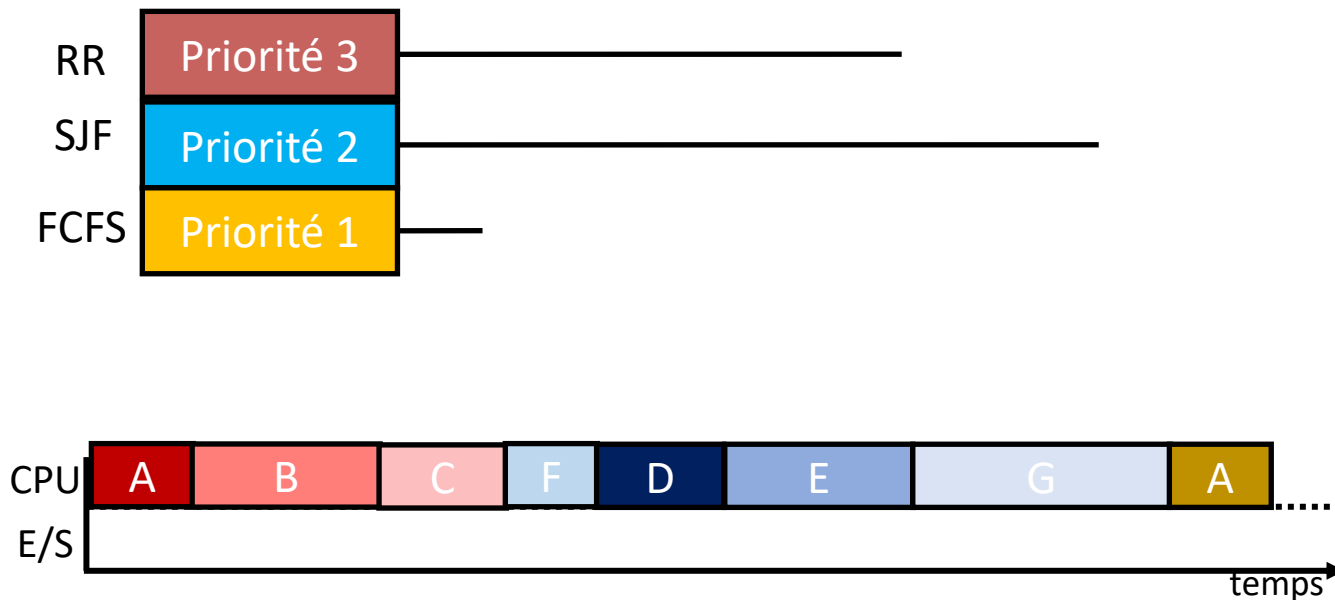
- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement par priorité

- L'ordonnancement par priorité
  - Regroupement des processus dans des catégories de priorité
  - L'ordonnancement se fait par catégorie
    - Plusieurs algorithmes peuvent être utilisés



# L'ordonnancement

## Ordonnancement équitable

- Jusqu'à présent, nous n'avons pas vraiment considéré les propriétaires des processus dans l'équité
  - Si l'utilisateur 1 lance 9 processus, et l'utilisateur 2 lance 1 processus
  - 90% de l'UC est utilisée par l'utilisateur 1, 10% par l'utilisateur 2
  - Certains systèmes tiennent compte des **propriétaires** et allouent un temps processeur à chacun
  - Exemple :

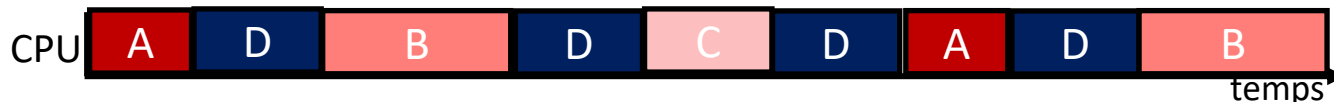
Utilisateur 1 :



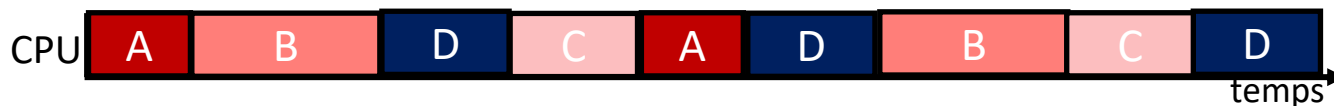
Utilisateur 2 :



- 50% - 50%



- $\frac{2}{3} - \frac{1}{3}$



## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)
  - Algorithme d'ordonnancement temps réel
  - Préemptif
  - Pour les processus cycliques
- Politique
  - L'algorithme assigne à chaque processus une priorité fixe égale à la fréquence de l'occurrence de l'événement qui le déclenche
  - Exemple :
    - Un processus qui doit s'exécuter toutes les **30ms (33 fois/s)** a une priorité de **33**
    - Un processus qui doit s'exécuter toutes les **40ms (25 fois/s)** a une priorité de **25**



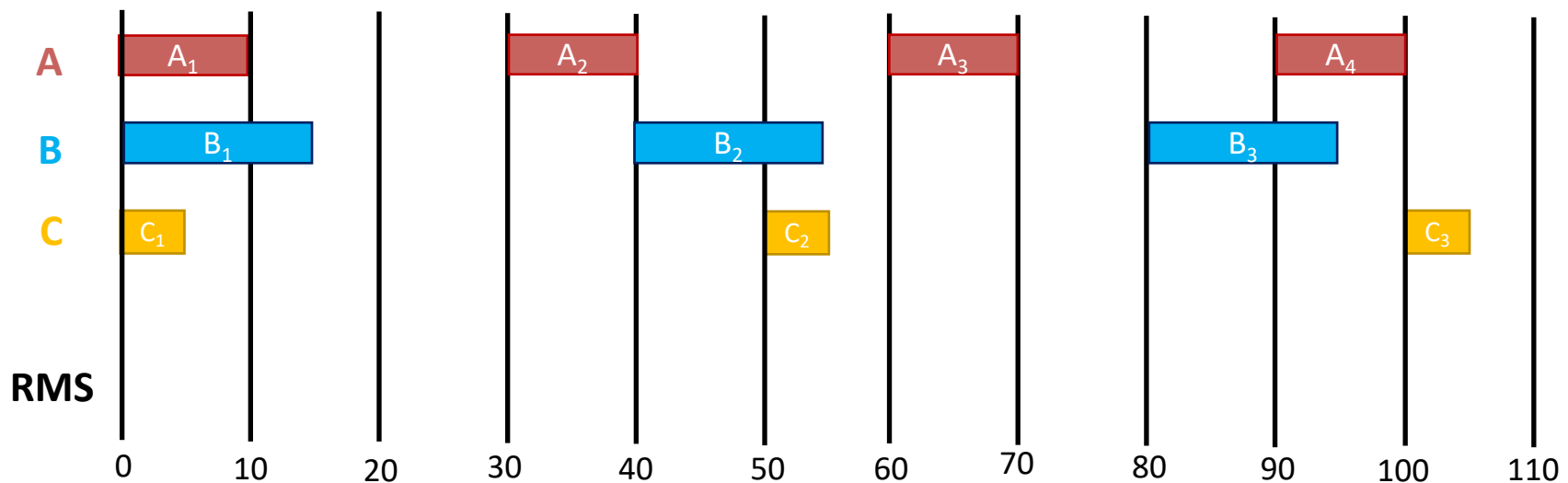
# L'ordonnancement

## Ordonnancement RMS

### ● Rate Monotonic Scheduling (RMS)

#### ● Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



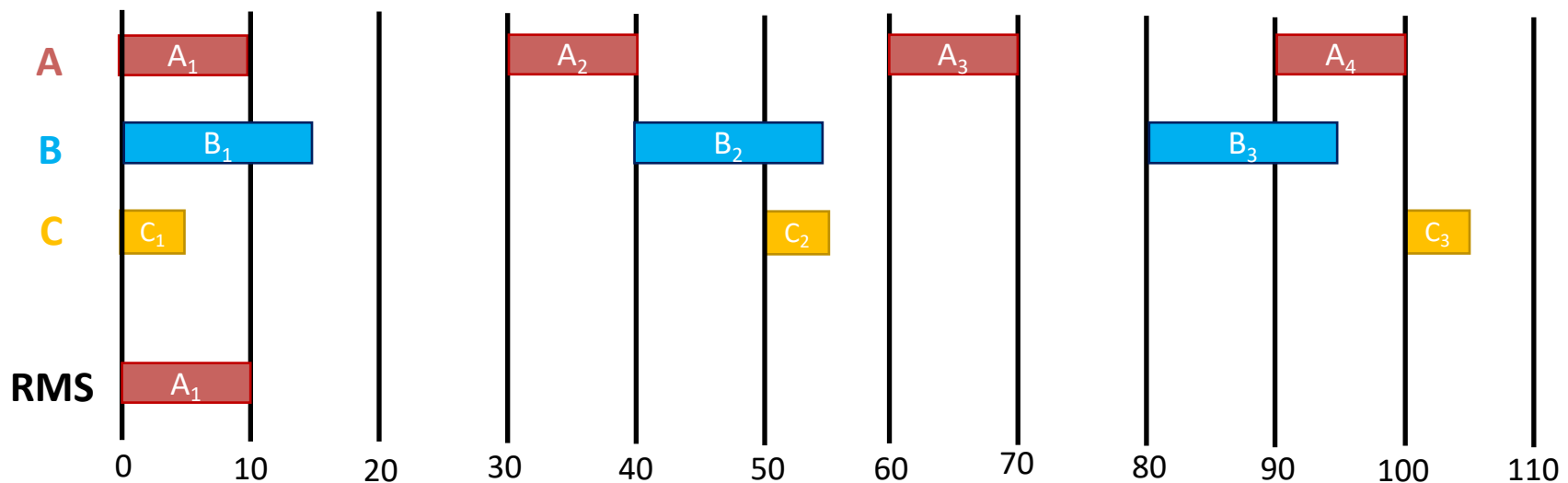
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



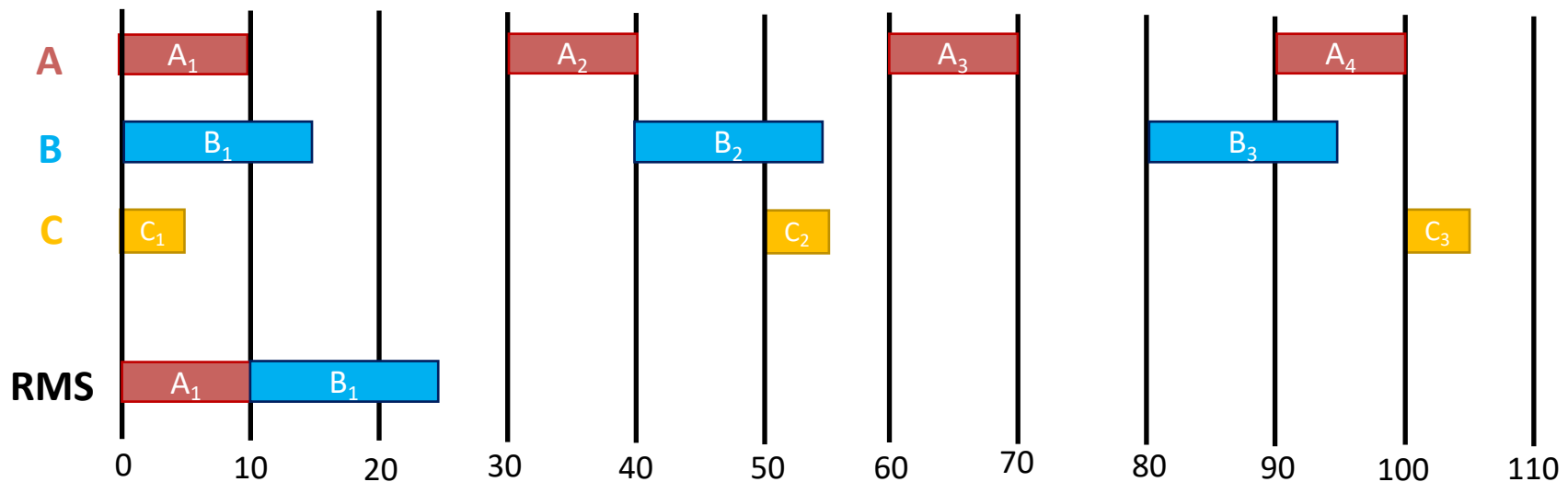
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
    - Chaque film a un framerate (image/s) différent (33, 25, 20)
    - Un processus est responsable de chaque film pour afficher chaque image
    - Le temps pour afficher une image varie selon les processus (10, 15, 5)



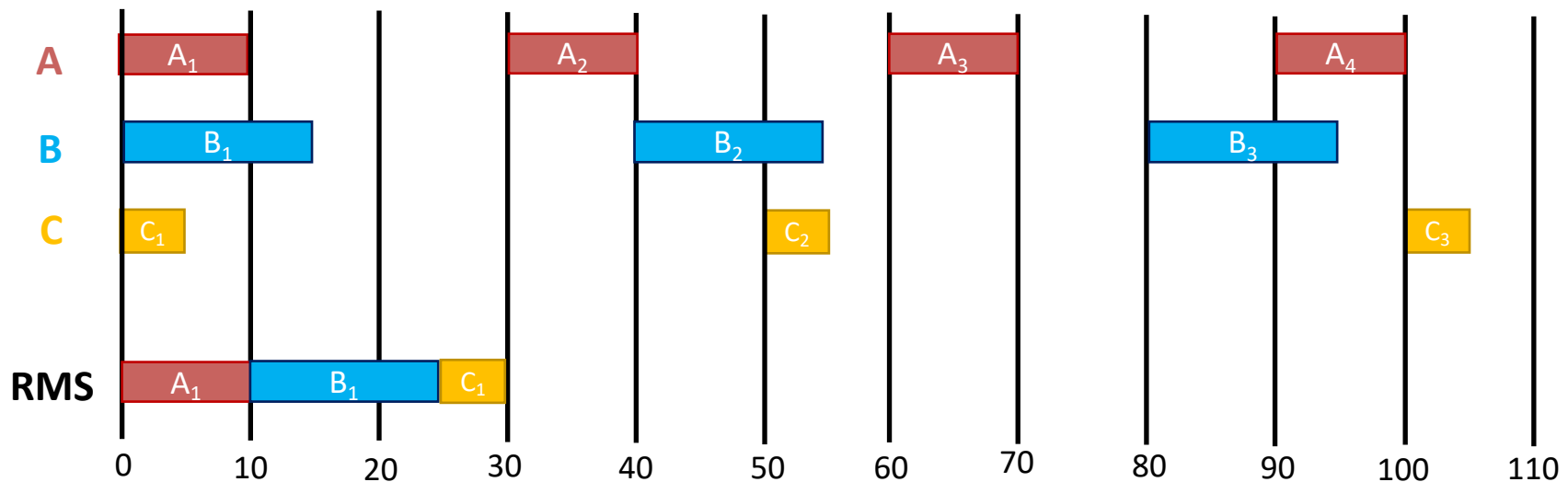
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)

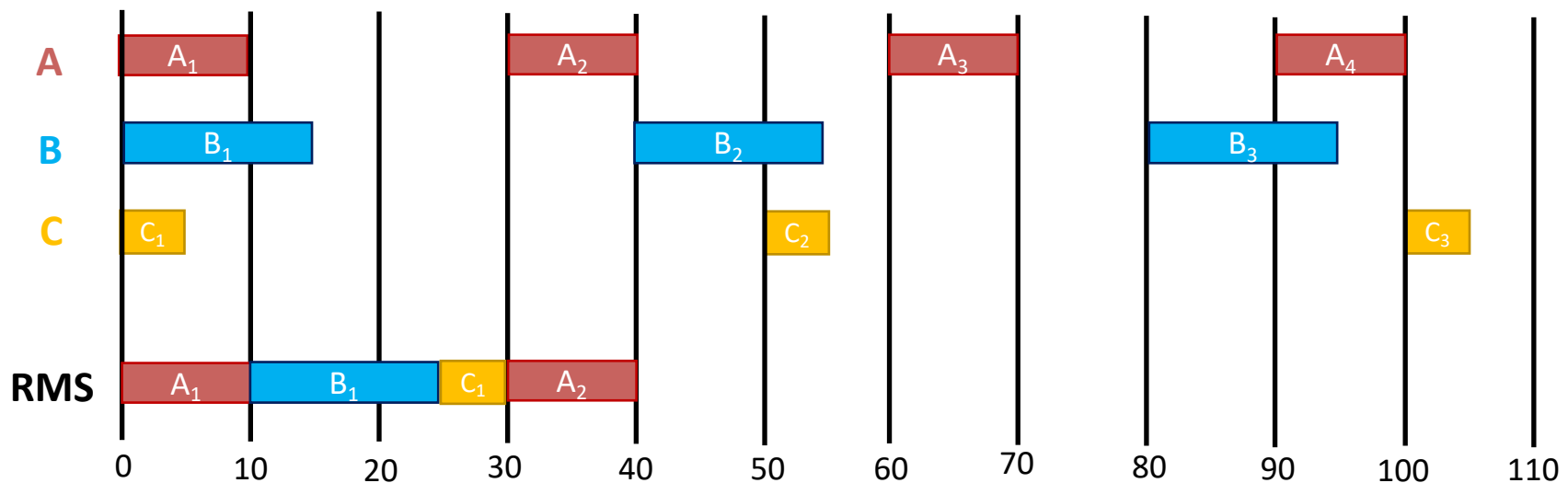


# L'ordonnancement

## Ordonnancement RMS

### ● Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)
  - 3 utilisateurs veulent regarder un film au même instant
  - Chaque film a un framerate (image/s) différent (33, 25, 20)
  - Un processus est responsable de chaque film pour afficher chaque image
  - Le temps pour afficher une image varie selon les processus (10, 15, 5)



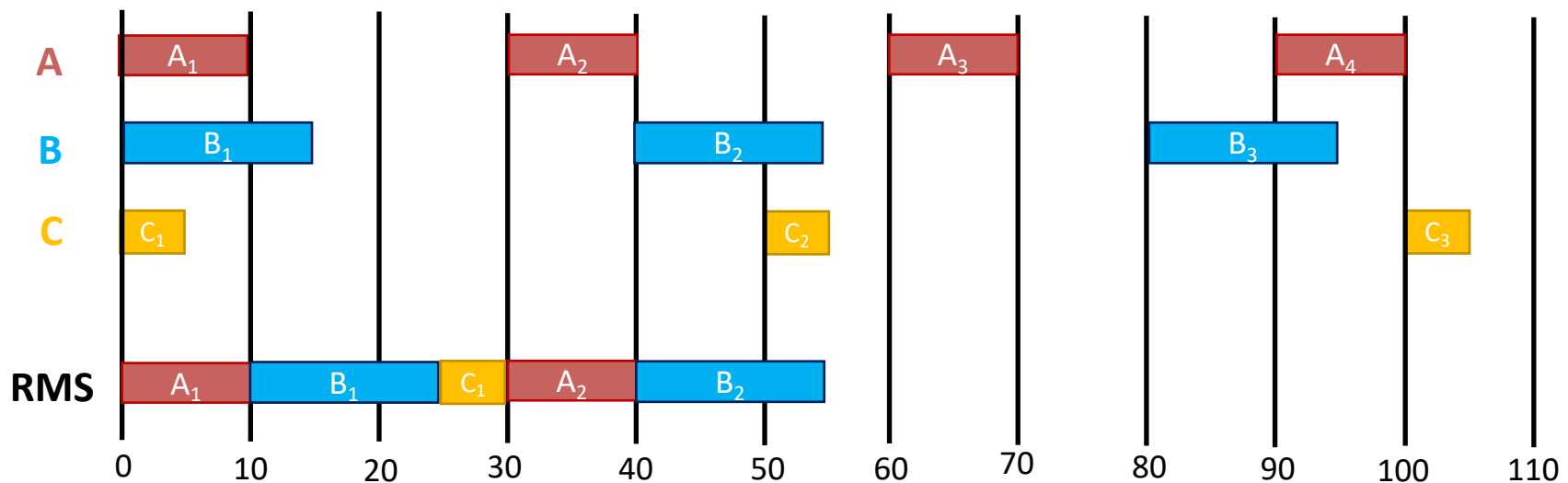
# L'ordonnement

## Ordonnement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



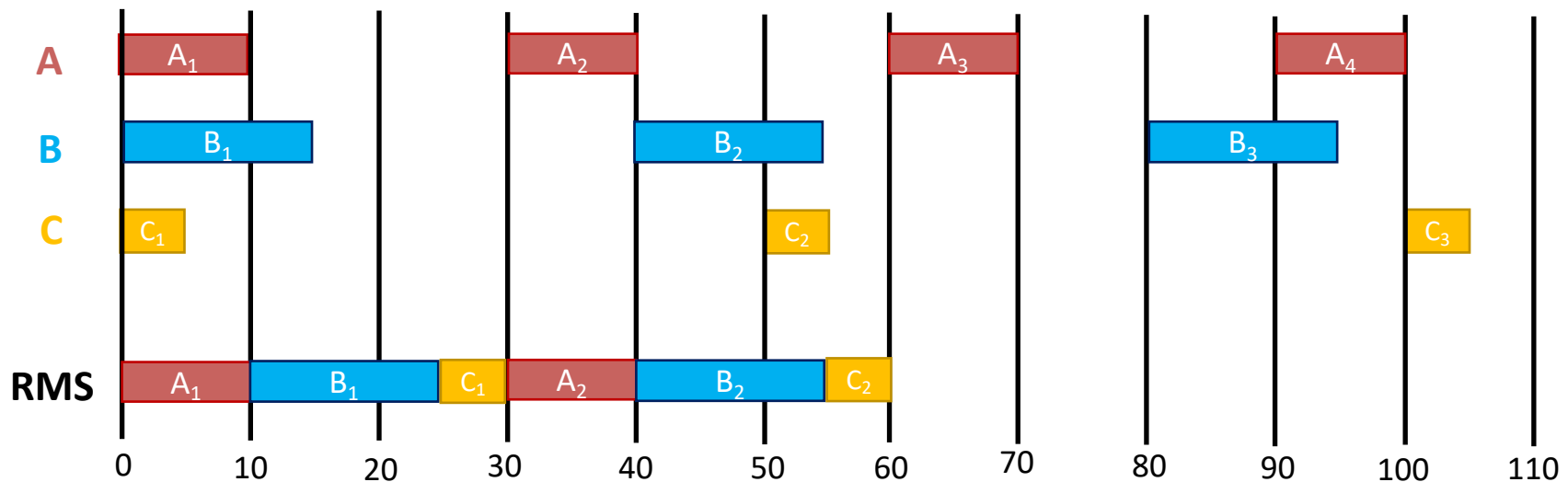
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
    - Chaque film a un framerate (image/s) différent (33, 25, 20)
    - Un processus est responsable de chaque film pour afficher chaque image
    - Le temps pour afficher une image varie selon les processus (10, 15, 5)



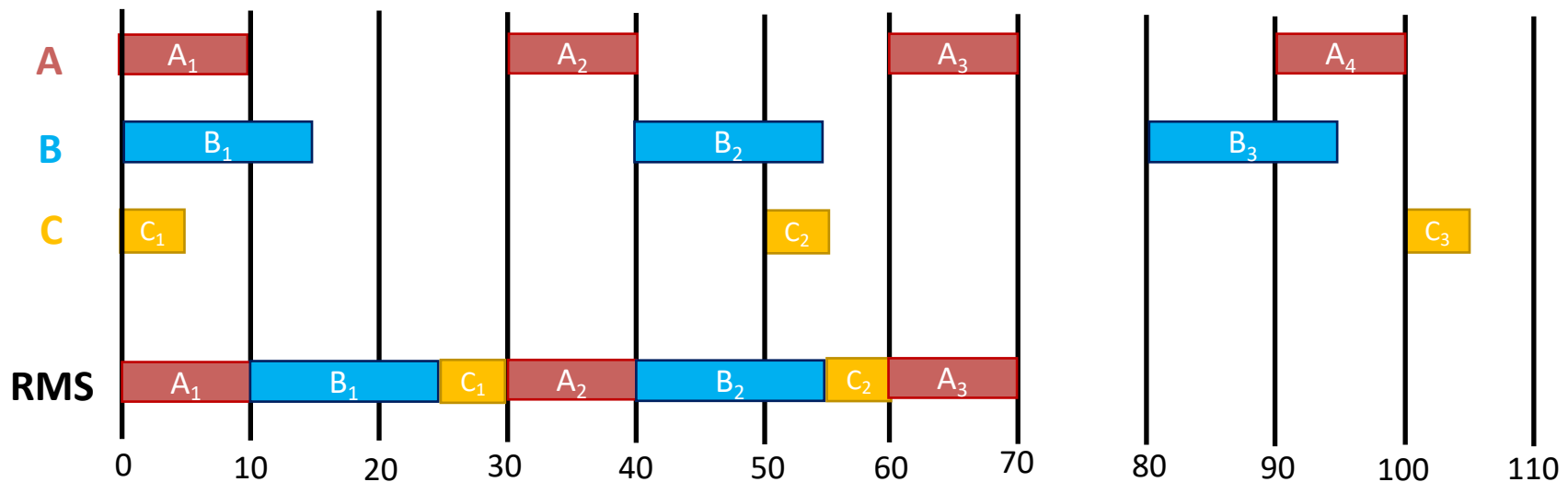
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)





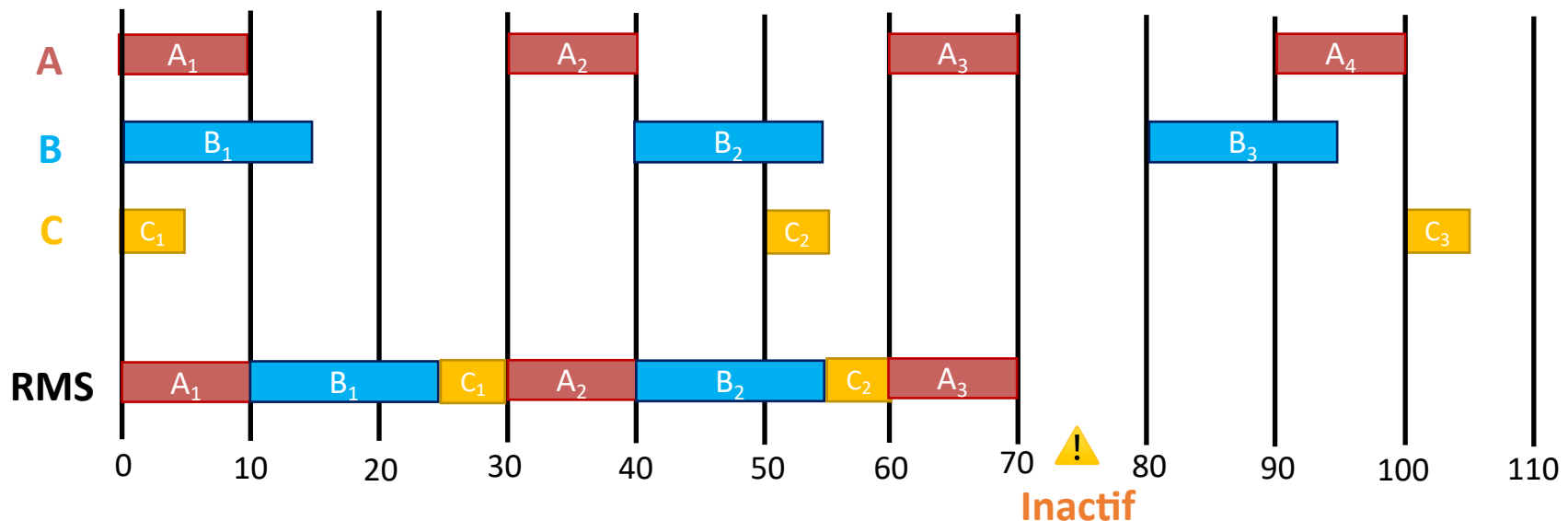
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



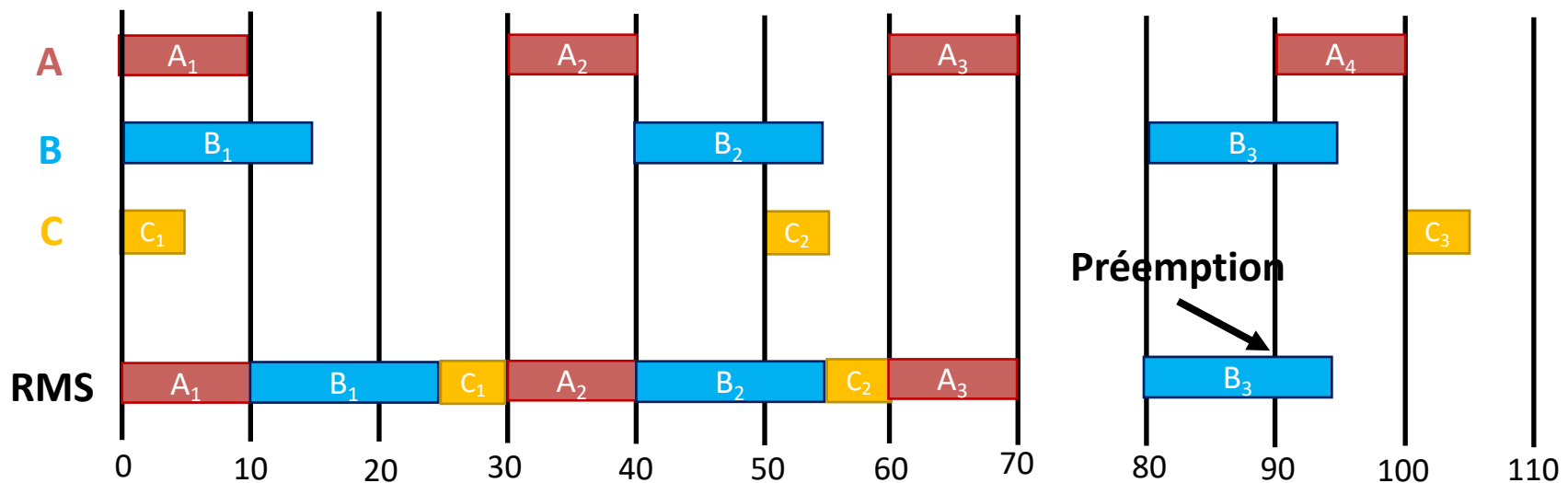
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



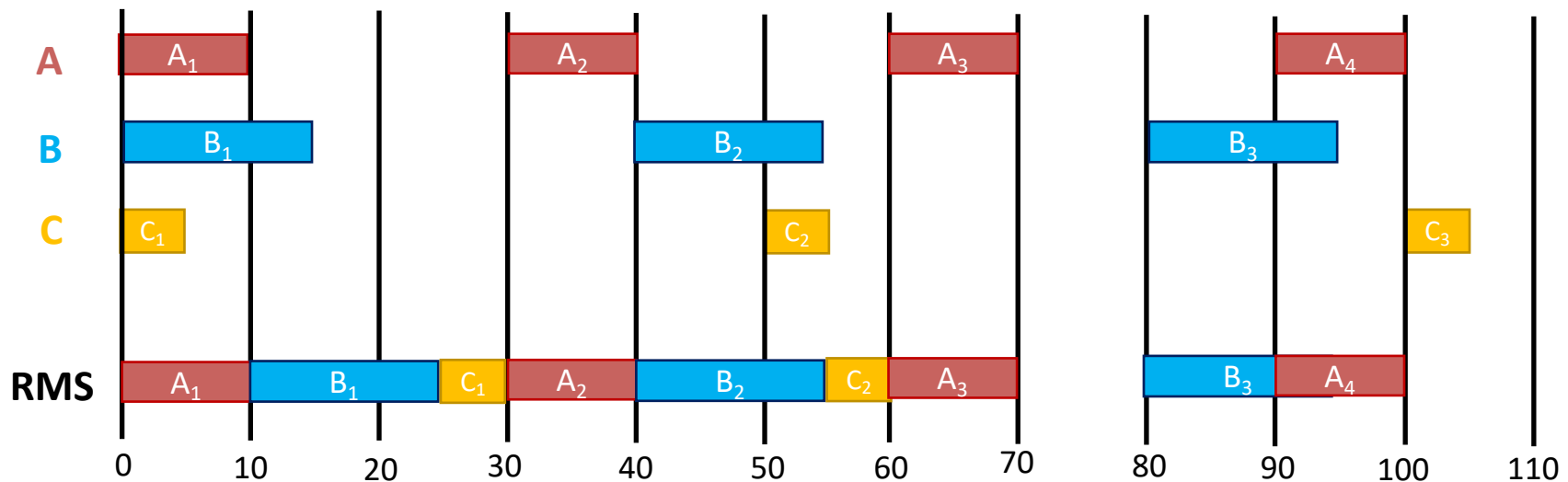
# L'ordonnancement

## Ordonnancement RMS

### ● Rate Monotonic Scheduling (RMS)

#### ● Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



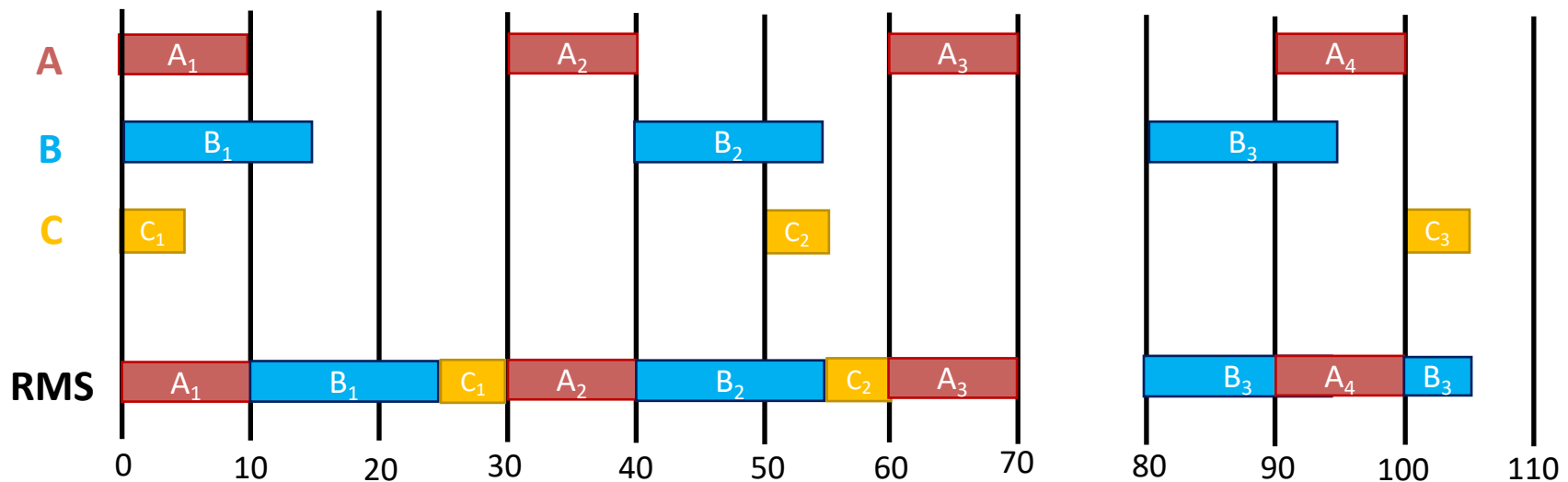
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
    - Chaque film a un framerate (image/s) différent (33, 25, 20)
    - Un processus est responsable de chaque film pour afficher chaque image
    - Le temps pour afficher une image varie selon les processus (10, 15, 5)



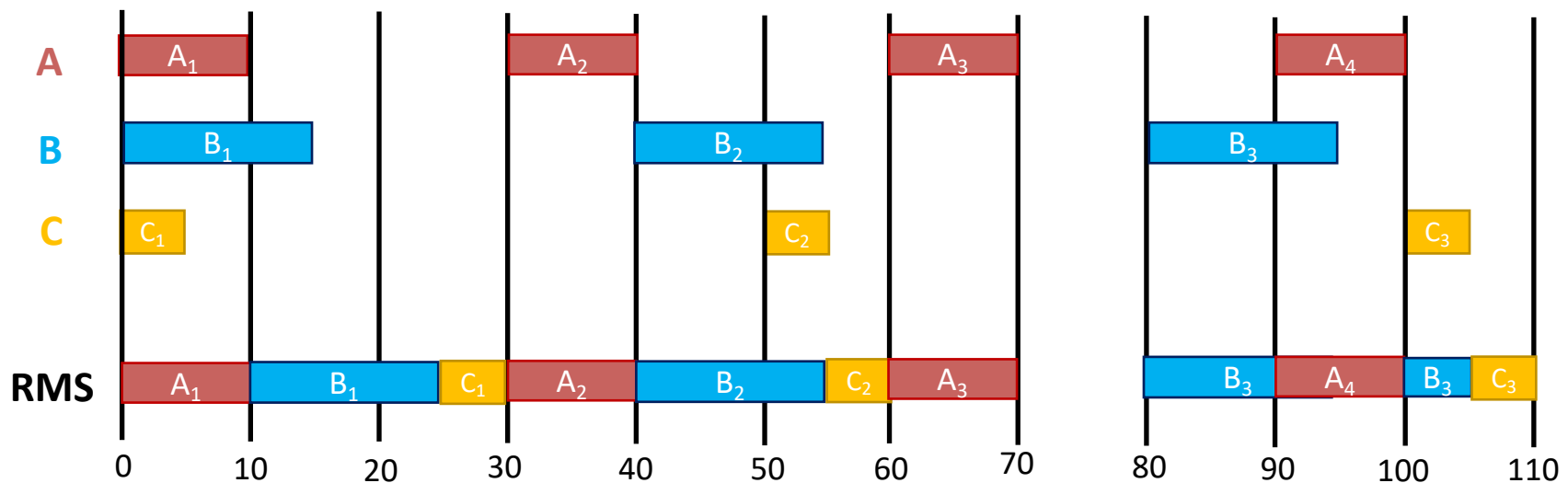
# L'ordonnancement

## Ordonnancement RMS

- Rate Monotonic Scheduling (RMS)

- Exemple : une plateforme de streaming (serveur vidéo)

- 3 utilisateurs veulent regarder un film au même instant
- Chaque film a un framerate (image/s) différent (33, 25, 20)
- Un processus est responsable de chaque film pour afficher chaque image
- Le temps pour afficher une image varie selon les processus (10, 15, 5)



## Evaluation des algorithmes d'ordonnancement

- Méthodologies d'évaluation

- Simulation déterministe : sur un scénario donné
  - Déroulement de l'algorithme : sur papier ou sur machine
- Modélisation stochastique
  - Théorie des files d'attente, chaîne de Markov...
- Instrumentation de système réel : benchmarking
  - Interférences de performances, choix de workload

- Nous : Simulation à la main

- Paramètres des tâches :
  - Durée d'exécution
  - Date d'arrivée

Tâche	Arrivée	Durée
T1	0	8
T2	1	4
T3	2	9
T4	3	5

## Evaluation des algorithmes d'ordonnancement

- Critères d'évaluation

- Taux d'utilisation du CPU

- Fraction du temps où le CPU est productif (occupé à exécuter)

- Débit

- Nombre de tâches terminées par unité de temps

- Non-privation

- Absence garantie de risque de famine (propriété qualitative)

- Temps de séjour

- Durée entre arrivée et terminaison de la tâche

- Temps d'attente

- Durée passée par la tâche dans la file d'attente

- Temps de réponse

- Durée entre arrivée et réponse de la tâche
    - « Réponse » à définir : affichage, calcul, etc...

# L'ordonnancement

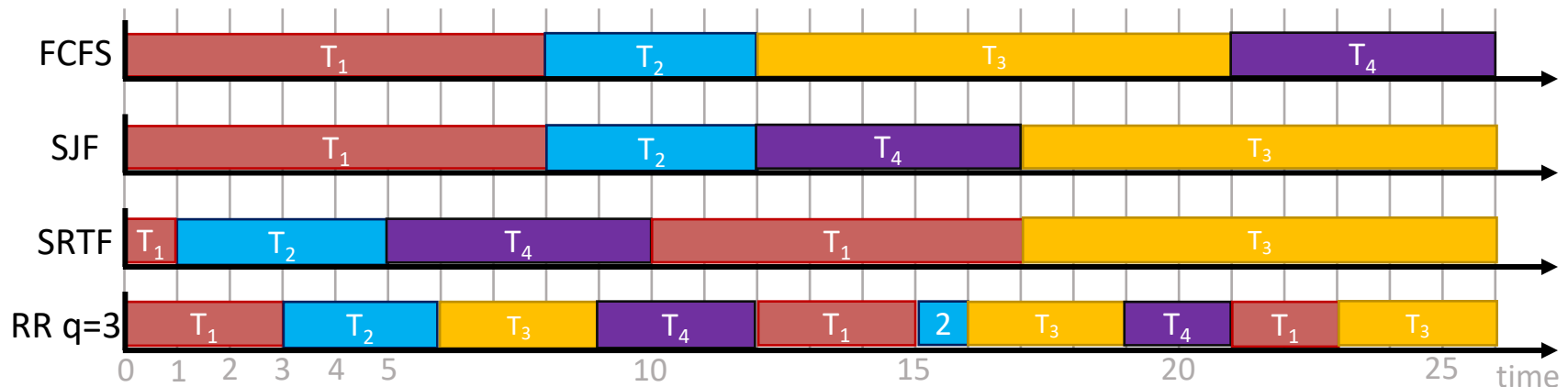
## Evaluation des algorithmes d'ordonnancement

### ● Critères d'évaluation

#### ● Exemple

- Soit le scénario suivant
- Les tâches sont ordonnancées selon plusieurs algorithmes

Tâche	Arrivée	Durée
$T_1$	0	8
$T_2$	1	4
$T_3$	2	9
$T_4$	3	5



#### ● Pour FCFS :

$$Temps_{sejour} = \frac{(8 - 0) + (12 - 1) + (21 - 2) + (26 - 3)}{4} = 15,25$$

$$Temps_{attente} = \frac{0 + (8 - 1) + (12 - 2) + (21 - 3)}{4} = 9$$



# L'ordonnancement

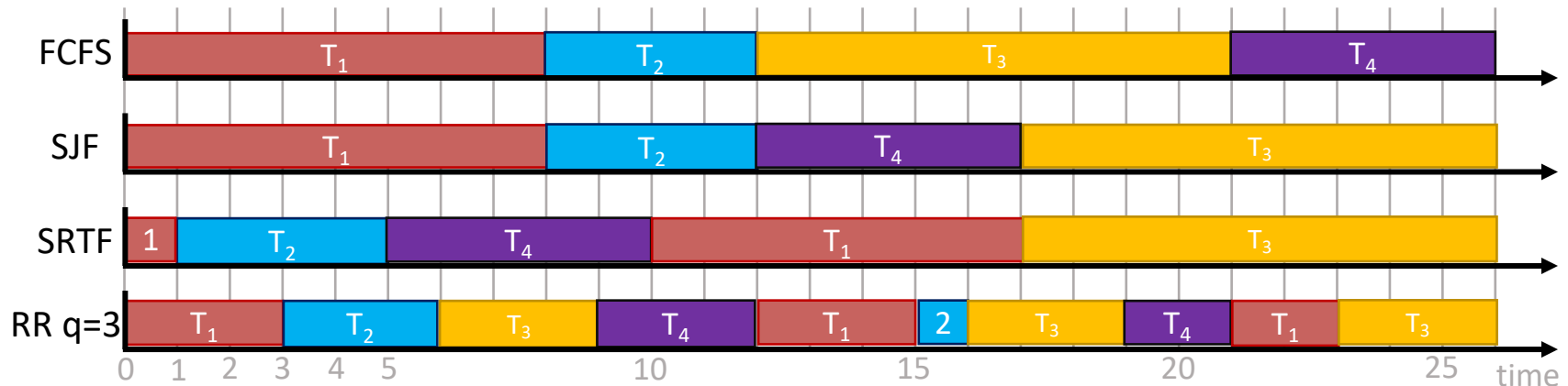
## Evaluation des algorithmes d'ordonnancement

### ● Critères d'évaluation

#### ● Exercice

- Calculer le temps de séjour et d'attente pour SRTF et RR

Tâche	Arrivée	Durée
$T_1$	0	8
$T_2$	1	4
$T_3$	2	9
$T_4$	3	5



- Pour SRTF :

$$Temps_{sejour} = \frac{(17 - 0) + (5 - 1) + (26 - 2) + (10 - 3)}{4} = 13$$

$$Temps_{attente} = \frac{(10 - 1) + 0 + (17 - 2) + (5 - 3)}{4} = 6,5$$

# L'ordonnancement

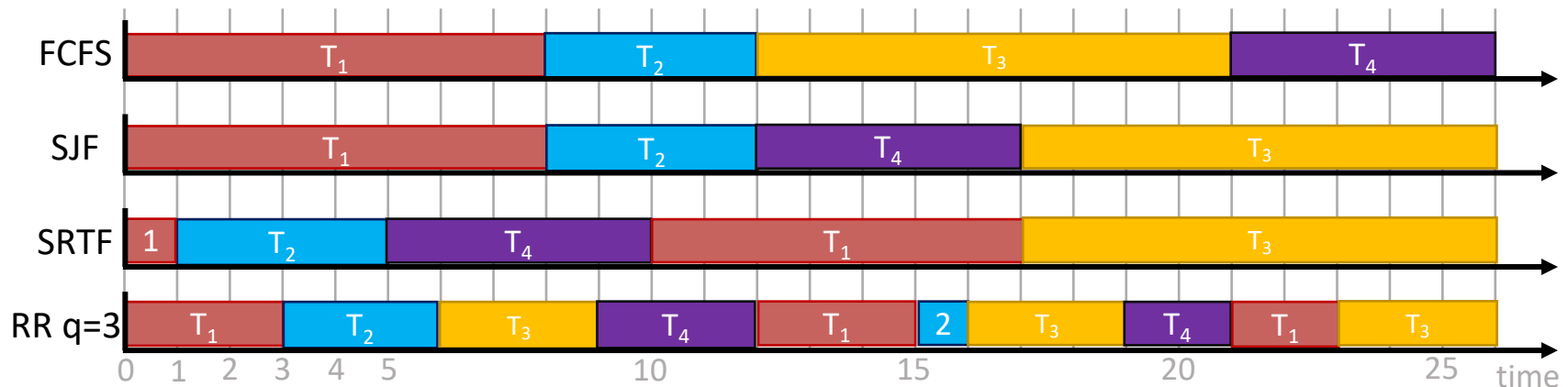
## Evaluation des algorithmes d'ordonnancement

### ● Critères d'évaluation

#### ● Exercice

- Calculer le temps de séjour et d'attente pour SRTF et RR

Tâche	Arrivée	Durée
$T_1$	0	8
$T_2$	1	4
$T_3$	2	9
$T_4$	3	5



- Pour RR :

$$Temps_{sejour} = \frac{(23 - 0) + (16 - 1) + (26 - 2) + (21 - 3)}{4} = 20$$

$Temps_{attente}$

$$= \frac{0 + (12 - 3) + (21 - 15) + (3 - 1) + (15 - 6) + (6 - 2) + (16 - 9) + (23 - 19) + (9 - 3) + (19 - 12)}{4} = 13,5$$

## Evaluation des algorithmes d'ordonnancement

- Analyse des résultats

	FCFS	SRTF	RR $q=3$
<b>Temps de séjour</b>	15,25	<b>13</b>	20
<b>Temps d'attente</b>	9	<b>6,5</b>	13,5

- Bien que RR offre les avantages d'équité et de non-privation, il est moins performant sur d'autres critères
- Choix en fonction des critères, des environnements
  - Objectifs à priori

# Bilan de cette séance

# Bilan de cette séance

## L'ordonnancement

- Il existe deux types d'algorithmes d'ordonnancement
  - Ordonnance coopératif (ou non préemptif)
  - Ordonnancement préemptif

- Nous avons vus plusieurs algorithmes

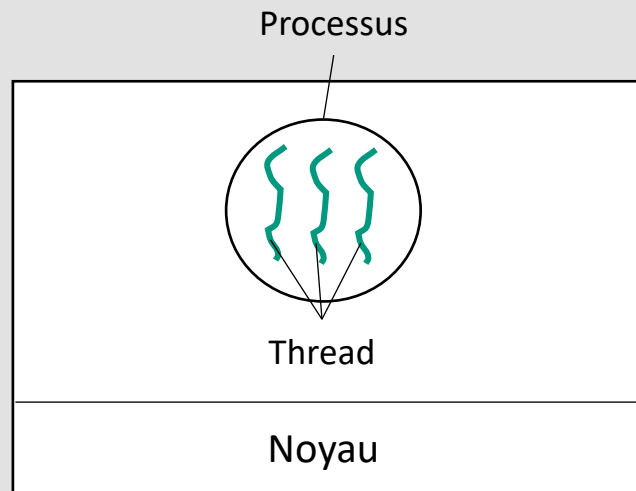
Premier arrivé, premier servi	Exécution du job le plus court en premier
Exécution du temps restant suivant le plus court	Exécution du ratio de réponse suivant le plus élevé
Ordonnancement de type tourniquet	Ordonnancement par priorité
Ordonnancement équitable	Ordonnancement RMS

- Nous avons vus comment les évaluer
  - Ex : Temps de séjour, temps d'attente

# La prochaine séance

## La prochaine séance

- Les threads
- La synchronisation



## Et maintenant ?

- **On passe aux exercices !**
  - Disponibles sur Moodle
- N'oubliez pas à l'issue de la séance
  - Quizz
  - Feedback

# Systemes d'exploitation

## ENSISA 1A

## Chapitre 2

### Processus