

Cours 3 : Ordonnancement de tâches

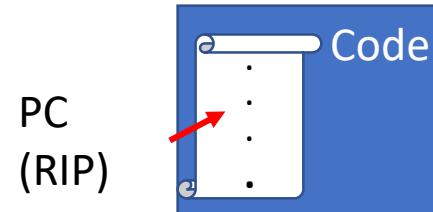
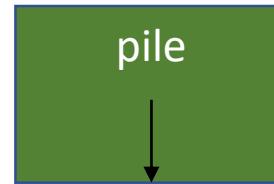
Introduction

- Hypothèse : système en multi-programmation \Rightarrow plusieurs tâches en concurrence pour l'accès au processeur
 - L'ordonnancement (scheduling) gère l'accès des tâches (processus) au processeur
- \Rightarrow Choisir **une tâche** parmi les tâches **prêtes**
- \Rightarrow Assurer que toutes les tâches aient l'accès au processeur : éviter la **famine**

Contexte d'une tâche

- 1 code et des variables = le **contexte mémoire**
3 zones : code
piles (variables locales)
données (variables globales + tas)

Tâche T (mémoire)



PCB :
Process
Control
Block

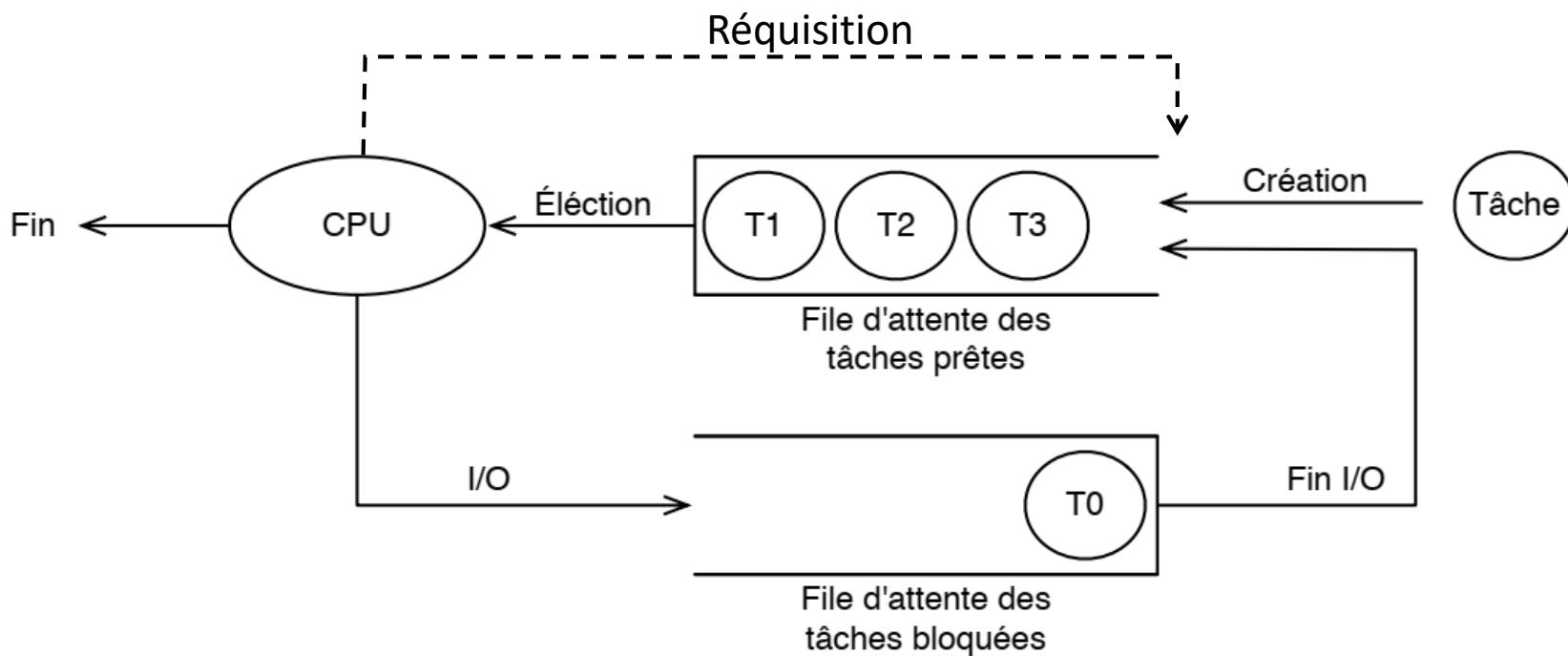
- des registres = le **contexte matériel**
PC (Program counter), registres piles, ...
- 1 état : Prêt / Bloqué / Elu
- 1 identifiant

PC
(RIP)

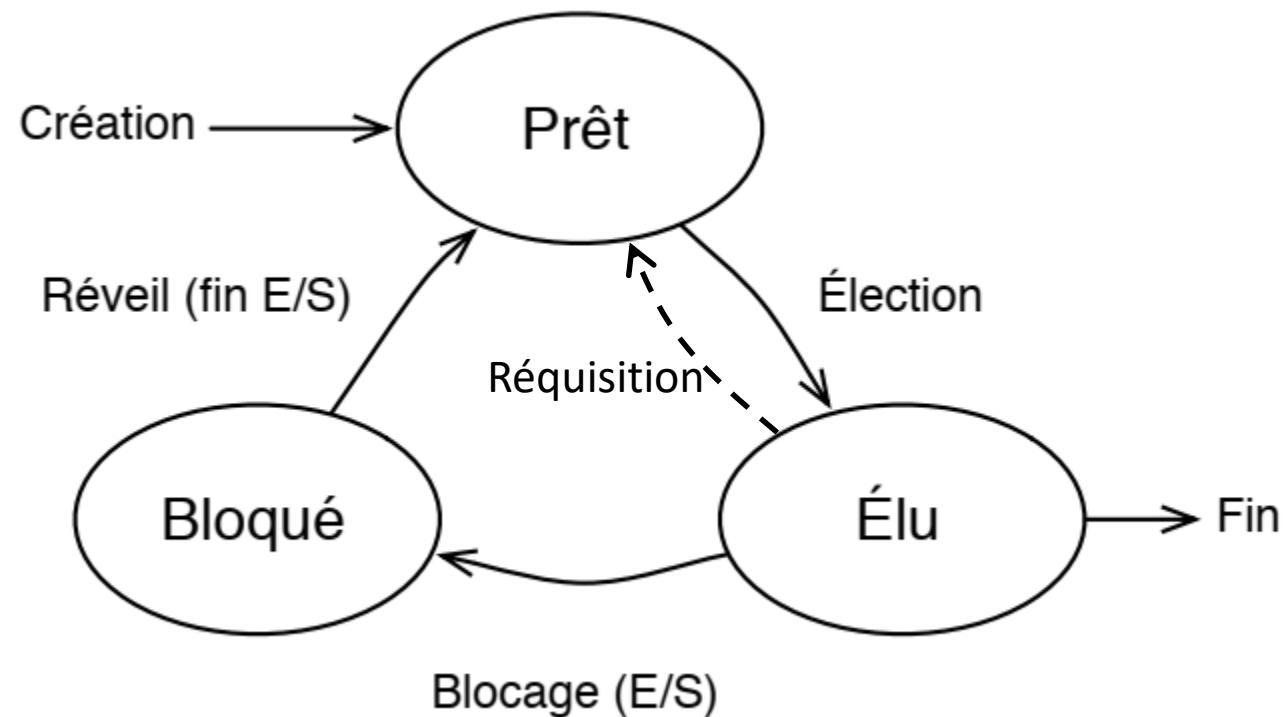
- Commutation T1 → T2 :
Sauvegarder contexte de T1 + Restaurer contexte de T2

Types d'ordonnancement - Batch

- Pas de temps partagé
- La tâche élue conserve le processeur jusqu'à sa fin, son blocage ou sa réquisition éventuelle

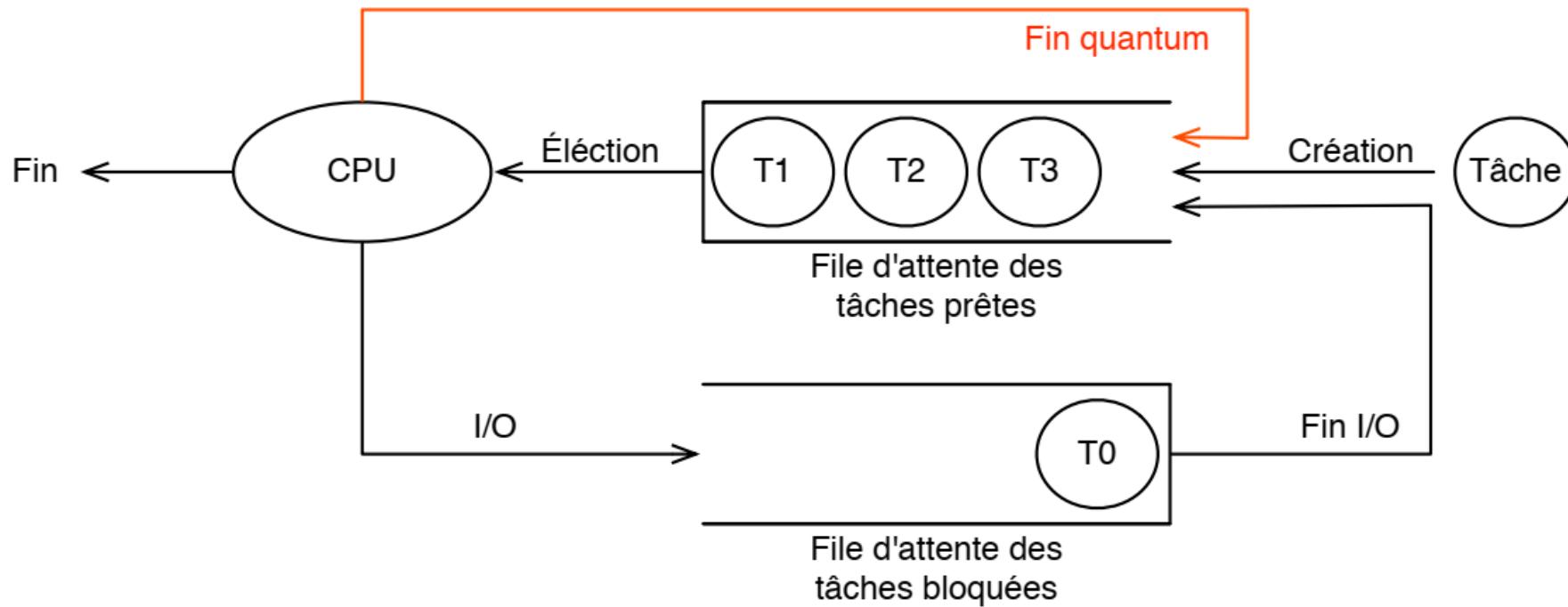


Batch – Graphe d'états

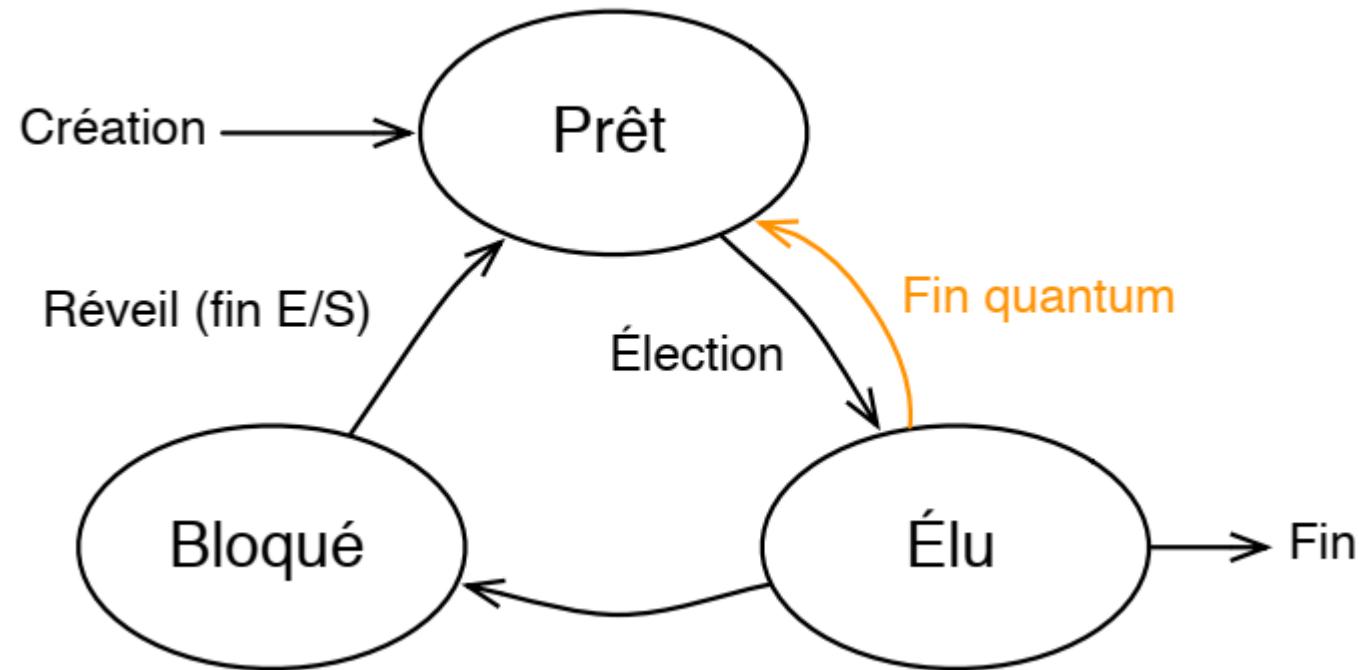


Types d'ordonnancement – Temps partagé

- Un quantum = temps maximal d'exécution continue d'une tâche sur le processeur

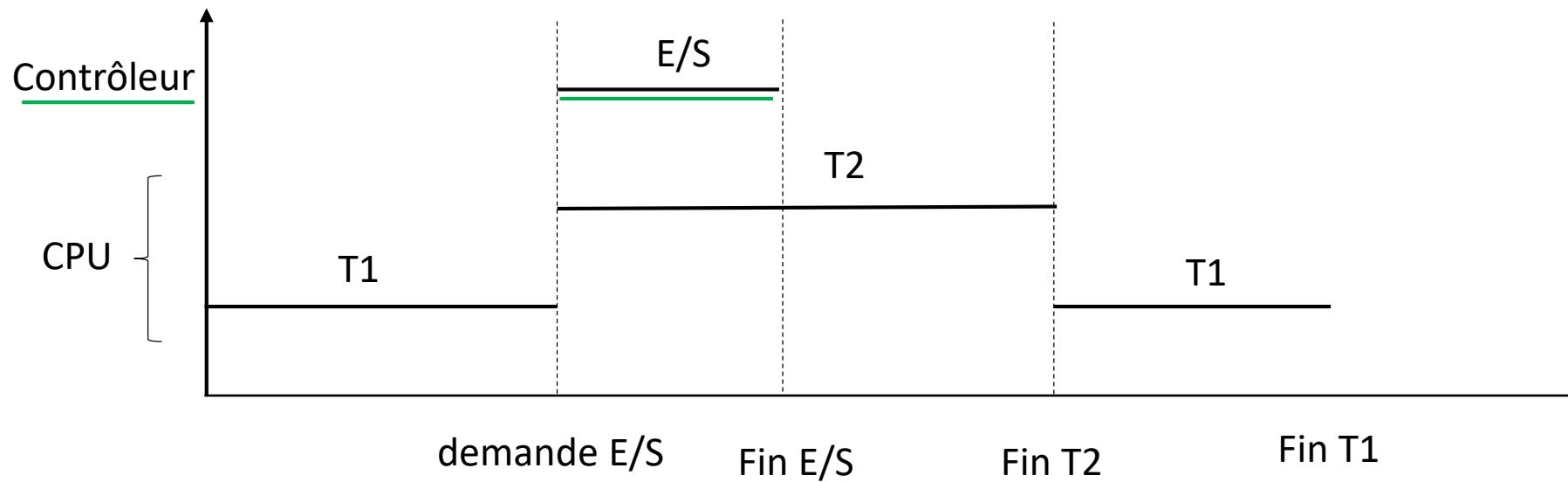


Temps partagé – Graphhe d'états



Réquisition

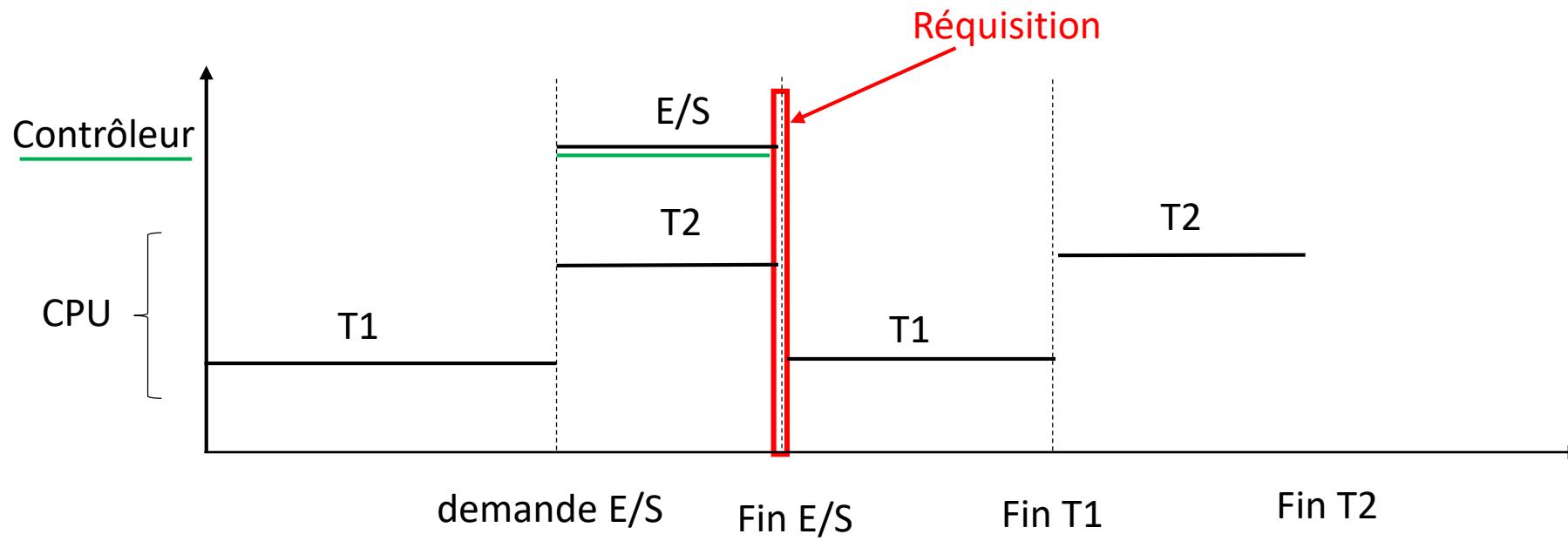
- Réquisition (ou Préemption) : Possibilité pour une **tâche Prête plus prioritaire** d'interrompre le processus élu pour lui réquisitionner (préempter) le processeur
- Exemple : 2 tâches T1 et T2 en mode batch avec priorité T1 > priorité T2



Sans réquisition

Réquisition

- Réquisition (ou Préemption) : Possibilité pour une **tâche Prête plus prioritaire** d'interrompre le processus élu pour lui réquisitionner (préempter) le processeur
- Exemple : 2 tâches T1 et T2 en mode batch avec priorité T1 > priorité T2



Avec réquisition

Algorithmes d'ordonnancement : les métriques

- **Temps de réponse** d'une tâche T_i (temps perçu par l'utilisateur)

$$Tr_i = \text{date de fin de } T_i - \text{date de création de } T_i$$

- **Temps d'attente** de T_i (temps où T_i n'a pas été élue)

$$Ta_i = Tr_i - \text{durée CPU } T_i$$

- Taux d'occupation du processeur (CPU)

$$\tau_{CPU} = \frac{\sum \text{durée CPU } Ti}{\text{Temps total}}$$

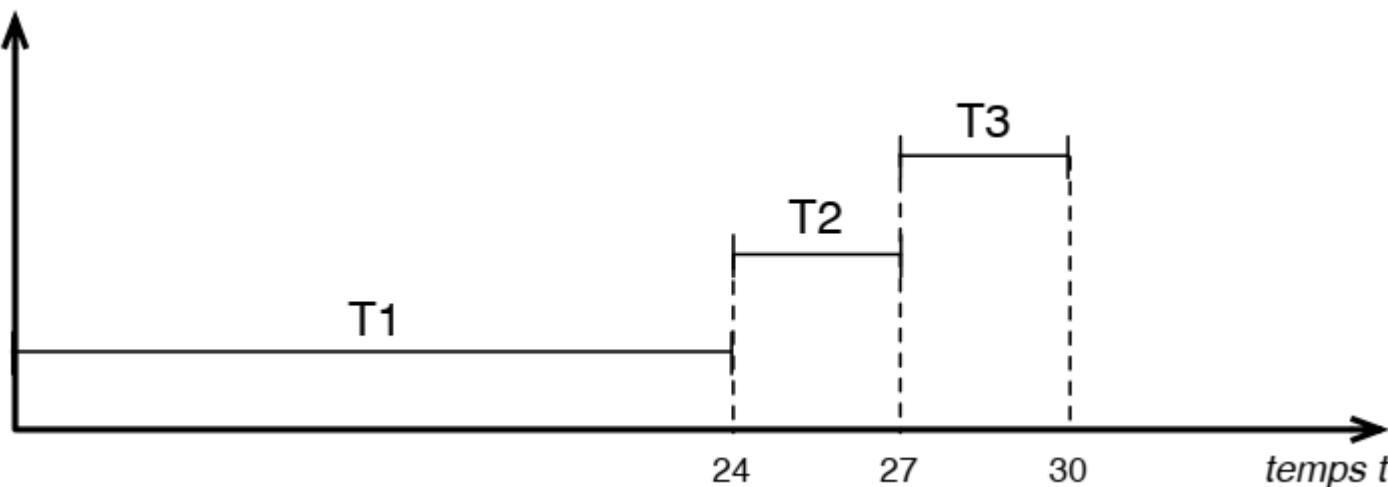
- Taux d'occupation de l'unité d'échange

$$\tau_{UE} = \frac{\sum \text{durée E/S } Ti}{\text{Temps total}}$$

1. Ordonnancements Batch

Batch : FCFS

- Premier arrivé, premier servi : **FIFO** (First In, First Out) ou **FCFS** (First Come, First Served)
- Les tâches sont traitées dans leur **ordre d'arrivée**.
- *Exemple* : 3 tâches T1, T2, T3 créées à $t=0$
T1 dure 24 secondes, T2 dure 3 secondes, T3 dure 3 secondes (pas d'E/S)



Batch : FCFS

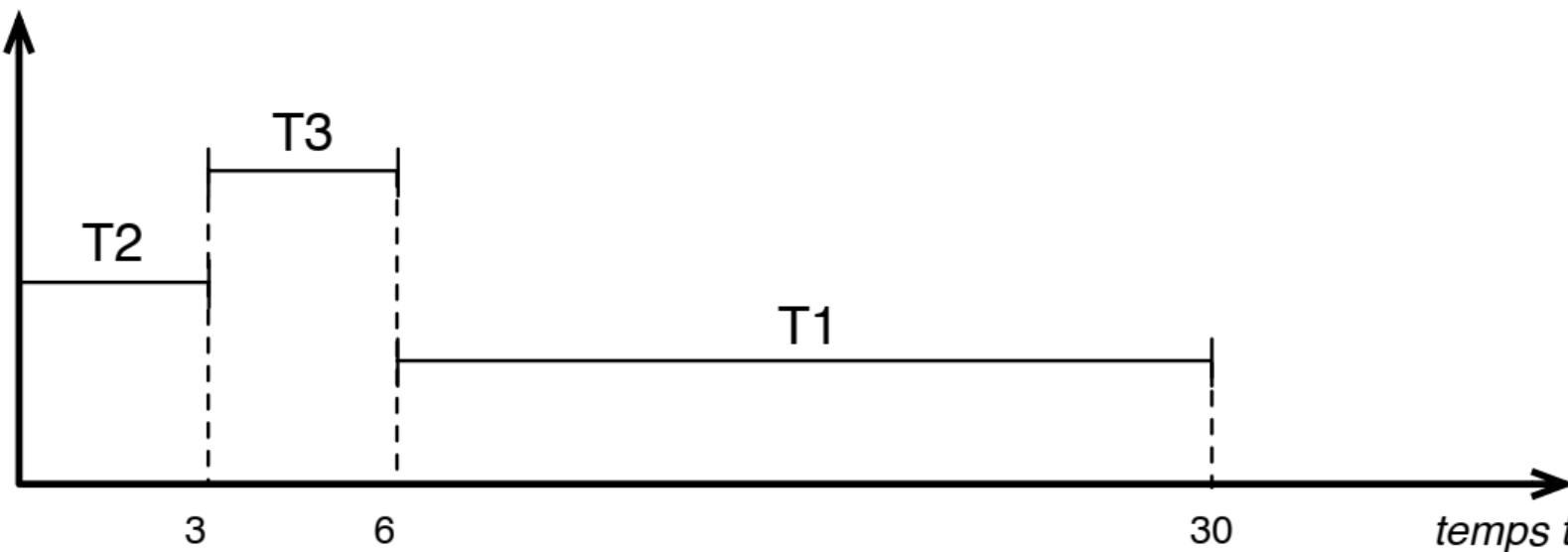
Temps de réponse	Temps d'attente
$Tr_1 = 24 - 0 = 24s$	$Ta_1 = 24 - 24 = 0s$
$Tr_2 = 27 - 0 = 27s$	$Ta_2 = 27 - 3 = 24s$
$Tr_3 = 30 - 0 = 30s$	$Ta_3 = 30 - 3 = 27s$

$$\text{Temps attente moyen} = \frac{0+24+27}{3} = 17 \text{ secondes}$$

Les tâches courtes (eg. T2 et T3) sont pénalisées

Batch : SJF

- *Plus courts travaux d'abord* : **SJF** (Shortest Job First) ou **SJN** (Shortest Job Next)
- Choisir la tâche prête ayant **le temps d'exécution restant le plus court**
- **Hypothèse** : le temps d'exécution des tâches est connu à l'avance



Batch : SJF

Temps de réponse	Temps d'attente
$Tr_1 = 30 - 0 = 30s$	$Ta_1 = 30 - 24 = 6s$
$Tr_2 = 3 - 0 = 3s$	$Ta_2 = 3 - 3 = 0s$
$Tr_3 = 6 - 0 = 6s$	$Ta_3 = 6 - 3 = 3s$

$$\text{Temps attente moyen} = \frac{6+0+3}{3} = 3 \text{ secondes}$$

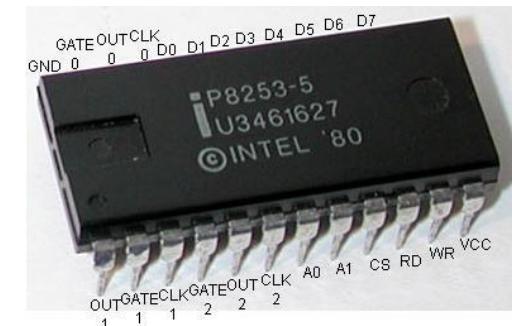
- ⊕ Stratégie SJF meilleure que FCFS
- ⊖ Temps d'exécution rarement connus à l'avance
- ⊖ Risque de famine des tâches longues

2. Ordonnancements à temps partagé

Temps partagé : Implémentation

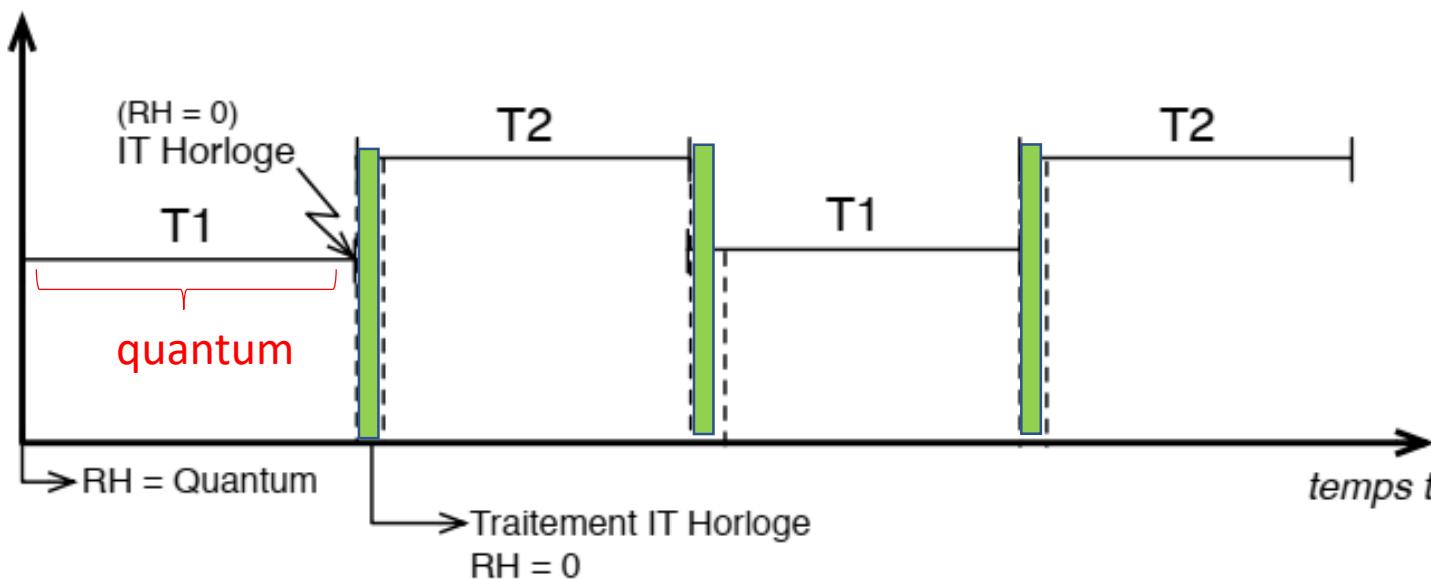
- Implémentation du quantum grâce l'**interruption horloge** générée par le composant Horloge (*PIT : Programmable Interval Timer*)
- Horloge
 - 1 composant matériel externe au CPU (~contrôleur)
 - Fonctionne à une certaine fréquence (~1 MHz)
 - Gère le Registre Horloge (RH ou Rtempo) :

Toutes les μ s (10⁻⁶s)
Décrémenter RH
Si RH = 0
Lever IT Horloge



Temps partagé : Implémentation

- Le RH est un registre privilégié : peut être réinitialiser uniquement en mode Système
- Le système met une valeur dans RH pour générer une interruption dans $N \mu s$.



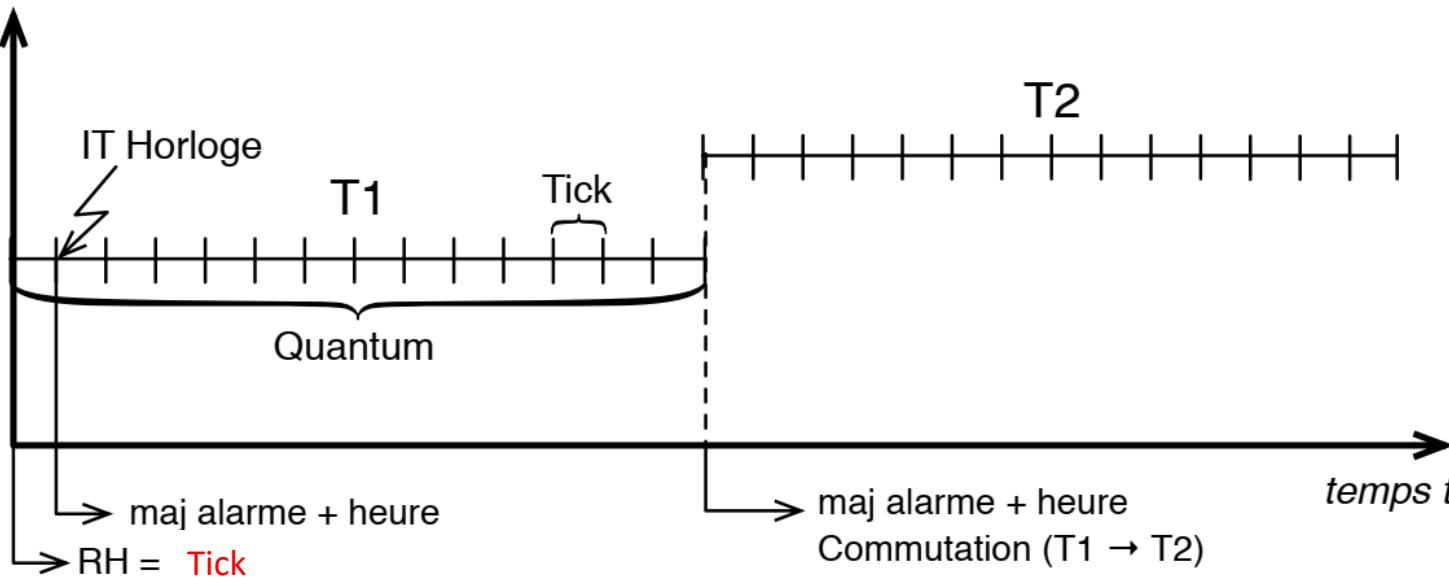
Temps partagé : Implémentation

- Traitement de l'interruption Horloge dans le système (en Mode S)
 1. Sauvegarder les registres du processus élu
 2. Traiter les alarmes
 3. Mettre à jour l'heure
 4. Élire une nouvelle tâche T
 5. Restaurer les registres de T
 6. Réinitialiser le registre RH

Temps partagé : quantum et tick

- Dans la plupart des systèmes, quantum ≈ 100 ms (suffisant pour percevoir le pseudo parallélisme) \Rightarrow gestion des alarmes peu précise
- \Rightarrow Utilisation de **Ticks** = la valeur (temps) entre deux interruptions horloge

$$\text{Quantum} = N \times \text{Tick}$$



Temps partagé : Algorithme RR

- Algorithme du Tourniquet : **RR** – Round Robin
- Accès des tâches à *tour de rôle* au processeur.

Création d'une tâche : Insertion en **queue** de la File des Tâches

Election : Choisir la première tâche prête en **tête** de la File

Fin de quantum (N ticks) : Réinsertion de la tâche élue en **queue** à l'état Prêt

Temps partagé : RR

- Exemple : Quantum = 40 ms, temps de commutation négligeable

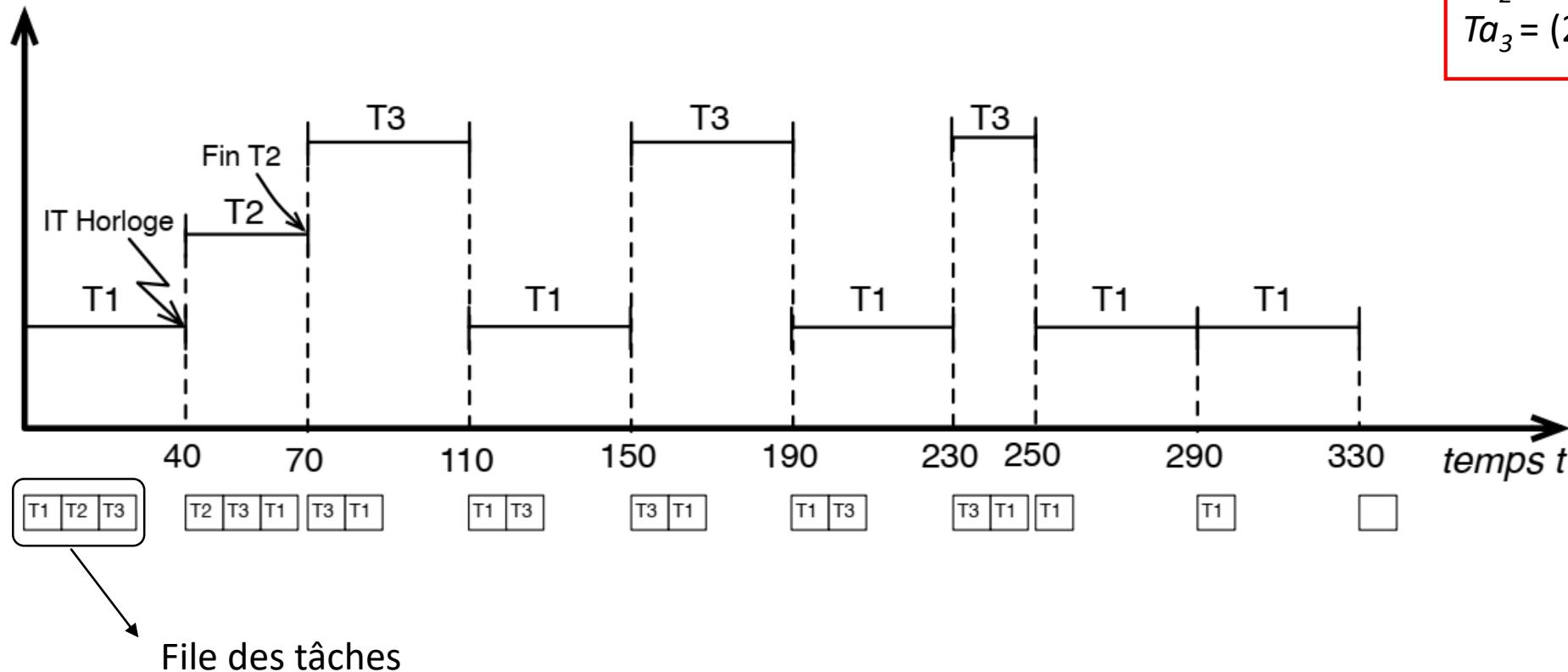
T1, T2, T3 créées dans cet ordre à $t = 0$:

T1 dure 200 ms, T2 dure 30 ms, T3 dure 100 ms

$$Ta_1 = (330 - 0) - 200 = 130 \text{ ms}$$

$$Ta_2 = (70 - 0) - 30 = 40 \text{ ms}$$

$$Ta_3 = (250 - 0) - 100 = 150 \text{ ms}$$



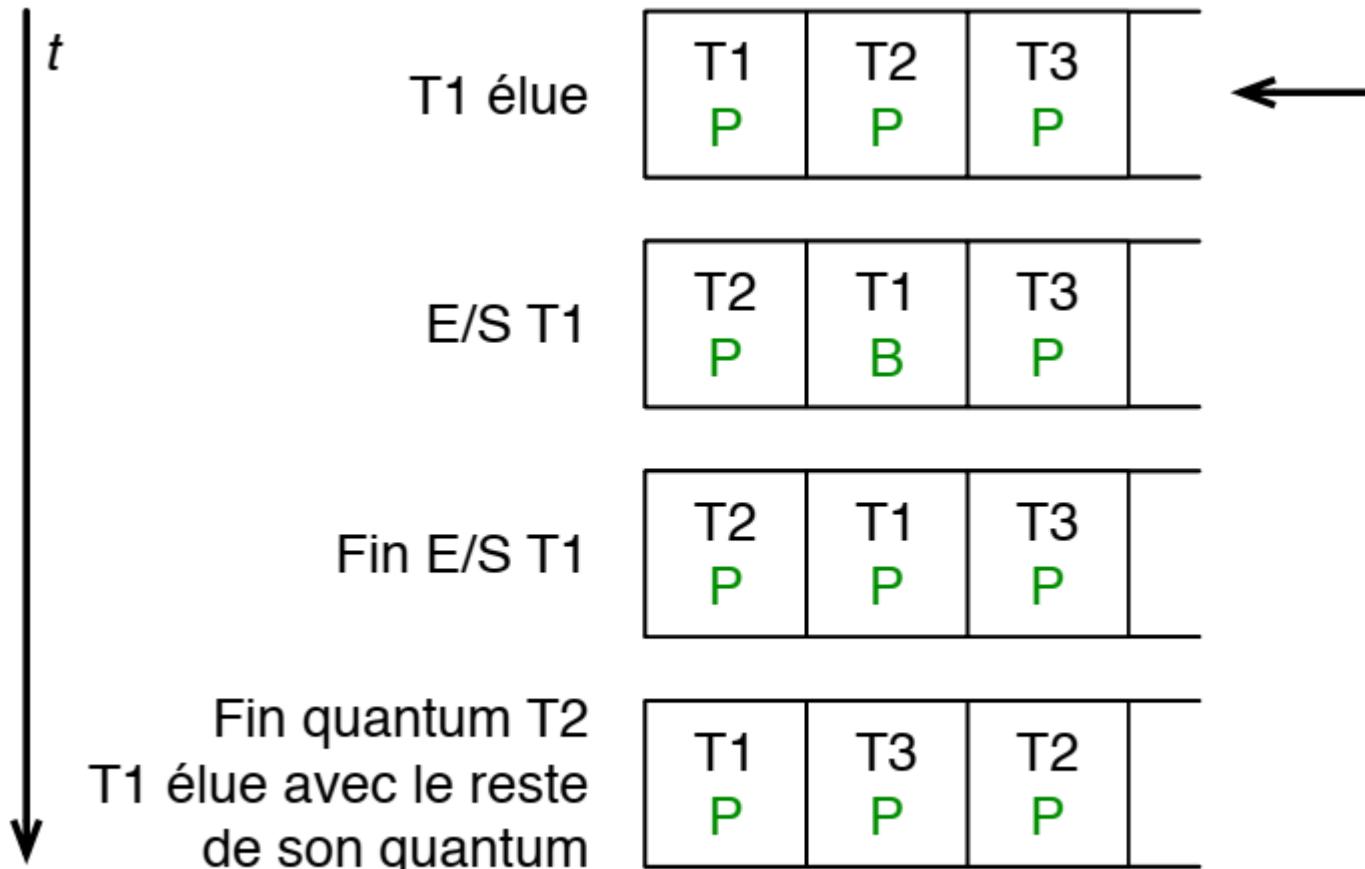
Temps partagé : RR – Entrées/Sorties

- Tâche élue T fait une E/S $\Rightarrow T$ passe à l'état "Bloqué"
- Plusieurs stratégies :
 1. Tourniquet de base : T est insérée dans **une liste de tâches bloquées**, en fin d'E/S, T est réinsérée **en queue de la liste des tâches prêtes**.
 2. Variante : 1 seule file, T **reste dans la file dans sa position** (en tête). Au réveil, **le quantum restant** de T lui est alloué.

La variante évite de trop pénaliser les tâches bloquées

Temps partagé : RR – Entrées/Sorties

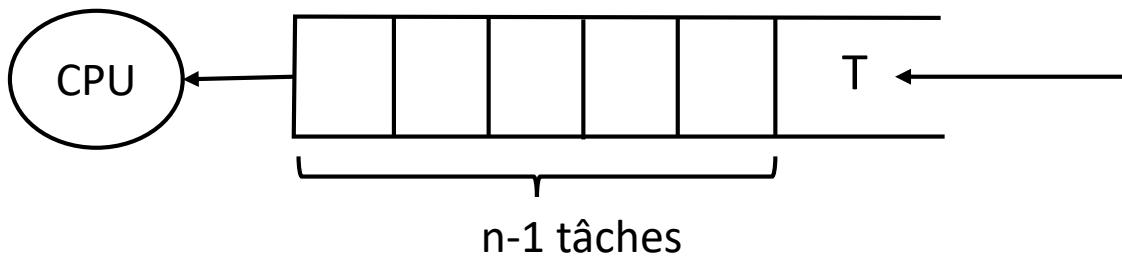
- Exemple : 1 file, P: Etat prêt, B : état bloqué



Temps partagé RR - Conclusions

⊕ Équitable, simple

⊕ Pas de famine : si une tâche T est prête, soit n le nombre de tâches et q la valeur du quantum alors T attend au plus $(n-1) \times q$ avant d'être élue



⊖ "Trop" équitable : pas de priorités entre les tâches

Temps partagé : priorité fixe

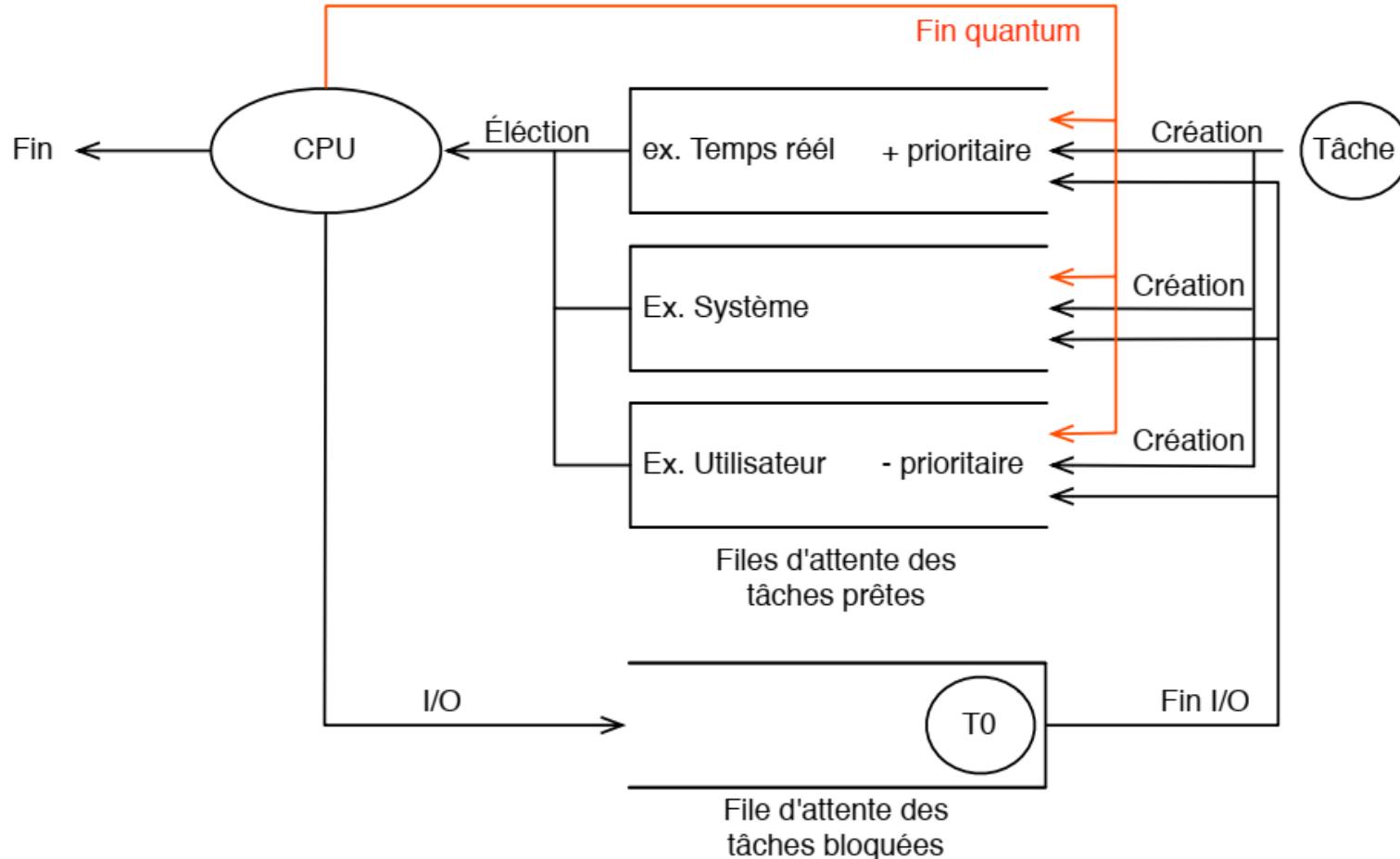
- Chaque tâche est associée à une priorité fixe tout au long de son exécution
- N niveaux de priorités et **une file par niveau**

Création d'une tâche : Attribuer un niveau à la tâche, insertion en queue de la file correspondante

Election : Choisir la tâche Prête la plus en tête dans la file non vide la plus prioritaire

Fin de quantum : Réinsertion de la tâche élue en queue de sa file

Temps partagé : priorité fixe



- + Prend en compte les tâches prioritaires
- Pas équitable : risque de famine des tâches les moins prioritaires

Temps partagé : priorité dynamique

- Les priorités affectées aux tâches changent en cours d'exécution
- N files de F_0 à F_{N-1} (F_0 la plus prioritaire)

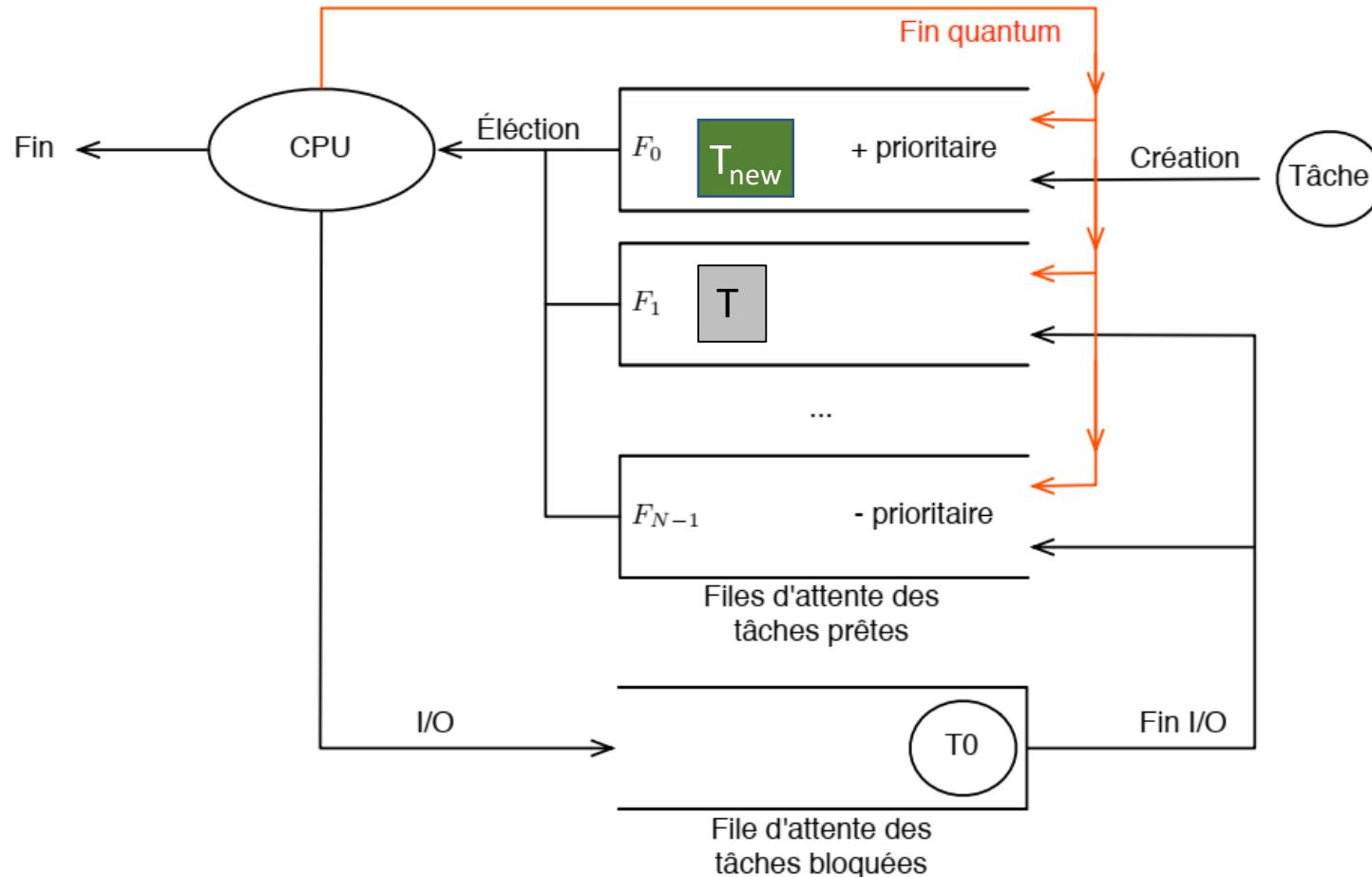
Création d'une tâche : Insertion de la tâche en queue de la file F_0

Election : Choisir la tâche Prête la plus en tête dans la file non vide la plus prioritaire

Fin de quantum : Réinsertion de la tâche élue en queue de la file suivante

Les tâches issues de la file F_{N-1} sont réinsérées en queue de la file F_0

Temps partagé : priorité dynamique



Temps partagé : priorité dynamique

Plus une tâche est élue, moins elle est prioritaire

⇒ une tâche ancienne est donc moins prioritaire

⇒ Approximation de SJF (*Shortest Job First*) sans connaître à l'avance la durée des tâches

Risque de famine des tâches longues si flot continu de création de tâches.

Résoudre la famine : régulièrement (tous les X quantum),
le système remonte la priorité des toutes les tâches prêtes.

Ordonnancement multiples

- Utilisation de plusieurs stratégies dans les systèmes récents

