



# Ordonnancement dynamique

---

## EDF : Early Deadline First

 Échéance la plus proche d'abord

## LLF : Least Laxity First

 Marge la plus courte d'abord

 Marge = échéance - durée restante de traitement-temps courant

# Exemple



Donner le diagramme par ordonnancement avec EDF des tâches :

	p	t	e
P1	60	20	60
P2	80	20	80
P3	120	30	120

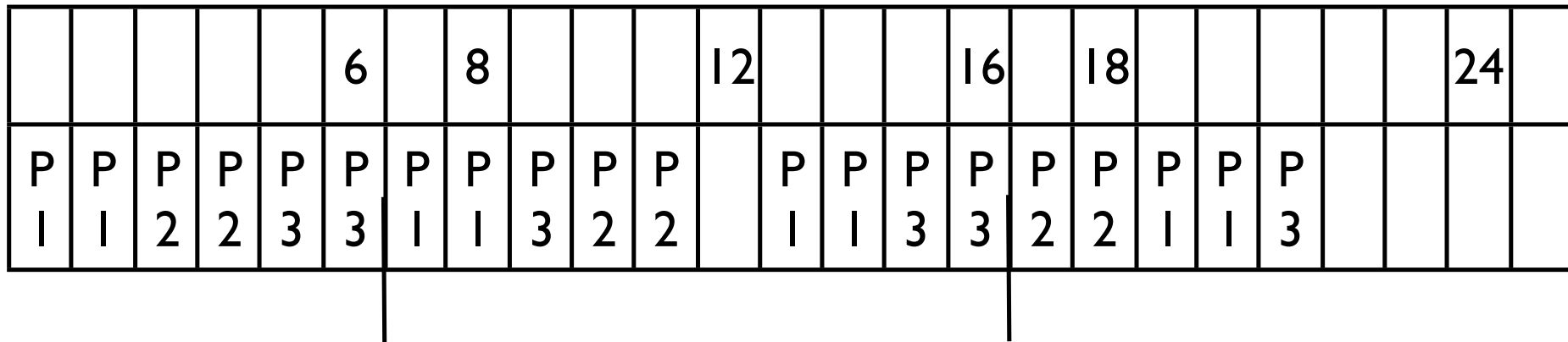
					6		8				12				16		18						24	
P	P	P	P	P	P	P	P	P	P	P		P	P	P	P	P	P	P	P	P				
1	1	2	2	3	3	3	1	1	2	2		1	1	3	3	3	2	2	1	1				

# Exemple



Donner le diagramme par ordonnancement avec LLF des tâches :

	p	t	e
P1	60	20	60
P2	80	20	80
P3	120	30	120









# A propos de EDF et LLF

---

-  Si on prend des conditions identiques à monotone à taux la condition de faisabilité est à 100% du temps CPU.
-  Néanmoins les algorithmes ne sont pas stables : en cas de surcharge du processeur des tâches peuvent ne pas respecter leurs échéances.

# Exercice I

---


- 2 tâches T1 et T2
- Ordonnancement par taux monotone
- Ordonnancement par EDF (early dead line first) : échéance la plus proche d'abord
- Représenter le diagramme temps.

	p	e
T1	2	4
T2	5	10




# Solution exercice I

---

  $2/4 + 5/10 = 1$

  $t_o = c_1 + c_2 = 7$

  $W_2(t_o) = 2*2 + 5 = 9$

  $W_2(t_l) = 3*2 + 5 = 11 <> t_l$

	p	c
T1	2	4
T2	5	10

# Tâches non périodiques

---

- Tâches indépendantes, si temps exécution est connu => LLF.
- Tâches indépendantes, si temps exécution est non connu => EDF (meilleur choix mais n'apporte pas la garantie).

# Algorithmes mixtes

---

- Tâches + Paramètres  $\Rightarrow$  Ordonnement statique
- Pendant l'exécution  $\Rightarrow$  tenir compte des paramètres (urgence, criticité,...)

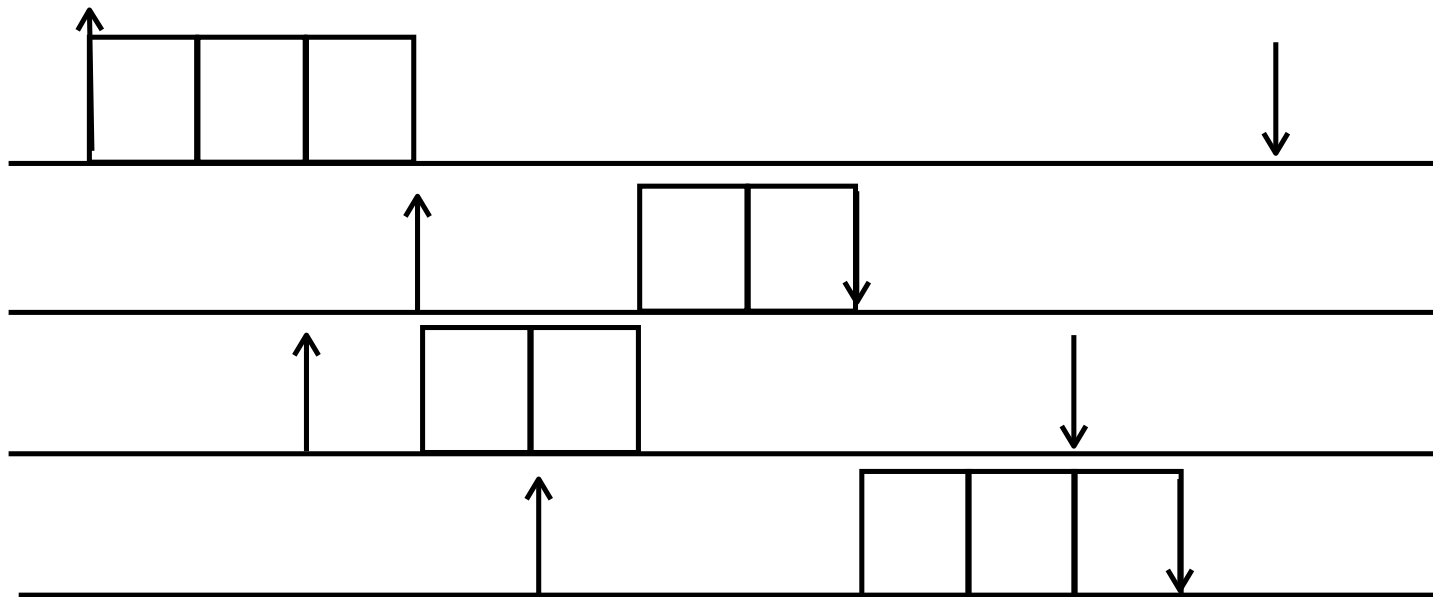
# Optimisation de EDF

---

- EDF est optimal (i.e : tt ensemble de tâches faisable pourra être ordonnancé)
- Algorithme idée :
  - on part d'un ordonnancement
  - par itération on déplace des unités de traitements pour tendre vers un ordonnancement EDF

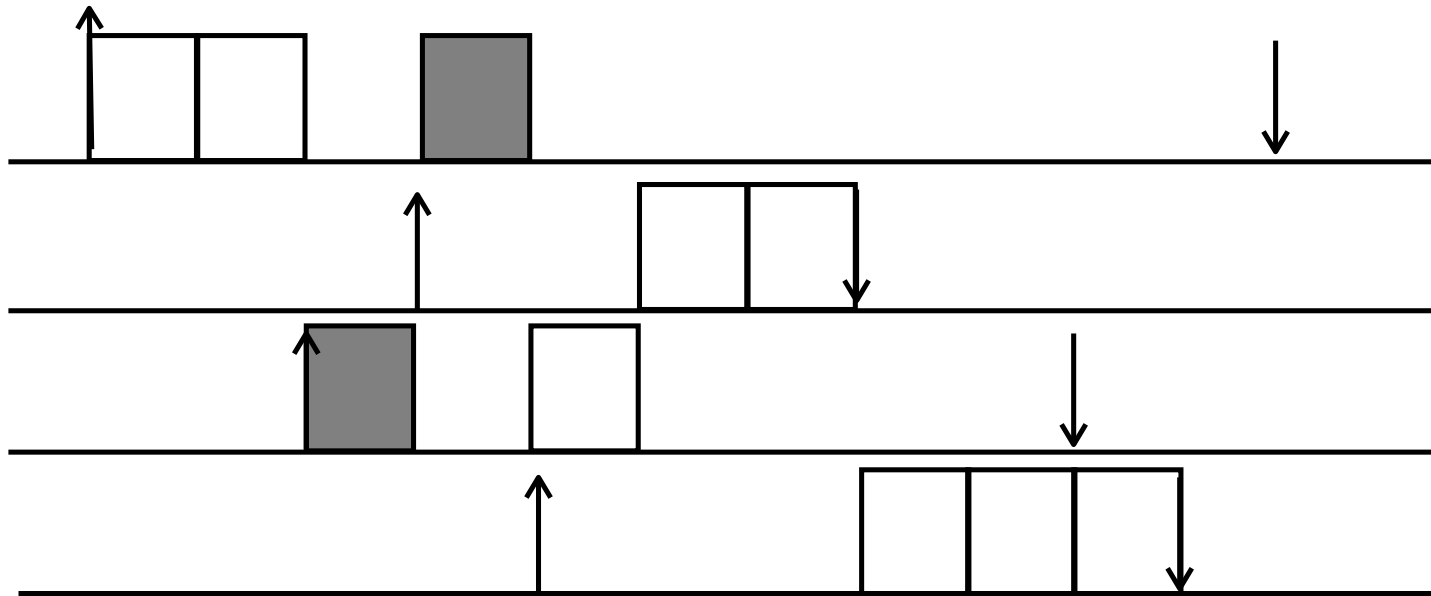
# Départ

---



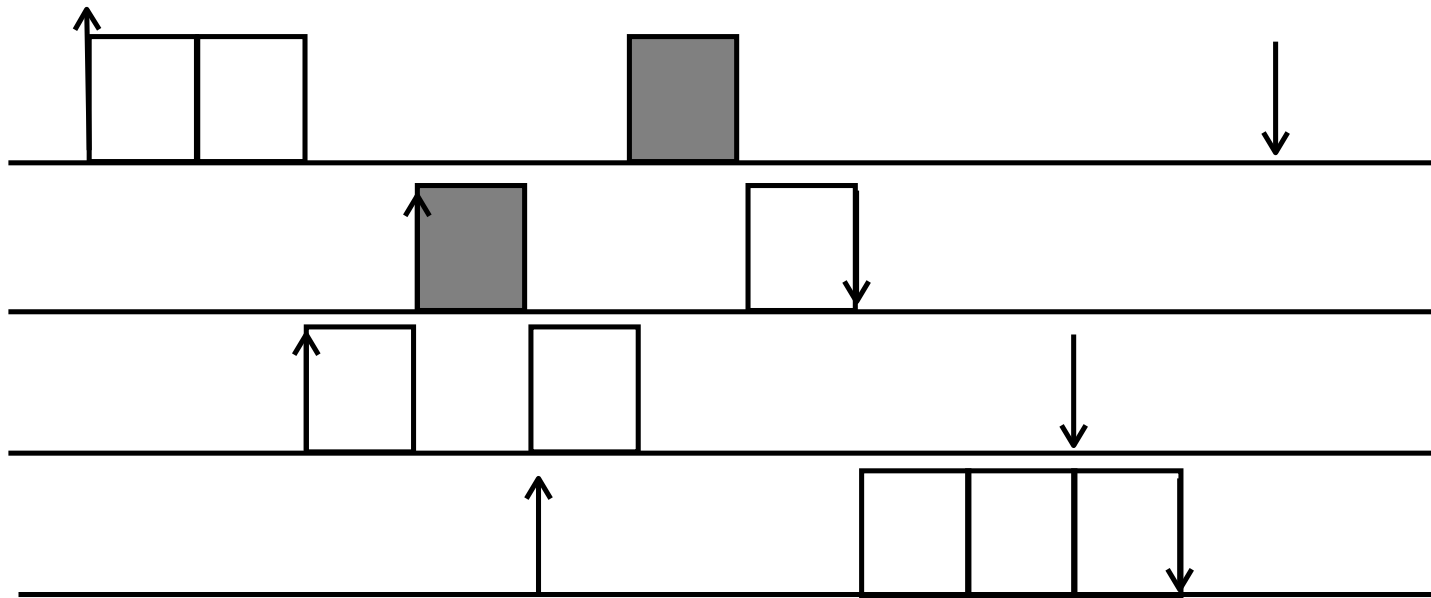
# Itération

- On “avance” dans le temps en faisant s’exécuter la tâche qui à l’échéance la plus proche en échangeant des parties du traitements.

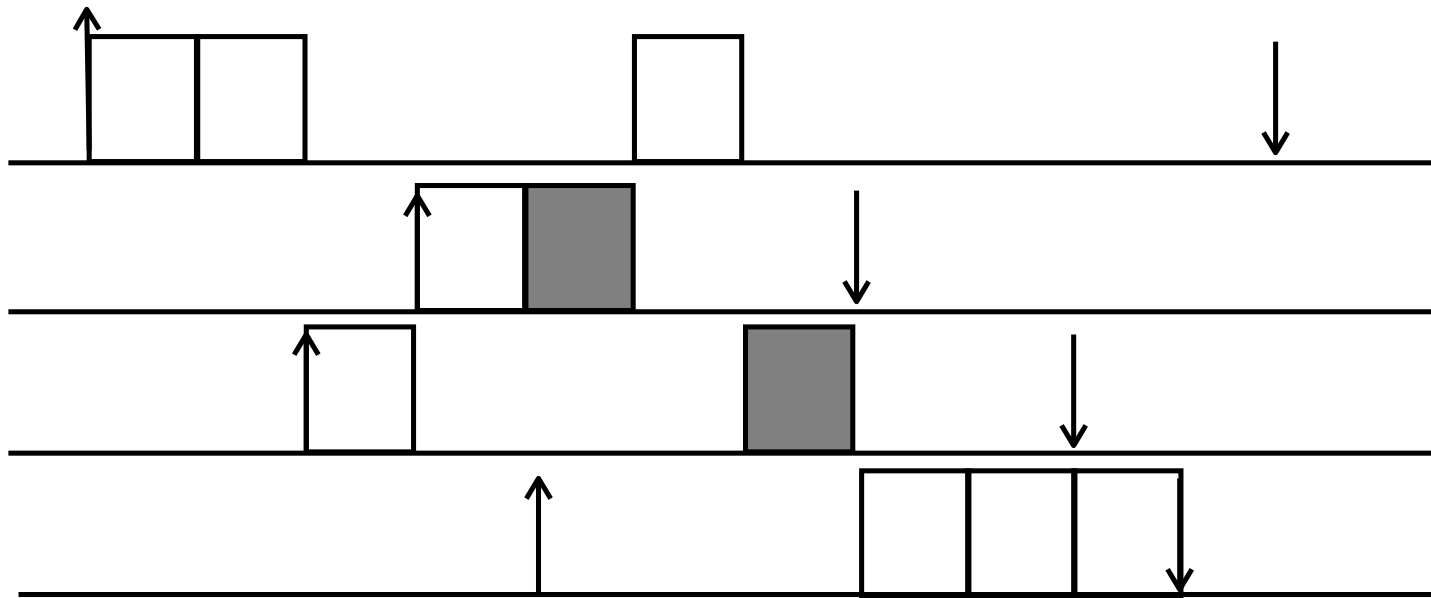


# Itération

---



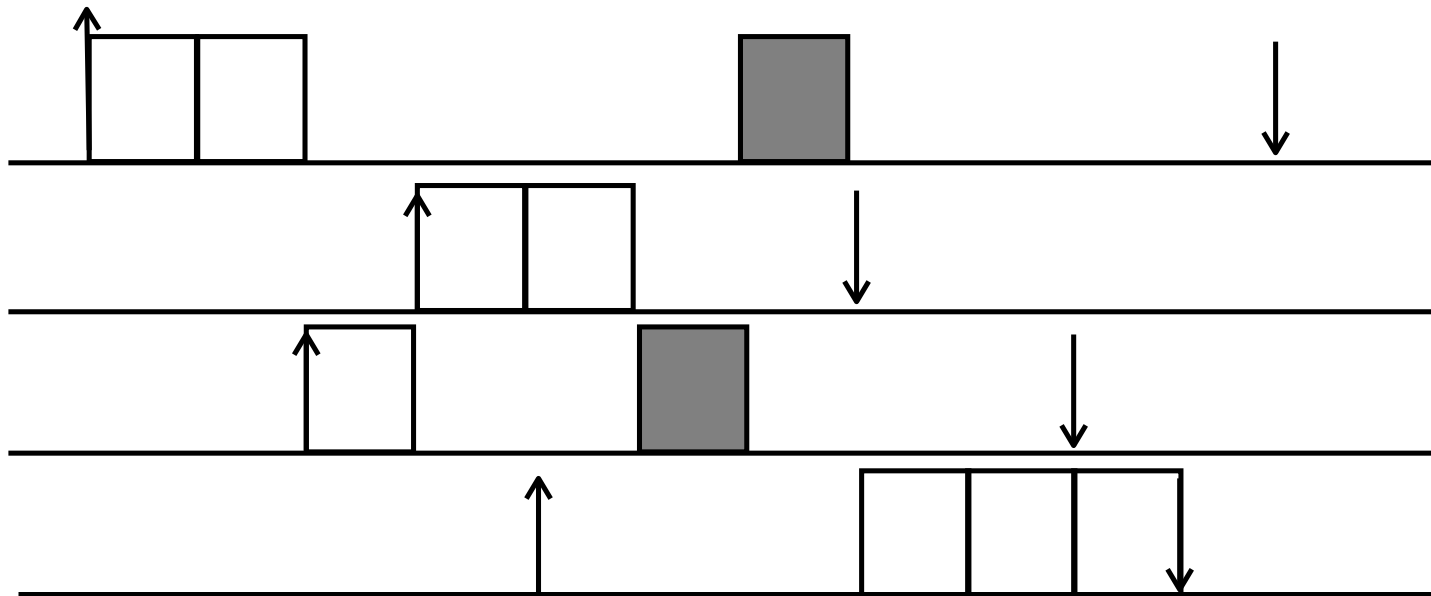
# Itération



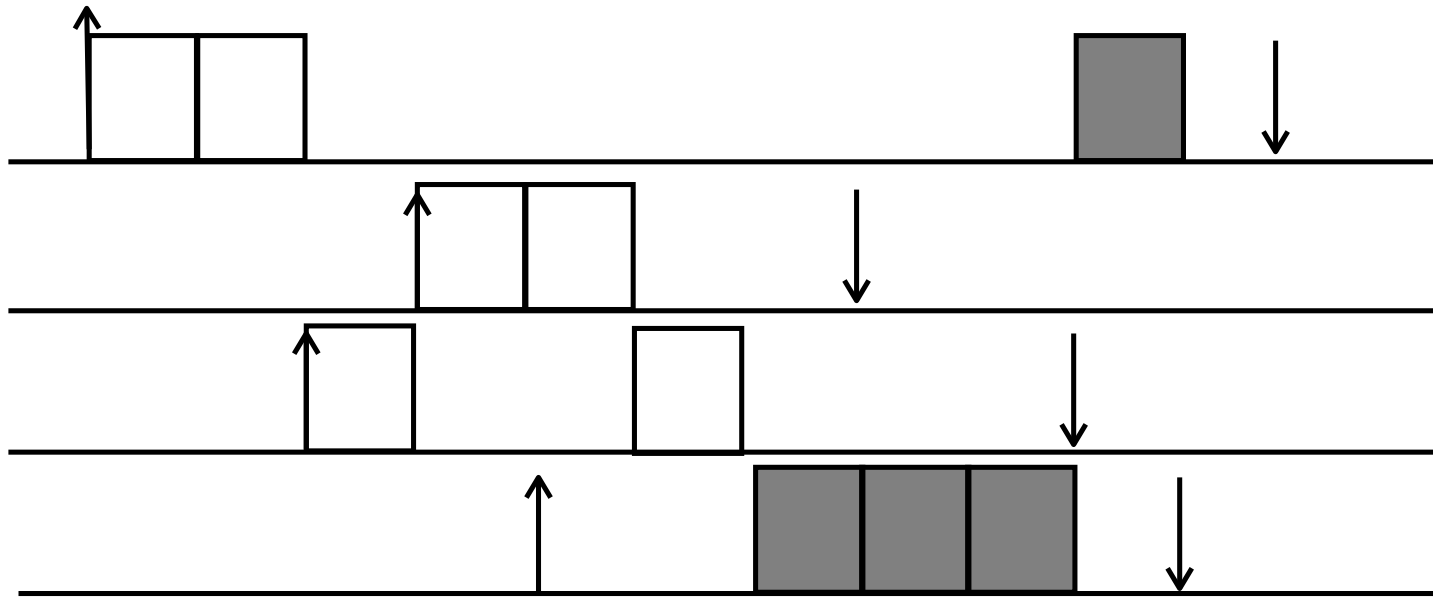


# Itération

---



# ... Itération



# Exercice

---



Donner les algorithmes des ordonnancement EDF et LLF en pseudo-code.

# Tâche avec communications

---

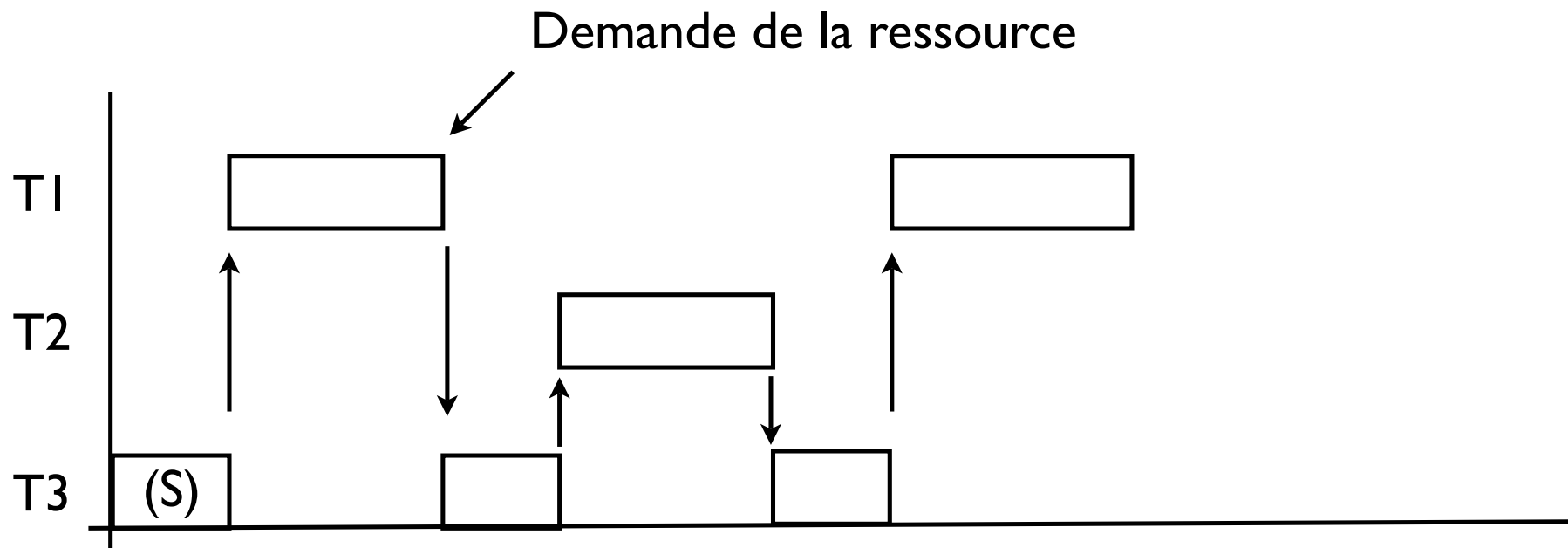
 Hypothèse du monotone par taux

 mais les tâches accèdent à des ressources communes => dépendance entre les tâches

# Exemple d'inversion de priorité



Priorité de  $T1 > T2 > T3$  qui partagent une ressource protégée par un sémaphore S



# Solution : héritage de priorité

---

- Pas de préemption si tâche en SC
  - si les accès au SC sont courts c'est acceptable sinon ça va être pénalisant pour les autres tâches
- => héritage de priorité

# Héritage de priorité

---



## Sémaphore à priorité (principe) :

gère la liste des tâches qui veulent acquérir le sémaphore.

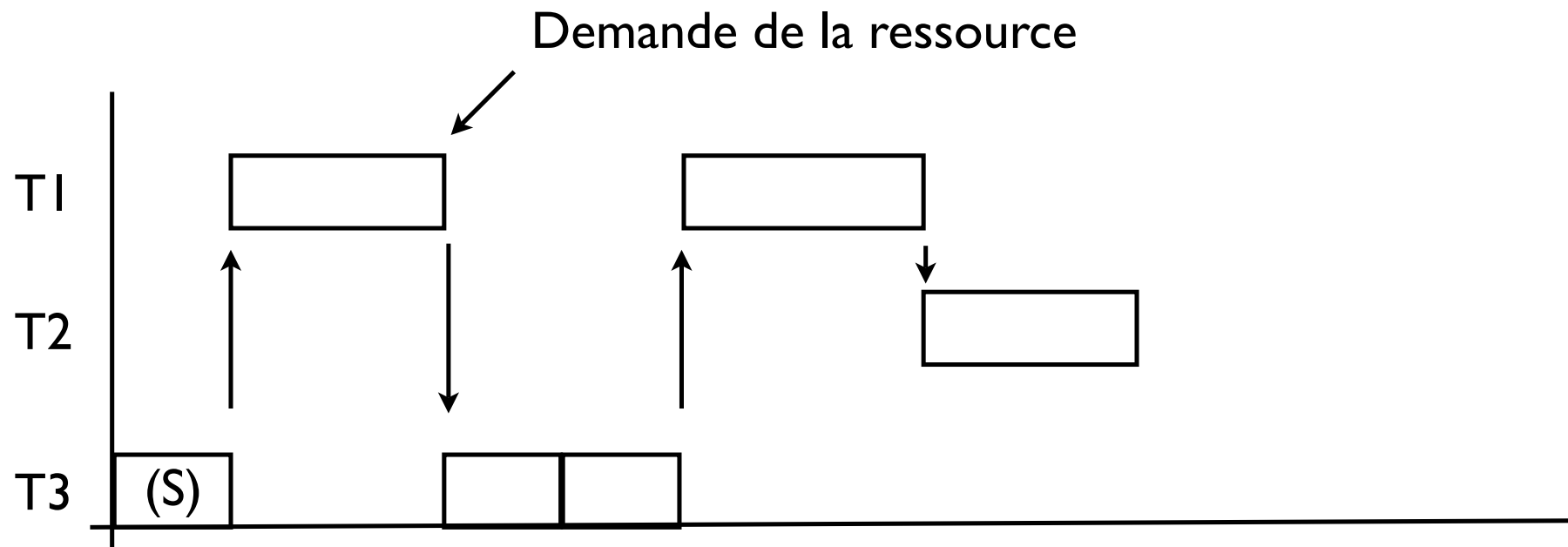
si T a obtenu le sémaphore, si T' demande le sémaphore :

si priorité de T < priorité de T' alors

    priorité T = priorité T'

fsi

# Exemple d'inversion de priorité






T3 a le sémaphore, T1 demande le sémaphore

=> Priorité de  $PrioT3 = PrioT1$



# Partage de plusieurs ressources

---

-  Dans le cas de plusieurs ressources possibilités de blocage (imbrication des sémaphores).
-  Mettre les SC en exclusion mutuelle
-  Ordonner les SC (une SC débute que si elle a une priorité supérieure à toutes les autres).

# Processus apériodiques


---

- Contrainte dure => processus périodiques
- Contrainte lâche => processus apériodiques
  - Priorité des tâches apériodiques = faible
  - Insertion d'une tâche périodique pour traiter les tâches apériodiques à traiter en priorité.

# Méthodes :

---

## Scrutation

-  Tâche périodique avec une fenêtre de traitement compatible avec les autres tâches

## Tâche de fond

-  Utiliser les temps laissés libre au fin de période

# Conclusion/Résumé

Monotone par taux	EDF	LLF
Statique	Dynamique	Dynamique
Test = Charge < 70%	Test = Charge < 100%	Test = Charge < 100%
Théorème de la zone critique => Précision	Moins de changements de contexte	
Optimal	Optimal	
	+/- quand les taches arrivent en désordre et $c_i$ inconnu	+/- quand les taches arrivent en désordre et $c_i$ connu