Ordonnanceur

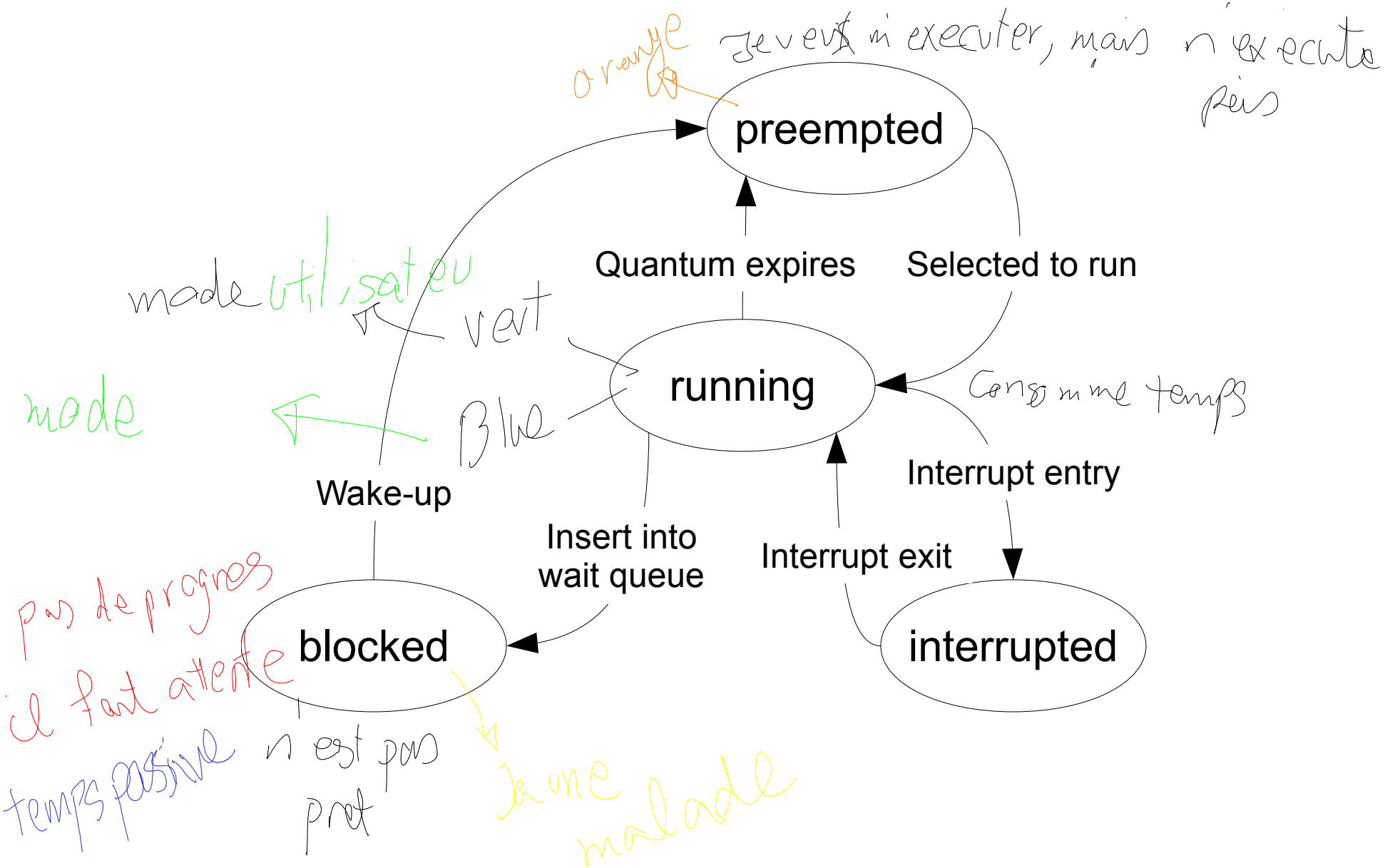
INF3173 – Principes des systèmes d'exploitation
Automne 2024

Francis Giraldeau
giraldeau.francis@uqam.ca
·
Université du Québec à Montréal

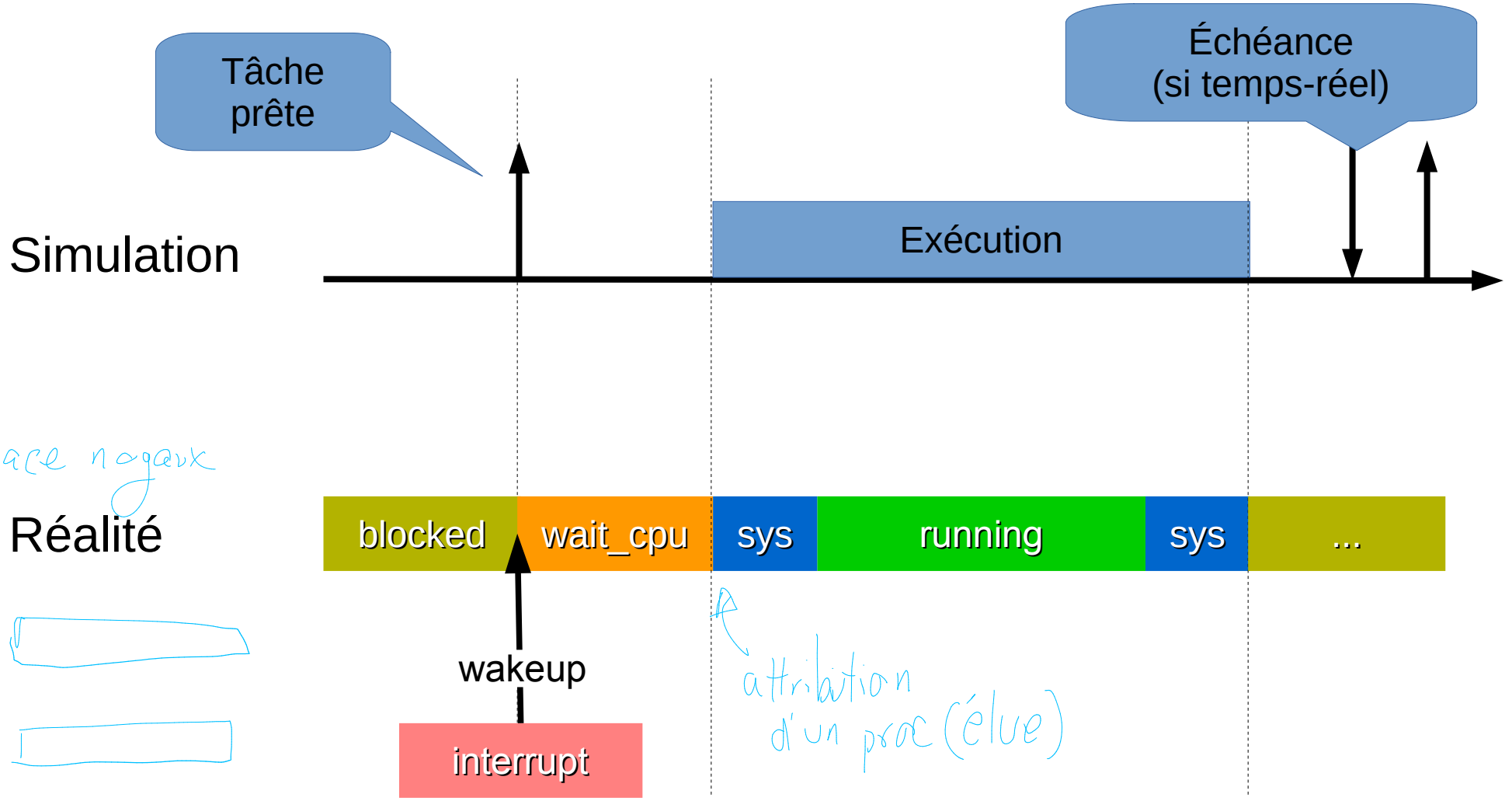
Agenda

- Rôle de l’ordonnanceur
- Caractéristiques
  - Préemptif
  - Quantum
  - Priorité
  - Multiprocesseurs
  - Types de charge (cpu, I/O)
  - Inversion de priorité
- Algorithmes classiques
- Étude de cas avec Linux

État des tâches

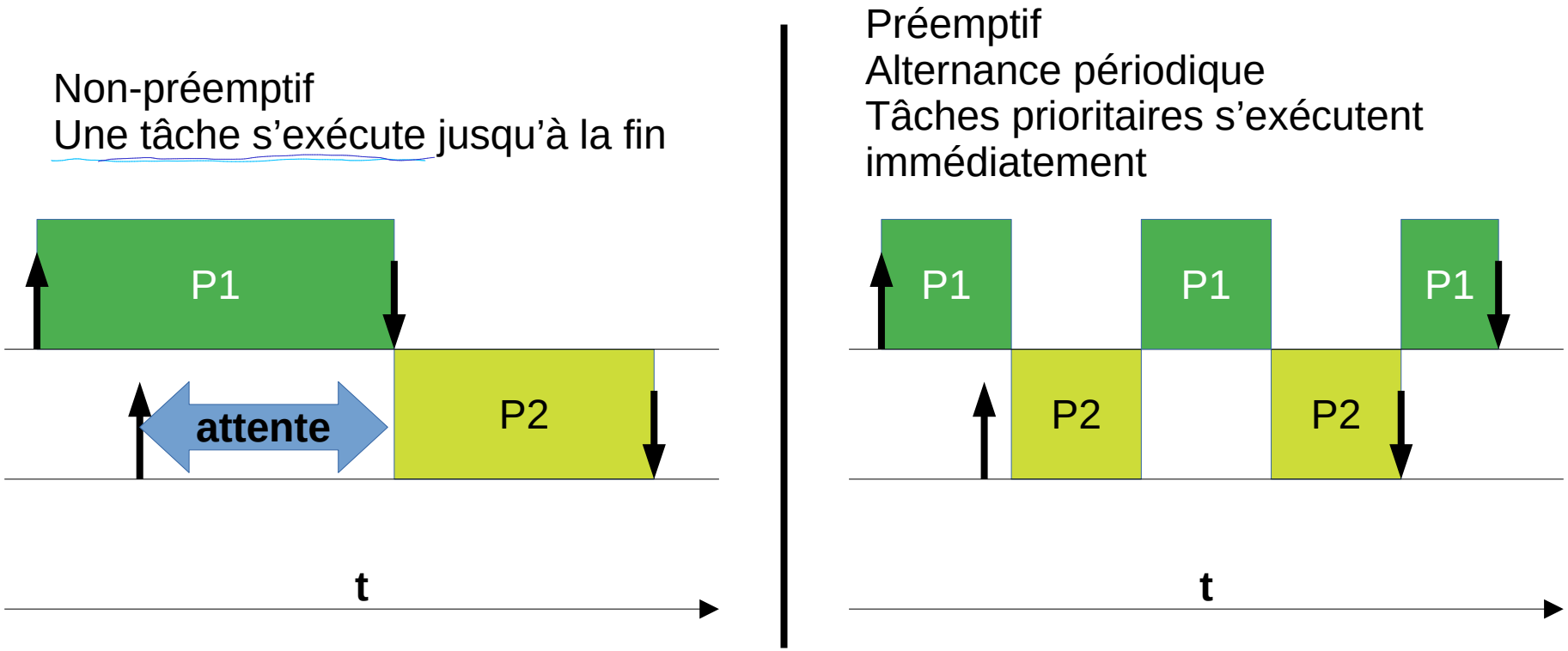


Simulation v.s. réalité



Préemptif v.s. Non-préemptif

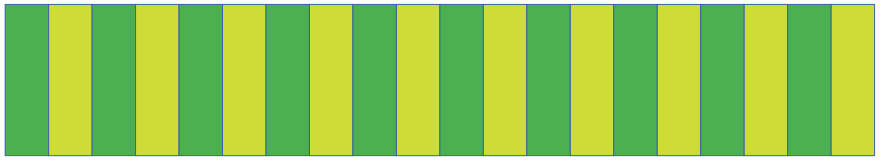
- La possibilité de changer une tâche qui est en train de s'exécuter à tout moment
- Les SÉ modernes sont préemptifs



Quantum

- Temps limite d'exécution d'une tâche avant de céder le processeur

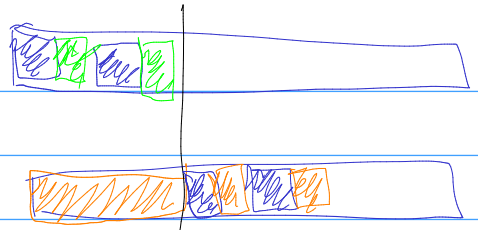
Période courte
Avantage: interactivité
Inconvénient: surcoût et baisse de l'efficacité des caches



Période longue
Avantage: améliore le débit total
Inconvénient: les tâches sont en pause longtemps

CPU1

CPU2



P1 50%

P2 50%

P3 100%

P1 P2

P3, P1

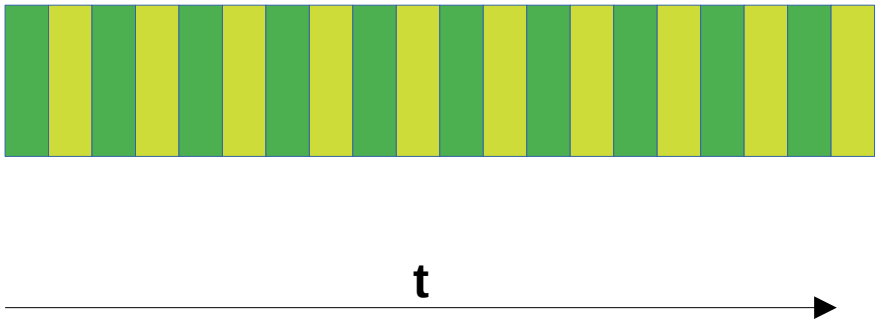
migration

date de tâche par proc.

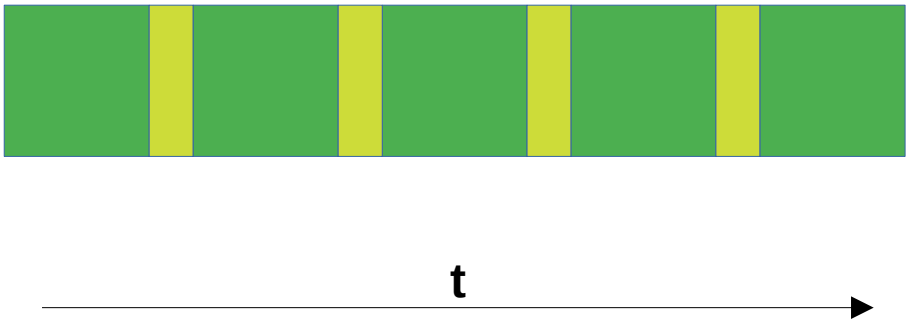
Priorité d’une tâche

- Proportion de temps processeur alloué en moyenne

Priorité P1 == P2
Alternance périodique uniforme

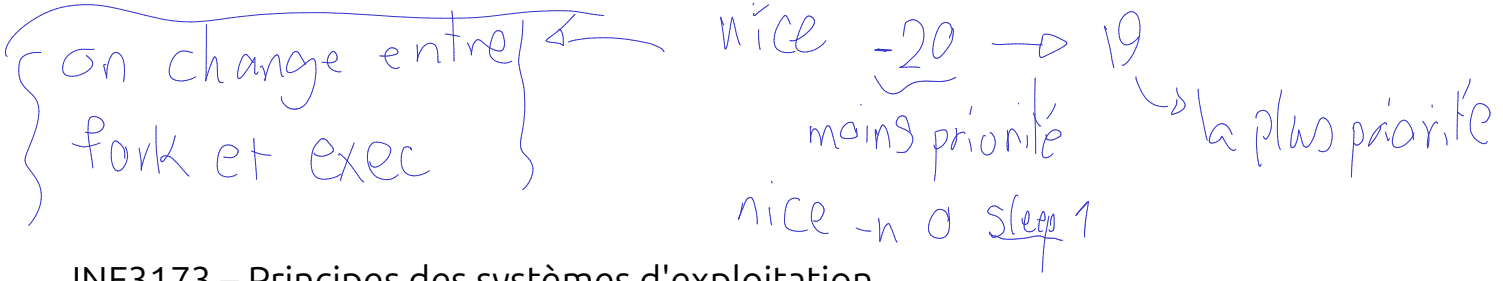


Priorité de P1 > P2
Quantum de P1 est allongé
En moyenne, P1 roule plus souvent
P2 roule quand-même à l’occasion
(pas de famine)



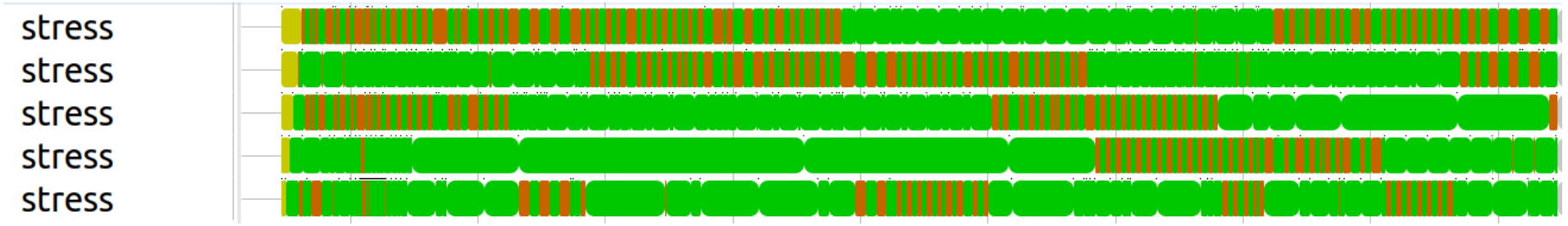
Répartition entre processeurs

- Si le nombre de tâches n'égal pas le nombre de processeurs
- Exemple: 3 tâches sur 2 processeurs
  - P1 et P2 roulent sur CPU 1 (50% du temps chacun)
  - P3 roule sur le CPU 2 (100% du temps)
- Il faut migrer des tâches d'un processeur à l'autre à l'occasion pour utiliser tous les coeurs
- ... Mais pas trop souvent, car cause un surcoût élevé

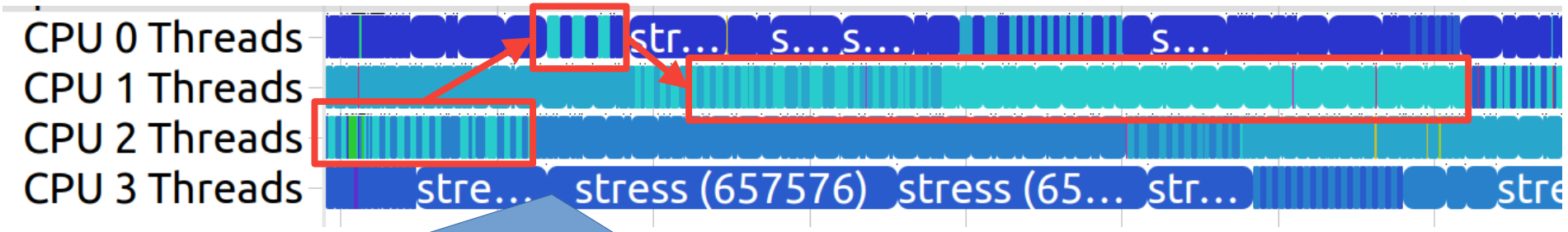


Exemple de rebalancement

Vue des tâches



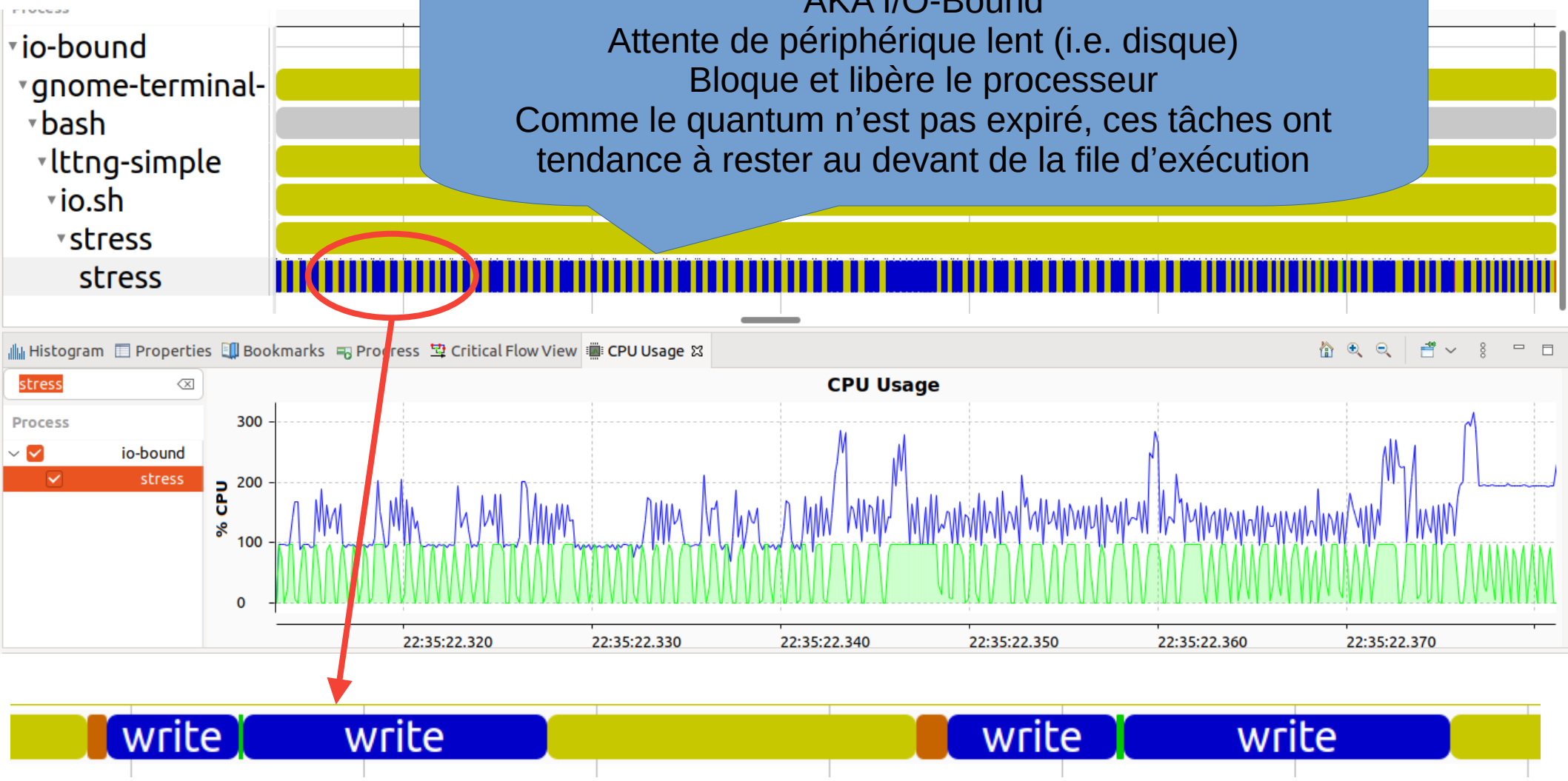
Vue des ressources (par processeur)



Il existe une file d'exécution par processeur.
En fonction du taux d'occupation de chaque processeur et des tâches à exécuter l'ordonnanceur peut migrer une tâche d'un processeur à un autre.

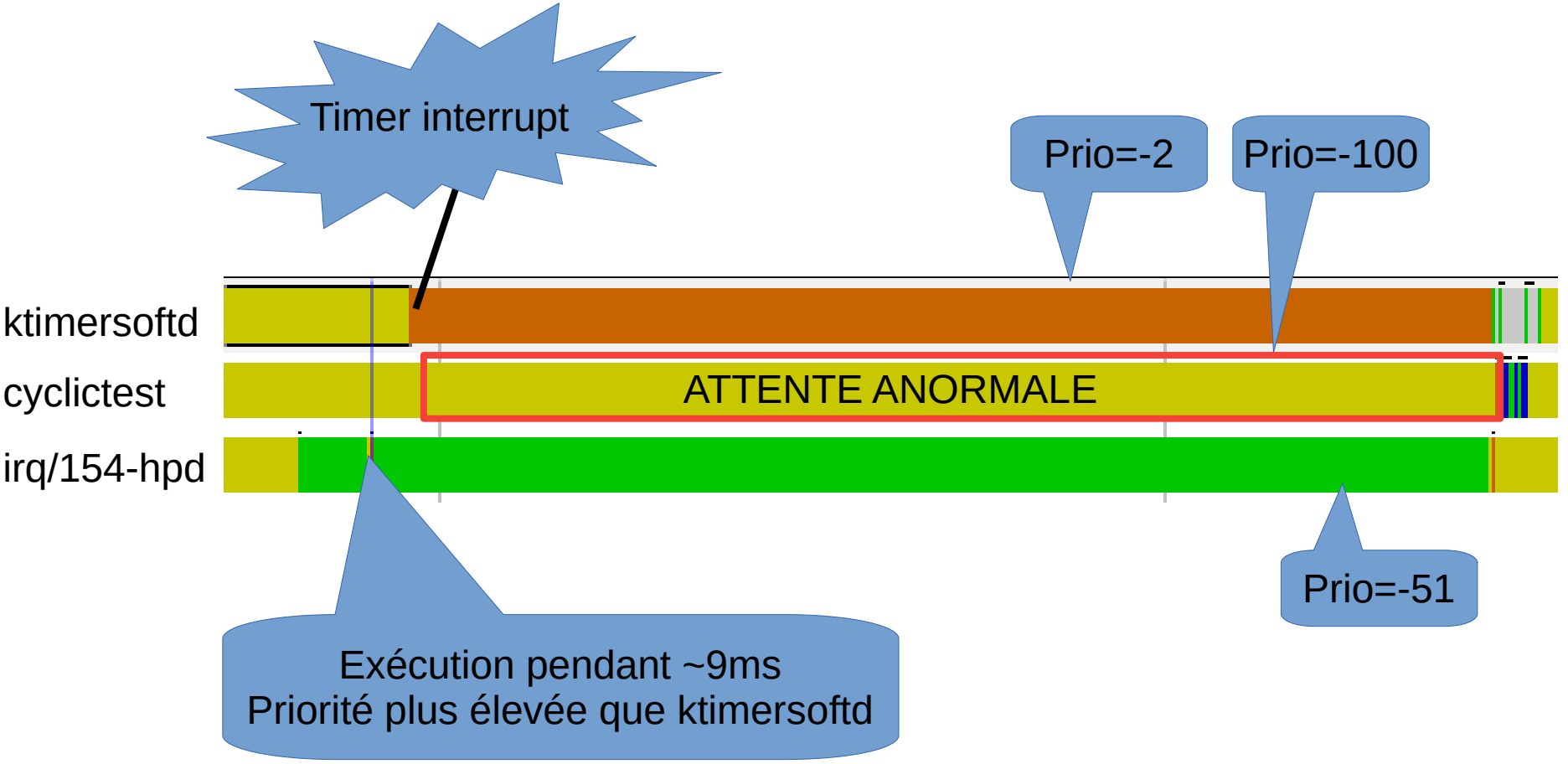
Processus limité en entrée-sortie

AKA I/O-Bound
Attente de périphérique lent (i.e. disque)
Bloque et libère le processeur
Comme le quantum n'est pas expiré, ces tâches ont tendance à rester au devant de la file d'exécution



Exemple inversion de priorité

- Se produit quand une tâche très prioritaire échoue à s'exécuter dans les temps à cause d'une dépendence d'une tâche de plus basse priorité.



Algorithme First Come First Served (FCFS) non-préemptif

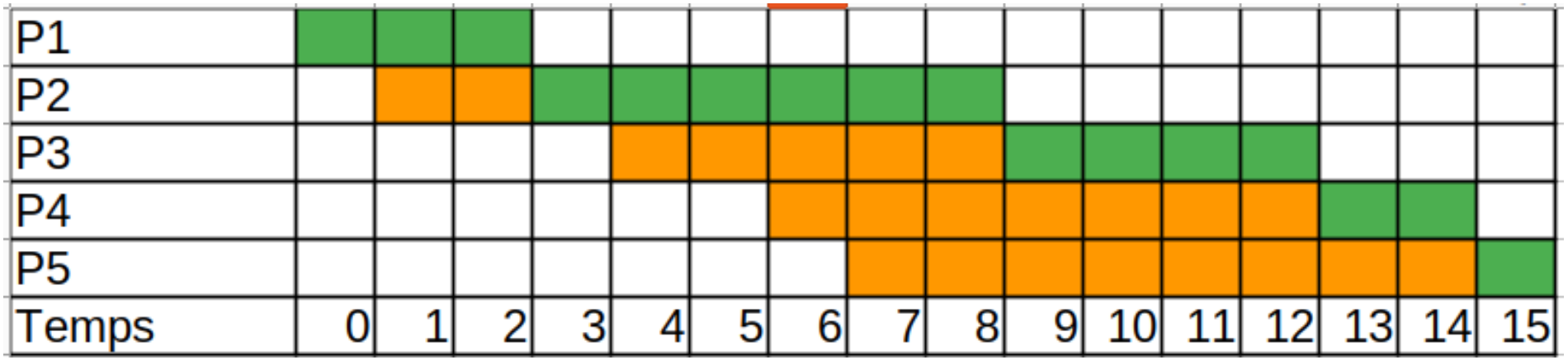
			FCFS	
Tâche	Arrivée	Exécution	Attente	Séjour
P1	0	3	0	3
P2	1	6	2	8
P3	4	4	5	9
P4	6	2	7	9
P5	7	1	8	9
Moyenne			4.4	7.6
Maximum			8	9

Temps séjour = attente + exécution

compt vert

compt orange

Charte de Gantt

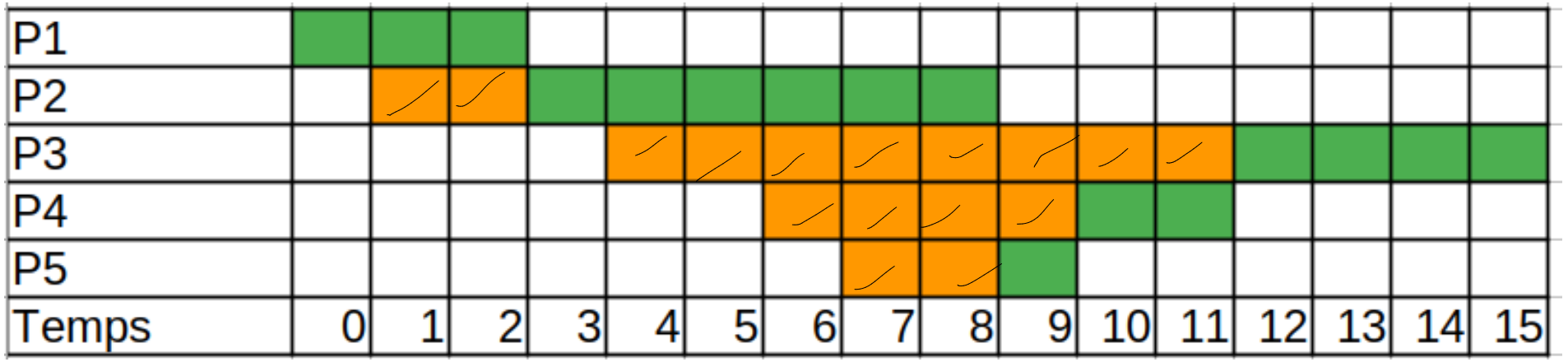


Algorithme Short Process First (SPF) non-préemptif

			SPF	
Tâche	Arrivée	Exécution	Attente	Séjour
P1	0	3	0	3
P2	1	6	2	8
P3	4	4	8	12
P4	6	2	4	6
P5	7	1	2	3

Moyenne	3.2	6.4
Maximum	8	12

Charte de Gantt



Algorithme Short Process First (SPF) non-préemptif

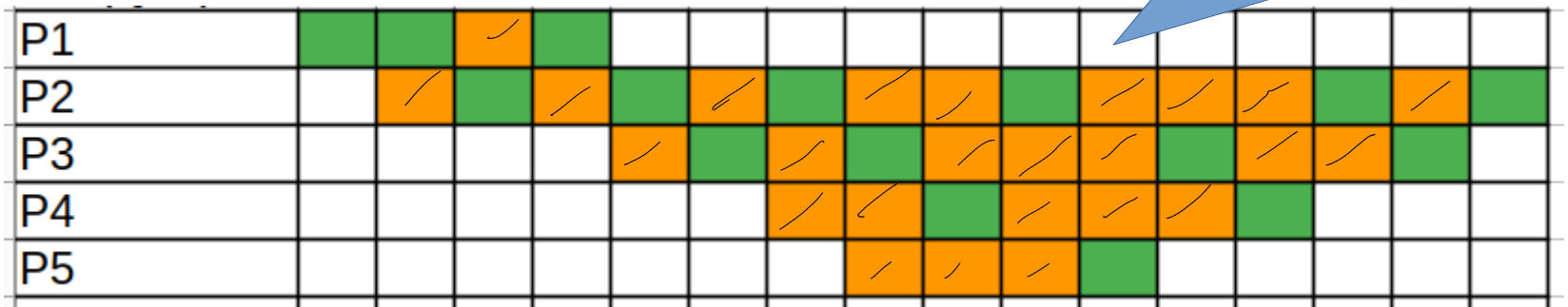
- Parmi les processus prêts, le processus élu est celui dont la prochaine rafale est la plus courte.
- Comment estimer le temps de la prochaine rafale ?
- Le temps de la prochaine rafale de chaque processus est estimé en se basant sur le comportement passé du processus.
- Soient Si et Ti les temps d'exécution estimé et mesuré pour la iième rafale, les estimations successives sont :
- Équation de récurrence: $S = a * T + (1 – a) * S$

Algorithme circulaire (Round-Robin) Préemptif

			RR	
Tâche	Arrivée	Exécution	Attente	Séjour
P1	0	3	1	4
P2	1	6	9	15
P3	4	4	7	11
P4	6	2	5	7
P5	7	1	3	4

Moyenne	5	8.2
Maximum	9	15

Mise à jour de la file d'exécution:
1) mettre la tâche courante à la fin
2) ajouter les nouvelles tâches à la fin



P1																
P2																
P3																
P4																
P5																
Temps	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	P1	P1	P2	P1	P2	P3	P2	P3	P4	P2	P5	P3	P4	P2	P3	P2
		P2	P1	P2	P3	P2	P3	P4	P2	P5	P3	P4	P2	P3	P2	
							P4	P2	P5	P3	P4	P2	P3			
								P5	P3	P4	P2					

Completely Faire Scheduler (CFS)
Préemptif

- Implémenté dans le noyau Linux
- Variante de Round-Robin avec priorité
- Arbre binaire contient les tâches triées par ordre croissant du temps d’exécution cumulatif (vruntime)
- La tâche à gauche est celle qui a le temps d’exécution le plus faible O(1)
- Quand le quantum de la tâche expire ou après réveil, insertion dans l’arbre en temps O(log n)

Exemples

- Programme werk:
  - Calcule pour un temps donné
  - Permet d’expérimenter différentes priorités et classes d’ordonnanceurs
- ./werk <nom> <durée>
- ./werk p1 | ./werk p2
- Varier la priorité avec nice -n 10

Trace werk

```
$ nice -n 4 werk p1 | werk p2
...
p1 elapsed=5.0s running=1.5s wait=3.5s share=0.30
p2 elapsed=5.0s running=3.5s wait=1.5s share=0.71
```