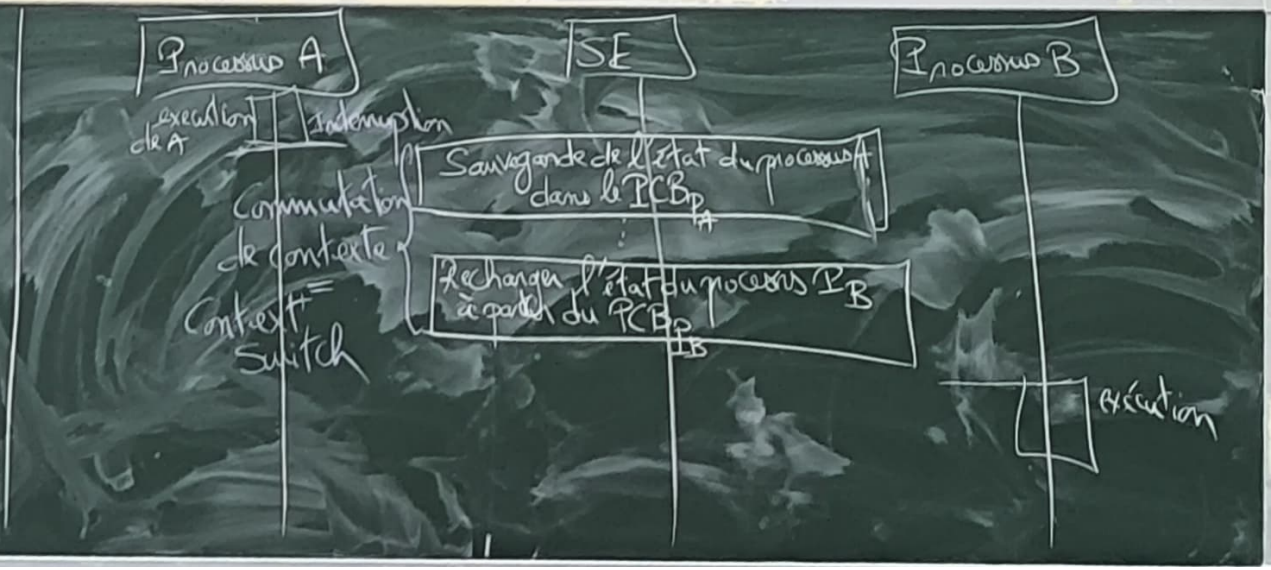
On considère l'algorithme STF avec priorité et commutation de contexte

On considère le tableau suivant et le TCC = 2 ms

TCC = temps de commutation de contexte

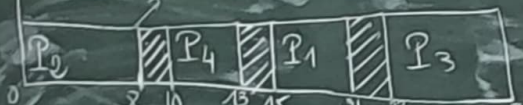
Arrivée	BT
	6
	8
2	7
	3



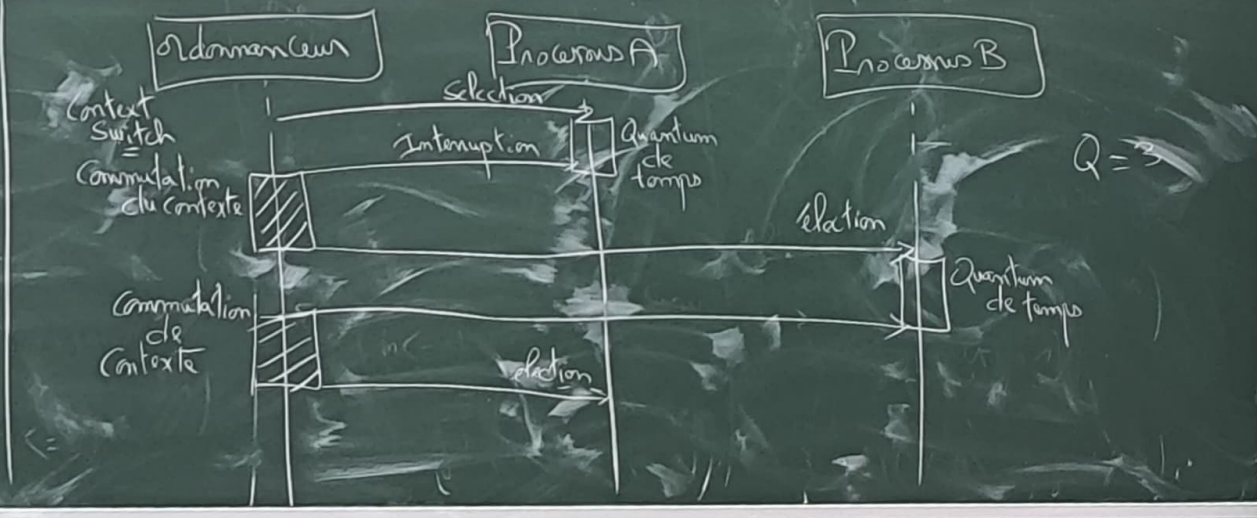
On considère l'algorithme STF avec priorité et commutation de contexte

On considère le tableau suivant et le TCC = 2 ms

TCC = temps de commutation de contexte

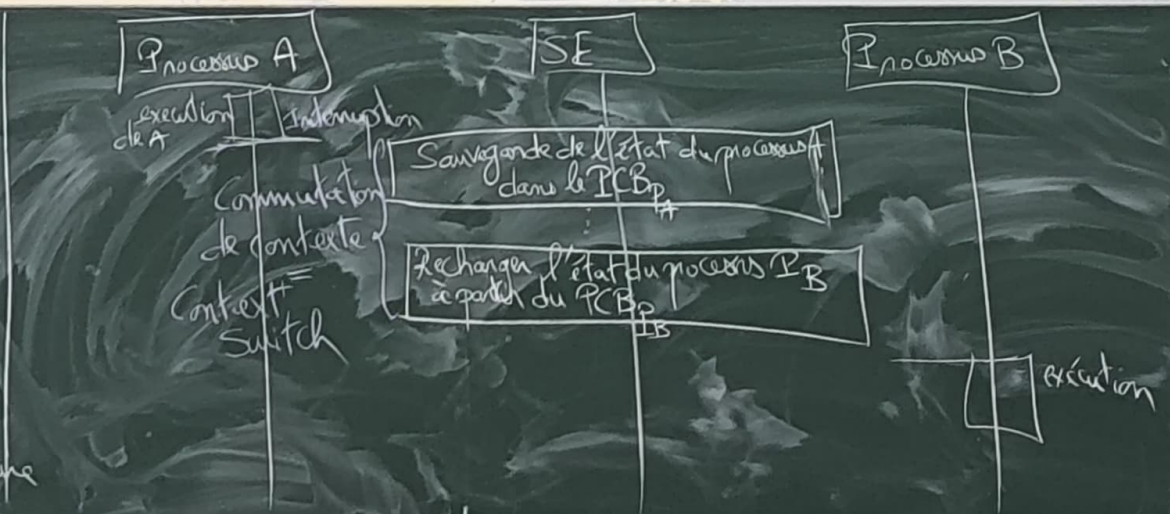


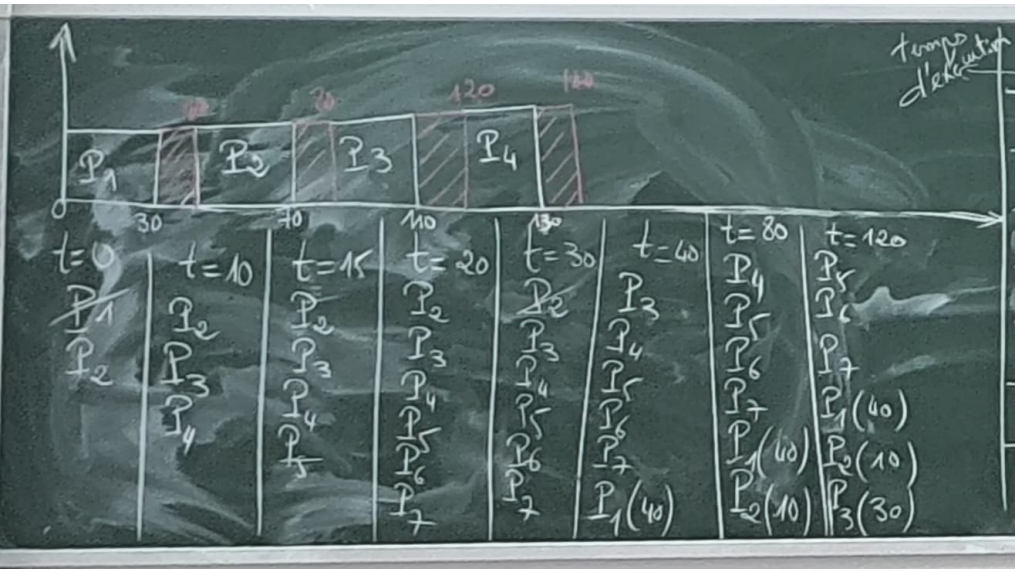
t=0	t=2	t=3	t=8	t=13	t=21
P1	P1	P1	P1	P1	~~P3~~
~~P2~~	P3	P3	P3	P3	
		~~P4~~	~~P1~~		



P	Arrivée	BT	Priorité	Temps de rotation	Temps d'attente	début
P1	0	6	2	21 - 0 = 21	21 - 6 = 15	
P2	0	8	(1)	8 - 0 = 8	8 - 8 = 0	4/30
P3	2	7	3	30 - 2 = 28	28 - 7 = 21	
P4	3	3	2	13 - 3 = 10	10 - 3 = 7	

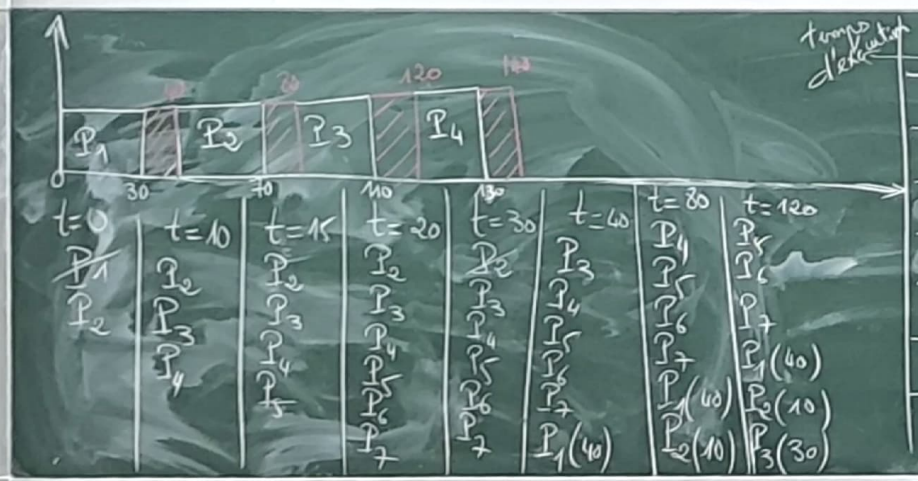
Hypothèse : le processus ayant la priorité la (+) faible est le plus prioritaire





temps d'exécution

P	BT	Arrivée	temps de rotation	temps d'attente	Débit
P1	70	0			
P2	40	0			
P3	60	10			
P4	10	10			
P5	20	15			
P6	40	20			
P7	10	20			



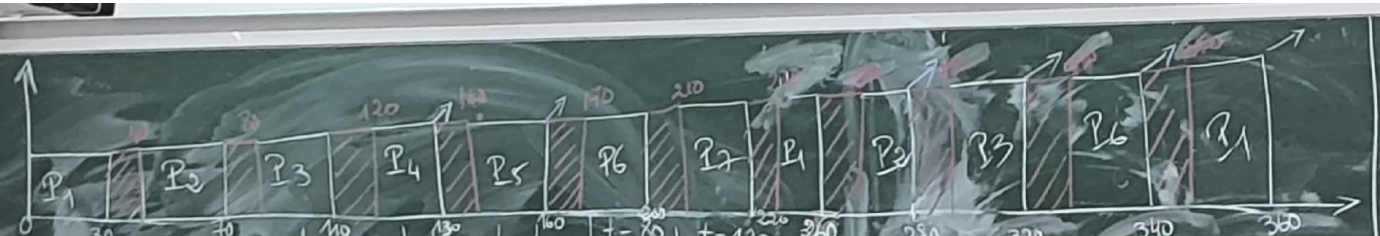
P	BT	Arrivée	temps de rotation	temps d'attente	Délai
P1	70	0			
P2	40	0			
P3	60	10			
P4	10	10			
P5	20	15			
P6	40	20			
P7	10	20			

t=140

Ex2 : On considère l'algorithme RR avec : Q = 30 ms

Hypothèse : 1- Placer par ordre alphanumérique les processus qui arrivent au même temps. TCC = 10 ms

2- L'ordre des processus dans la FA est en FIFO



t=0	t=10	t=15	t=20	t=30	t=40	t=80	t=120
P1	P2	P2	P2	P2	P3	P4	P5
P2	P3	P3	P3	P3	P4	P5	P6
	P4	P4	P4	P4	P5	P6	P7
		P5	P5	P5	P6	P7	P1(40)
			P6	P6	P7	P1(40)	P2(10)
			P7	P7	P1(40)	P2(10)	P3(30)

Remarque

Si on ↗ quantum

* moins de commutation de contexte
* Meilleurs pour les processus longs

Si on ↘ quantum

* Plus de Commutation de contexte
* Meilleur pour les processus courts
* Réduit l'efficacité du processeur

t=140	t=210
P6	P1(40)
P7	P2(10)
P1(40)	P3(30)
P2(10)	P6(10)
P3(30)	

Remarque : Pour les critères de performance :

- temps de rotation : Critère à diminuer (↘)
- temps d'attente : Critère à diminuer (↘)
- Le débit : Critère à augmenter (↗)

Ex2 : On considère l'algorithme RR avec : Q = 30 ms

Hypothèse : 1- Placer par ordre alphanumérique les processus qui arrivent au même temps. TCC = 10 ms

2- L'ordre des processus dans la FA est en FIFO

if (wait(&status) > 0)

WIFEXITED(status) > 0 (fin de tâche avec succès)

WEXITSTATUS => On peut afficher la valeur du code de retour de la variable status

WIFSIGNALED => S'il y a un signal extérieur qui impose une fin de tâche forcée

WTERMSIG => Renvoyer le code de retour du signal envoyé

L'appel-système "exec" permet d'exécuter des commandes, des appels systèmes dans d'autres scripts ou pg

```
execle("/bin/mkdir", "mkdir", toto, NULL, NULL);
```

var d'env — fin de la liste

... + on peut ajouter des variables d'env

les arguments sont passés ... qu'une liste

exec :
- execl
- execlp
- execle
- execv
- execvp
- execve

```
execl("/bin/mkdir", "mkdir", "test", NULL);
```

- "/bin/mkdir" : path = chemin de la commande exécutable
- "mkdir" : nom de la cmd
- "test" : nom du dossier
- NULL : fin de la liste des arguments

```
execlp("mkdir", "toto", NULL);
```

PATH de la commande ... mis dans execl

exec V => char *args[] = {"mkdir", "toto", NULL}

V : vecteur

```
execv("/bin/mkdir", args);
execvp(args)
execve("/bin/mkdir", args)
```

WIFSIGNALED => S'il y a un signal extérieur qui impose une fin de tâche forcée

WTERMSIG => Renvoyer le code de retour du signal envoyé