La Knack : <https://github.com/Knackie/telecom/blob/master/Cours/3A/Promo%202018-2021/SETR/TD1.pdf>

## Projet SCEPTRE

### Exercice 1 :

Producteur unique / Consommateur unique :

### 

**Producteur :**

ENTRER (R1)

ENVOYER (X, F) *// Ajoute la donnée dans la file d'attente*

SIGNALER (F\_Non\_Vide, Conso) *// Signal que la donnée est arrivé au conso*

SORTIR (R1)

**Consommateur :**

ENTRER (R1)

si VIDE (F) alors

EFFACER (F\_on\_Vide) *// Pour être sur que F\_Non\_Vide n’est pas déjà arrivé et*

*que l’attente va bien être bloquante*

SORTIR (R1)

ATTENDRE (F\_Non\_Vide) *// Attente de production*

sinon

SORTIR (R1)

fin si

RETIRER (Y, F) *// Retire le message de la file*

### 

### 

### Exercice 2 :

**Pseudo Code :**

**P(s)**

cpt - -;

si (cpt < 0) alors <mettre tâche en q>

**V(s)**

cpt + +;

si (cpt <= 0) alors réveiller tâche en q>

**En SPECTRE :**

**P(s) :**

ENTRER(R1) *// Personne d’autre va exécuter la fonction entretemps*

cpt = cpt - 1

si (cpt < 0) alors

ENVOYER (TACHE\_COURANTE, F) *// Ajoute la tâche dans la file*

EFFACER (E1) *// Remet les événements de la liste dans l’état non arrivé*

SORTIR (R1) *// Libère la fonction P(s)*

ATTENDRE (E1) *// wait*

sinon

SORTIR(R1) *// Libère la fonction P(s)*

fin si

**V(s) :**

ENTRER (R1) *// Personne d’autre va exécuter la fonction entretemps*

cpt = cpt + 1

si (cpt <= 0) alors

RETIRER (T, F) *// Retire la tâche de la file. T la tâche en tête de file*

SIGNALER (E1, T) *// Réveille la tâche suivante dans la file*

fin si

SORTIR(R1) *// Libère la fonction V(s)*

### 

## Ordonnancement

### Exercice 1

| Temps | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RM | A | A | B | B | C | C | A | A | B | B |
| EDF | A | A | B | B | C | C | C | A | A | B |
| LLF | A | A | B | B | C | C | A | A | C | B |

| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | / | A | A | C | C | B | B | A | A |
| B | / | A | A | C | C | C | B | B | A |
| B | / | A | A | C | C | B | B | A | A |

| 20 | 21 | 22 | 23 |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | / | / | / |  |  |  |  |  |  |
| A | / | / | / |  |  |  |  |  |  |
| C | / | / | / |  |  |  |  |  |  |

RM => Priorité fixe

EDF => 1/Temps avant la deadline

LLF => D - t - C(t)

**RM :**

**Test acceptabilité :**Test suffisant pour prouver que les tâches peuvent être ordonnancés

Ici = 0.83 -> Test très contraignant donc test de terminaison

**Test de terminaison**

i = 1, TA

W(t=0) = Ca = 2

W(t=2)=2 <= Da = 6

i = 2, TA, TB

W(t = 0) = Cb = 2

W(t = 2) =

i = 3 TA, TB, TC

w(t=0) = = Cc = 3

w(t=3) = = Ca + Cb + Cc = 7

w(t=7) = 2Ca + Cb + Cc = 9

w(t=9) = 2Ca + 2Cb + Cc = 11

w(t=11) = 2Ca + 2Cb + Cc = 11

Test de terminaison finito => Les tâches respectent les contraintes de temps qqsoit l'ordre des tâches à l'infini

**EDF :**

**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancés

Ici = 0.83 -> C'est bon

**LLF**

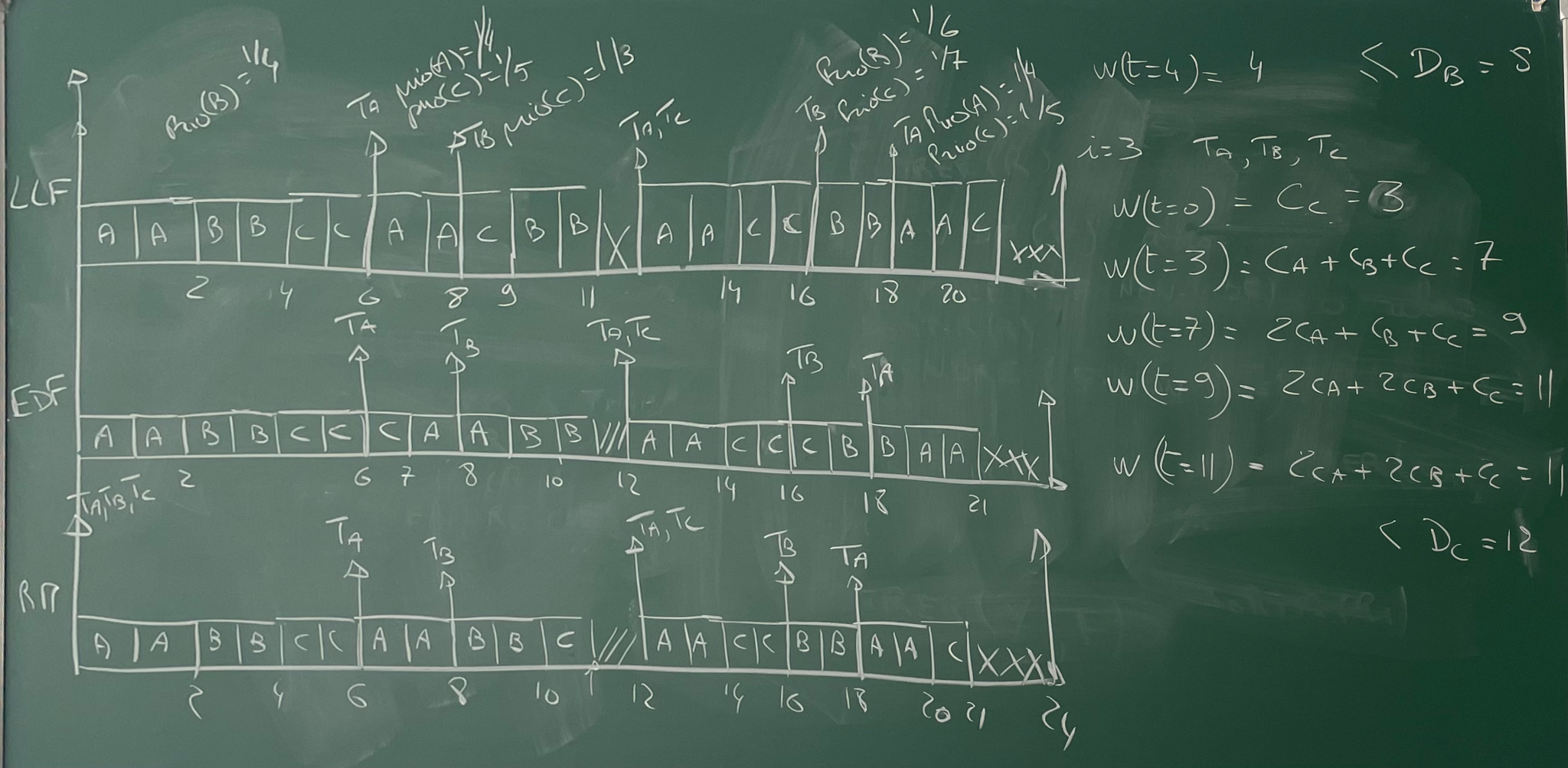
**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancés

Identique qu' EDF => c'est bon

Priorité =

La marge :



### Exercice 2

**EDF :**

**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancées

⇒ OK

**RM :**

**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancées

**Test de terminaison :**

Priorité:

i = 1, TA (1/TA = ¼ et 1/TB = 1/10)

=> OK

i = 2, TA, TB

⇒ Cassé pas ok

| Temps | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RM | A | A | B | B | A | A | B | B | A | A | B |
| EDF | A | A | B | B | A | A | B | B | B | A | A |

| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B |  |  |  |  |  |  |  |  |  |
| B | A | A | B | B | B | B | B | A | A |

B ne respect pas les échéances car il finit sa première activation après 10

### 

### Exercice 3

**EDF :**

**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancées

⇒ OK

**RM :**

**Test acceptabilité :**

Test suffisant pour prouver que les tâches peuvent être ordonnancées

0.84

⇒ KO

Test de terminaison :

T1 = TA / 3 et T2 = TB / 3 ⇒ C’est le même problème que l’exercice 2 donc ces tâches ne peuvent pas être ordonnancées avec RM.

### Exercice 4

(R : date de réveil, C : capacité (temps nécessaire pour s'exécuter), D=T : deadline/périodicité)

Tav = (r0 = 0, C1 = 25, D=T= 50)

Tar = (r0 = 0, C1 = 100, D ? )

**RM :**

Priorité:

i = 1, Tav (1/Tav = ¼ et 1/Tar = 1/10)