NOM CUESTA

Prénom Quentin

Promo 2017 M1

Date 07.04.2016

14,

CUESTA Quentin
M1 - 2015

MATIÈRE Real-time Systems. 15/24

1. Course questions.

1. Un système temps-réel est un système qui gère des tâches asynchrone dans un délais donné. Il est indispensable aux systèmes embarqués (1)

2. Dans une exécution "hard real-time", les processus ne peuvent pas manquer leur fenêtre temporelle. Alors que dans une exécution "soft real-time", les processus peuvent manquer leur fenêtre temporelle. dans une certaine mesure ! (1)

3. Determinism = Déterministe : signifie que pour un même cas d'utilisation, l'exécution sera toujours la même.
Reliability = fiabilité : Un système est dit fiable si il est stable, maintenable et qui ne perd aucune information dans le temps. (1,5)
Predictability = Prédictabilité : un système est dit prédictible si son exécution prévoit tout les cas de figure. (0,5)

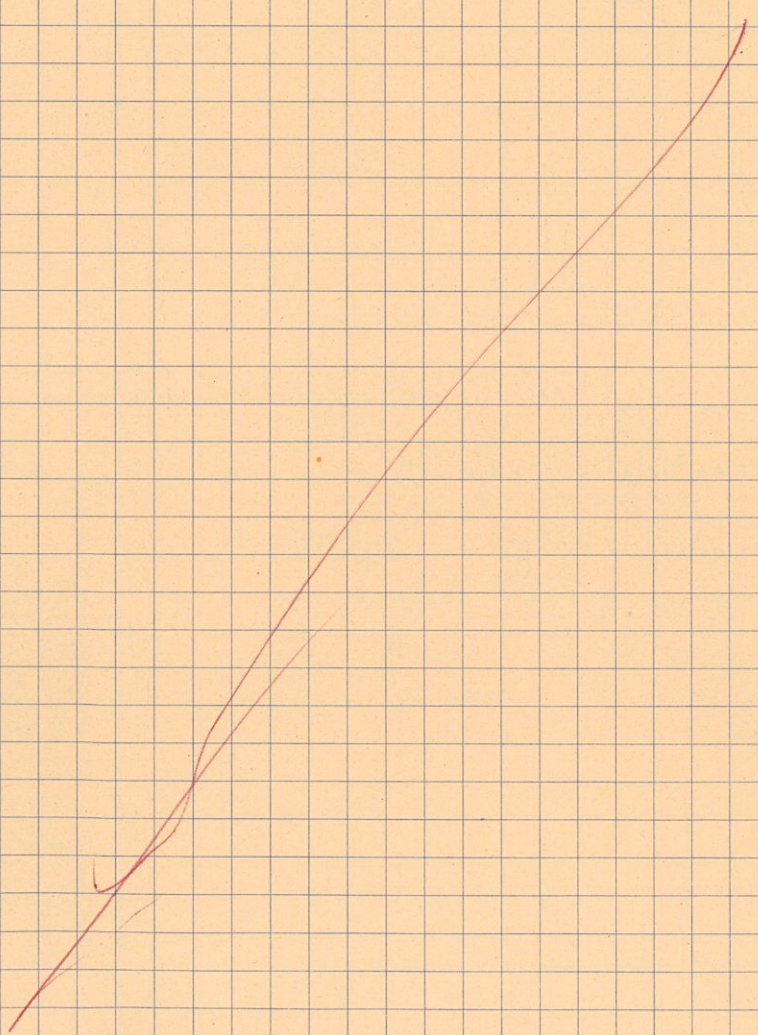
4. Dans les deux cas, des processus ou des tâches peuvent rater leur fenêtre temporelle. Dans le premier cas car l'exécution sera trop lente et les tâches vont s'empiler, il peut donc y avoir une perte d'information. Dans le deuxième cas, car l'exécution sera trop rapide et les tâches n'auront pas le temps de se ~~terminer~~.

1

5 Lorsque nous envoyons trop vite une rafale d'interruption, le système UNIX n'a pas le temps de toutes les traiter et certaines interruptions se perdent. On peut corriger partiellement le protocole en ajoutant un compteur d'interruption. Mais il n'existe pas de réelle solution car si la rafale est trop rapide pour le compteur alors les interruptions ne seront encore pas toutes prises en compte.

4

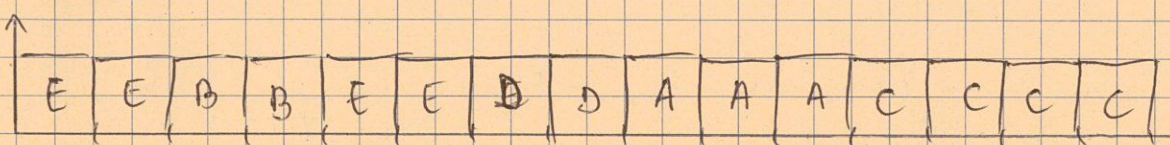
7



2 Scheduling.

8. Ici nous avons la variable temporelle de départ et la variable temporelle de deadline identique : t.
Il est donc difficile de comprendre si la tâche doit s'arrêter à l'instant t ou si t représente C : sa période. Dans le doute je prendrais le fait que les tâches s'arrêtent à l'instant t.
Si non il faut simplement ajouter à la deadline le temps t auquel la tâche à démarré.

1,5



t = 2 : $P_E = 4 - 2 = 6$; $P_B = 9 - 2 = 5$

t = 3 : $P_E = 8 - 3 = 5$; $P_B = 7 - 3 = 4$; $P_C = 15 - 3 = 12$

t = 4 : $P_E = 8 - 4 = 4$; $P_C = 15 - 4 = 11$

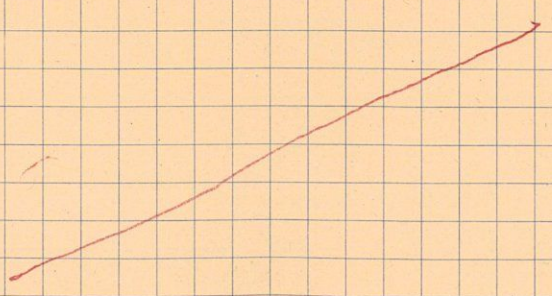
t = 5 : $P_E = 8 - 5 = 3$; $P_C = 15 - 5 = 10$; $P_A = 11 - 5 = 6$

t = 6 : $P_C = 15 - 6 = 9$; $P_A = 11 - 6 = 5$; $P_D = 9 - 6 = 3$

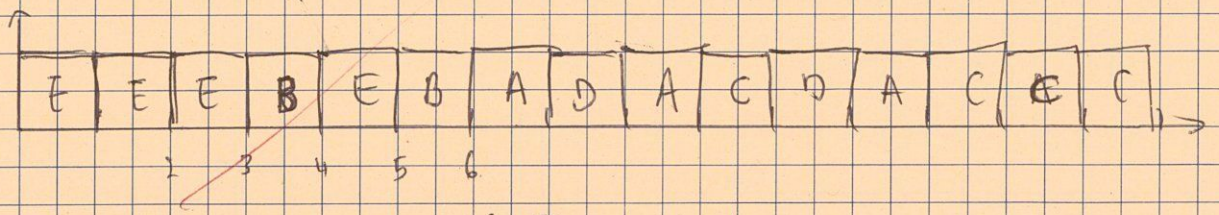
t = 8 : $P_C = 15 - 8 = 7$; $P_A = 11 - 8 = 3$

Dans le cas contraire nous aurons juste E s'exécutant complètement avant B.

9.



→



t=2

$P_E = (9-2)-2 = 4$

$P_B = (9-2)-2 = 5$

t=3

$P_B = (9-3)-2 = 4$

$P_E = (8-3)-1 = 4$

$P_C = (18-3)-4 = 11$

Q5

t=4

$P_E = (8-4)-1 = 3$

$P_B = (9-4)-1 = 4$

$P_C = (18-4)-4 = 10$

t=5

$P_B = (9-5)-1 = 3$

$P_C = (18-5)-4 = 9$

$P_A = (16-5)-3 = 8$

t=6

$P_C = (18-6)-4 = 8$

$P_A = (16-6)-3 = 7$

$P_D = (15-6)-2 = 7$

t=7

$P_C = (18-7)-4 = 7$

$P_A = (16-7)-2 = 7$

$P_D = (15-7)-2 = 6$

t=8

$P_C = (18-8)-4 = 6$

$P_A = (16-8)-2 = 6$

$P_D = (15-8)-1 = 6$

t=9

$P_C = (18-9)-4 = 5$

$P_A = (16-9)-1 = 6$

$P_D = (15-9)-1 = 5$

t=10

$P_C = (18-10)-3 = 5$

$P_A = (16-10)-1 = 5$

$P_D = (15-10)-1 = 4$

NOM Du CUESTA

Prénom Quentin

Promo 2017 M1

Date 07.04.2016

MATIÈRE STR

1a. $W \leq U(n)$

$$W = \frac{2}{5} + \frac{1,5}{4} + \frac{0,5}{3}$$

$$= \frac{4}{10} + \frac{4,5}{12} + \frac{2}{12}$$

$$= \frac{4}{10} + \frac{6,5}{12}$$

$$= \frac{48}{120} + \frac{65}{120}$$

$$= \frac{113}{120} \simeq 0,94 \leq 1$$

1,5

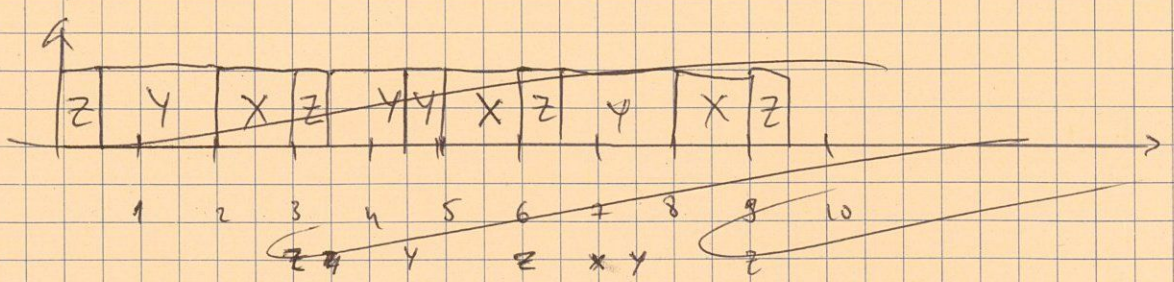
RMS est possible mais pas optimal car

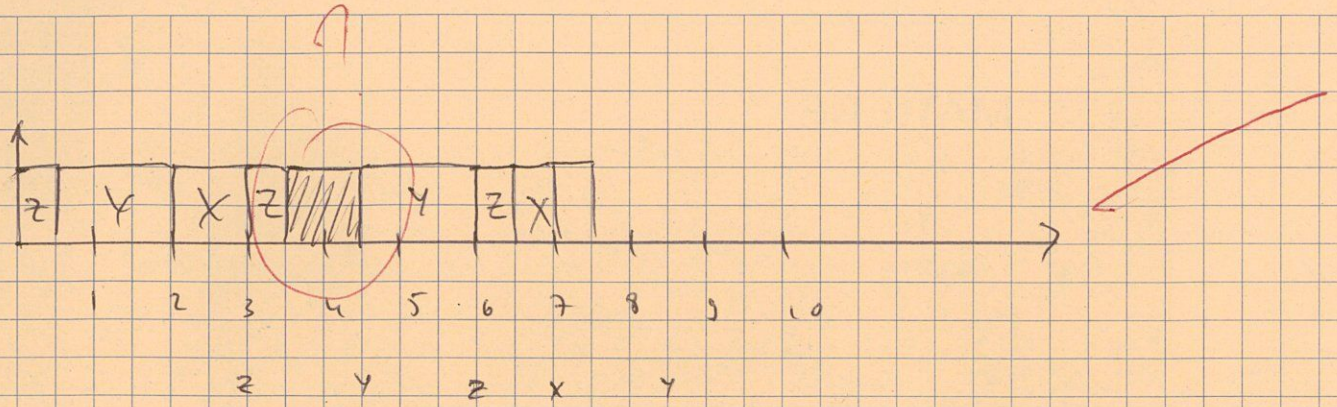
$W > 0,78$ $\Rightarrow W > U(3)$

mais RMS est possible car $W < 1$

4 $P_x = \frac{1}{5}$ $P_y = \frac{1}{4}$ $P_z = \frac{1}{3}$

$P_z > P_y > P_x$





À $t = 7$ nous avons la première deadline manquée.

12. il manque 0,5 second donc il faut augmenter la vitesse du CPU de 20% pourquoi ?

13. EDf et LLF sont optimaux car il ne leur que $W \leq 1$ comme condition. (1)

14. il peut subvenir une situation de deadlock. (1)

15

une seule sémaphore :

A	B
P(sem1)	P(sem1)
V(sem1)	V(sem1)

inverser les sémaphores de B

A	B
P(sem1)	P(sem1)
P(sem2)	P(sem2)
V(sem2)	V(sem2)
V(sem1)	V(sem1)

(1,5)

utiliser un mutex :

A	B
P(mut)	P(mut)
V(mut)	V(mut)

A6.

initialisation

semaphor : occupé = 0 représente le nombres de cases occupés

semaphor : libre = N représente le nombres de case vide

deux threads : Reader() et Writer()

```
Writer()                        Reader()
    P(libre)                        P(occupé)
        write(buffer(data)              read(buffer(data).
    V(occupé)                       V(libre)
```

plusieurs => mutex

Pas besoin de mutex car le Reader ne pourra jamais dépasser le Writer. Si non il faudrait mettre un mutex sur chaque case du buffer.

2

Quentin CUESTA.

3 Synchronization

Exercice 3.1 :

Below the two threads `A` et `B` :

A	B
P(sem1)	P(sem2)
P(sem2)	P(sem1)
...	...
V(sem2)	V(sem1)
V(sem1)	V(sem2)

Question 14. What problem can occur during the parallel execution of the threads `A` and `B` ?

Question 15. Find three different options to solve this problem. What are the new versions of threads `A` and `B` for each option ?

Exercice 3.2 : Read / write buffer

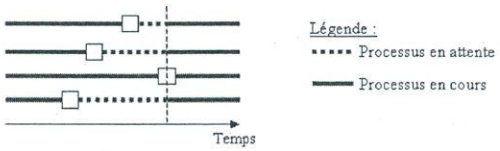
This problem consists of a set of threads and a shared buffer (N elements buffer). Several "writer" threads that write data into the shared buffer (`write_buffer(data)` function) and one "reader" thread that reads data from the shared buffer (`read_buffer()` function). The buffer management is not to achieve, simply use the functions (`write_buffer(data)`) and (`read_buffer()`) in the pseudocode of each thread types.

System constraints :
- Each `Writer()` threads will perform M writes in the buffer (M > N) ;
- The `Reader()` thread will read data from the buffer as long as there in ;
- Reads and writes can be performed in **parallel**.

Question 16. Write pseudocodes of the `Reader( )` and `Writer( )` threads within the constraints of the system. The synchronization between threads will be done with mutex and/or semaphores and the primitves P() and V() (as in the course). **Specify the usefulness of each mutex and semaphore(s) you used.**

Exercice 3.3 : Synchronization barrier for N threads

A synchronization barrier ensures that several threads have reached a particular moment. Thus, there are several threads that run in parallel and they need to wait for the last of them to finish (one section of the code) before they can all move on. The figure below illustrates the operation of this synchronization barrier (or rendez-vous) for 4 threads.



Question 17. Write the code of a synchronization barrier mechanism between N threads using mutex and/or semaphores and the primitves P() and V() (as in the course). Fill the empty cells in the program given in figure 17.

Notes : – Note that some of the empty cells can be remain empty and some other(s) can contain multiple lines of code if needed ;
– La global variable « `nb_process` »represents the remaining number of threads to be synchronized (i.e. the number of threads that have not reached the barrier). It is initialized to N.

```
void barriere(void)
{
    int i = 0;
    P( mutex )
    nb_process--;
    
    if(nb_process == 0)
    {
        for(i=0 ; i<N ; i++)
        {
            move_on()
        }
        nb_process = N;
    }
    V( mutex)
}
```