



NOM BERTRAND  
 Prénom Alexandre  
 Promo p2017  
 Date .....

M1 - 2015



		1	3		
--	--	---	---	--	--



BERTRAND Alexandre  
 M1 - 2015

**MATIÈRE** Real time systems

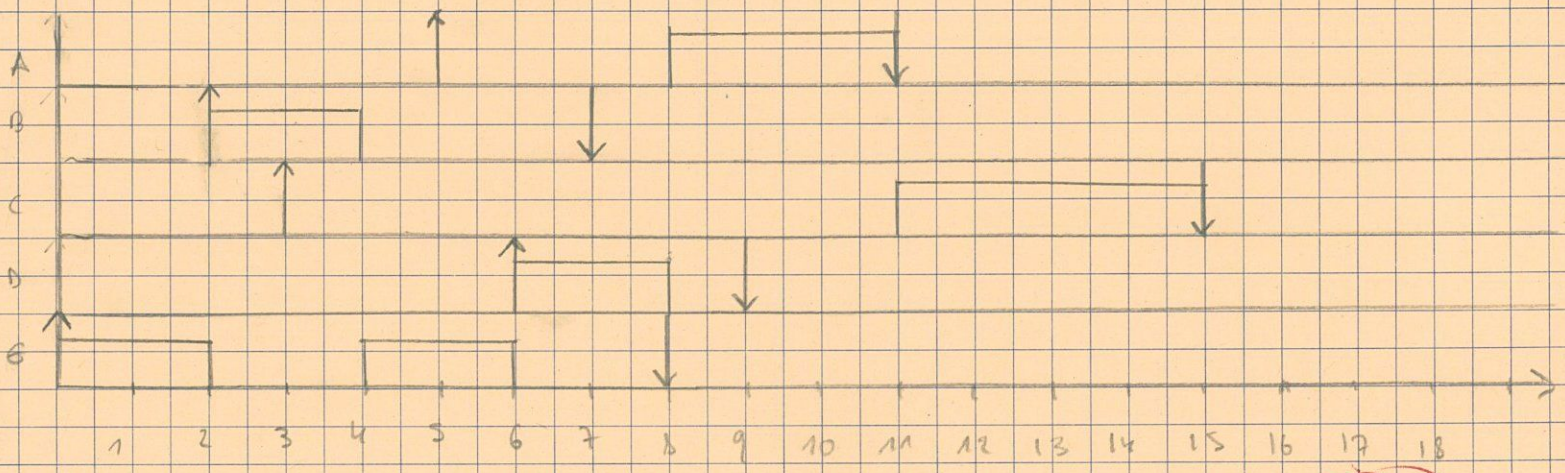
14/24

2. Scheduling

Question 8 : EDF

priority =  $\frac{1}{\text{deadline}}$       $P_A = \frac{1}{11}$  ;  $P_B = \frac{1}{7}$  ;  $P_C = \frac{1}{15}$  ;  $P_D = \frac{1}{9}$  ;  $P_E = \frac{1}{8}$

$P_B > P_E > P_D > P_A > P_C$

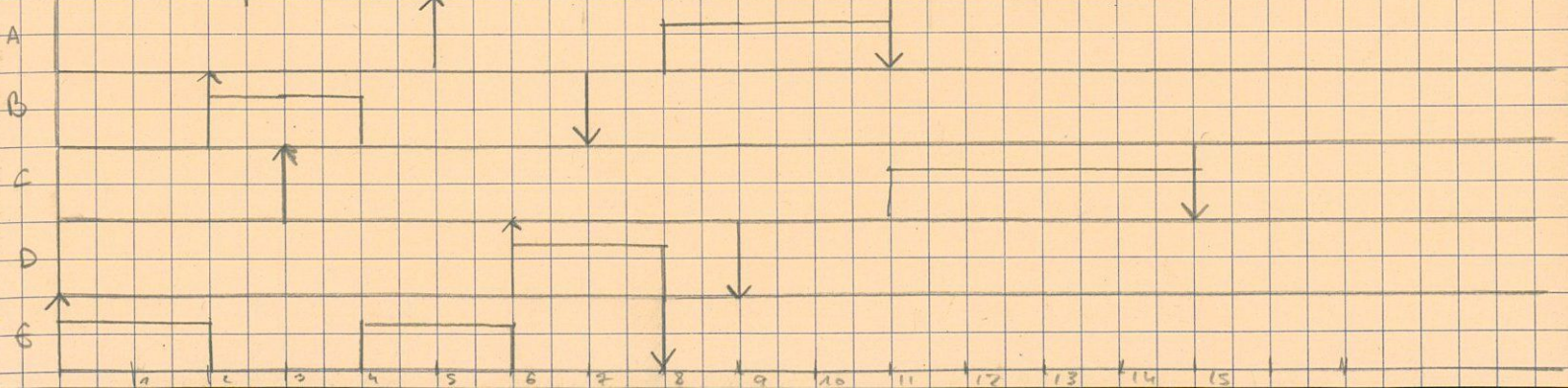


3

Question 9 : LLF

priority =  $\frac{1}{\text{laxity}}$       $M = (t_f - t_c) - R$

t = 0s	t = 2s	t = 3s	t = 5s	t = 6
	$M_B = 7 - 2 - 2 = 3$	$M_B = 7 - 3 - 1 = 3$	$M_A = 11 - 5 - 3 = 3$	$M_A = 11 - 6 - 3 = 2$
prio E	$M_E = 8 - 2 - 2 = 4$	$M_C = 15 - 3 - 4 = 8$	$M_C = 15 - 3 - 4 = 6$	$M_C = 15 - 6 - 4 = 5$
	$P_B > P_E$	$M_E = 8 - 3 - 2 = 3$	$M_E = 8 - 5 - 1 = 2$	$M_D = 9 - 6 - 2 = 1$
		$P_B > P_E > P_C$	$P_C > P_A > P_C$	$P_D > P_A > P_C$



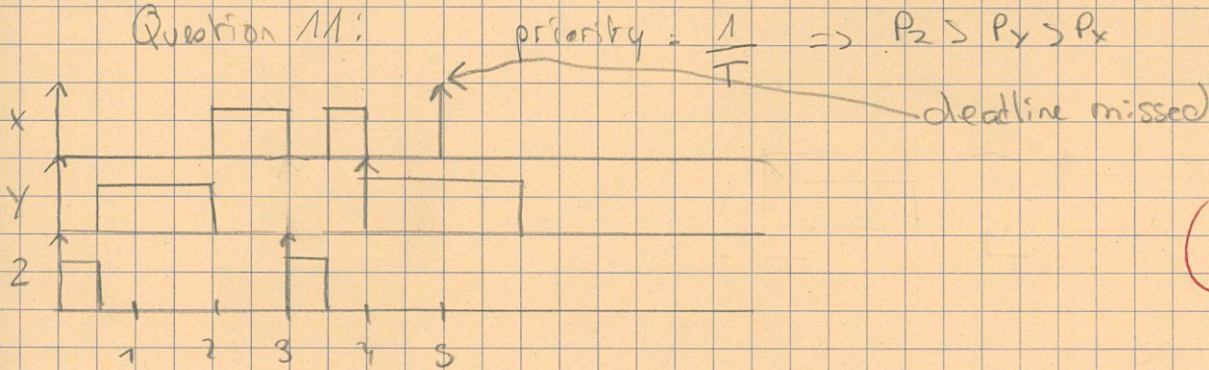
Question 10:

$$W = \frac{2}{5} + \frac{1,5}{4} + \frac{0,5}{3} = \frac{8}{20} + \frac{7,5}{20} + \frac{0,5}{3} \\ = \frac{15,5}{20} + \frac{0,5}{3} = \frac{46,5}{60} + \frac{10}{60} = \frac{56,5}{60}$$

$W \leq 1$   
 $W \geq U(s)$  (condition satisfied) *maybe*  
 $\Rightarrow$  the RMS is not optimal, there is a solution but some deadlines won't be respected.

④

Question 11:



①

Question 12:

the X miss  $\frac{0,5s}{5s} = 10\%$ . the CPU need to go at least 10% faster

Question 13:

$W \leq 1$  so they should be optimal but here the deadline is the end of the period so the EDF will miss deadline too.

①

### 3. Synchronization

Question 14: A and B can block each other.

A use sem 1 and need sem 2  
 B use sem 2 and need sem 1 => blocked

1

Question 15:

inverse sems in B or A

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

1

use a mutex

A	B
P(mutex)	P(mutex)
...	...
V(mutex)	V(mutex)

Question 16

We need 2 semaphores  
 lib = N  
 occ = 0  
 and a mutex

Writer()	Reader()
P(lib);	P(occ)
P(mutex);	read_buffer();
write_buffer(data);	V(occ);
V(mutex);	
V(lib);	

3

Question 17 voir feuille

Question 1:

A real time system is a system that can handle asynchronous task from real elements. **time?**

Question 2: hard real time mode is a mode that doesn't allow to tasks to exceed the windows scheduled for that task. and the soft mode allow it.

Question 3:

Determinisme: Quand l'exécution est toujours la même

Predictability: l'exécution varie et le programme doit prendre en compte toutes les possibilités

reliability: Quand le programme est maintenable et qu'il ne perd/deteriore pas les ressources.

0,5

1

Question 4:

et donc ?

with a less efficient CPU: The loop will take a longer time  
with a more efficient CPU: The loop will take less time.

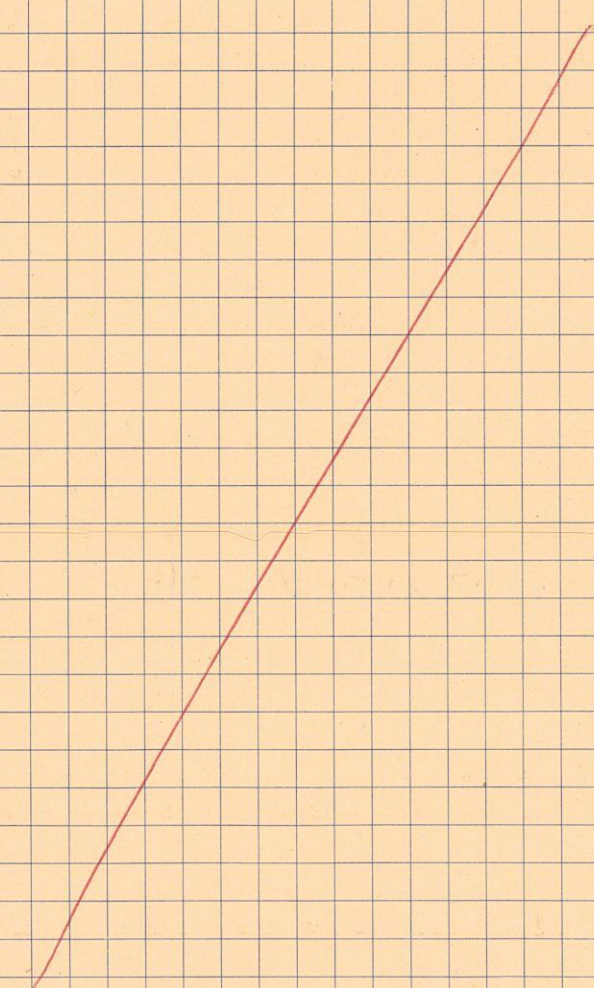
Question 7:

It's when a task with a low priority hold a resource and this task is interrupted by a task with a higher priority that need the same resource, but cannot use this resource because the other task hasn't release the resource. To resolve this problem we can implement the priority inheritance: when the task is interrupt it take the priority of the second task that way, it finish with the resource and release it.

Question 5:

The handler can be overtaken and will not execute every interruption.

Q3



## Real-time systems (1h45)

M1 IL 2015-2016

Les documents de cours ne sont pas autorisés, sans calculatrice

### 1 Course questions

**Question 1.** Give the definition of a real-time system.

**Question 2.** Explain the concepts of "hard real-time" and "soft real-time" ?

**Question 3.** Give the definitions and the french translation of the terms : determinism, reliability and predictability.

**Question 4.** What problem can occur in the case of loop programming if we change to a less efficient processor (only in term of execution time) ? And with a more efficient processor ?

**Question 5.** What problem can appear when sending a burst of interruptions on a UNIX system ? Why are there no "real" solution to this problem ?

**Question 6.** What the WCET and what is it for ?

**Question 7.** What is "priority inversion" problematic scenario and what is the "priority inheritance" protocol ?

## 2 Scheduling

Reminders -  $W = \sum \frac{C}{T}$ ;  
-  $U(1)=1$ ,  $U(2)=0.83$ ,  $U(3)=0.78$  ( $=46.8/60$ ).

### Exercice 2.1 : Sporadic tasks scheduling

A system must perform the following 5 tasks :

- $A$  takes 3 seconds, is ready at  $t = 5$  and its deadline is at  $t = 11$ ;
- $B$  takes 2 seconds, is ready at  $t = 2$  and its deadline is at  $t = 7$ ;
- $C$  takes 4 seconds, is ready at  $t = 3$  and its deadline is at  $t = 15$ ;
- $D$  takes 2 seconds, is ready at  $t = 6$  and its deadline is at  $t = 9$ ;
- $E$  takes 4 seconds, is ready at  $t = 0$  and its deadline is at  $t = 8$ .

**Question 8.** Give the scheduling with EDF. Indicate priorities between tasks. When the priorities of two tasks are equal, choose : A to B, B to C, etc.

**Question 9.** Give the scheduling with LLF. Indicate priorities between tasks on each time interval (and the calculations to get it). When the priorities of two tasks are equal, choose : A to B, B to C, etc.

### Exercice 2.2 : Periodic scheduling

Set of periodic jobs  $X$ ,  $Y$  et  $Z$  :

- $X$  takes 2 seconds every 5 seconds;
- $Y$  takes 1.5 seconds every 4 seconds;
- $Z$  takes 0.5 seconds every 3 seconds.

All jobs are ready at  $t = 0$ .

**Question 10.** Is the RMS scheduling policy is optimal? Is there a solution according to RMS? Justify your answers. No scheduling is requested to answer to these questions.

**Question 11.** Give the CPU scheduling with RMS policy (until the first missing deadline).

**Question 12.** Determine the minimum CPU speed increase allowing RMS to become optimal. No scheduling requested.

**Question 13.** Are EDF and LLF optimal? No scheduling requested.

### 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.

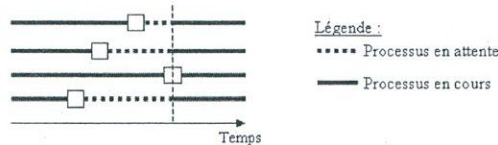
##### System constraints :

- 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 primitives `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 primitives  $P()$  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 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--;
    P(nb)
    if(nb_process == 0)
    {
        for(i=0 ; i<N ; i++)
        {
            V(mutex)
        }
        nb_process = N;
    }
    V(nb)
}
    
```