Tutorial Session N°2: Events and Locks

Objective: Learn how to control and coordinate multiple threads using Events and Locks, so threads can communicate safely and avoid conflicts when accessing shared resources.

Exercise 1

Two processes, P1 and P2, where P1 has higher priority than P2, share a critical resource. The scheduling rules ensure that P1 is executed as soon as it becomes ready (higher priority first).

While P1 is performing I/O, process P2 enters the critical section. Then, process P1 becomes ready while P2 is still in the critical section. Process P2 is then suspended in favor of process P1. P1 requests to enter the critical section.

1. What happens if we use the TAS instruction?
2. Give a solution using events.

Exercise 2

The most common situation of cooperation between processes occurs when data is transmitted from one process to another. Let's consider the simplest case: A keyboard process reads a character typed on a terminal keyboard, and transmits it to a screen process, which displays it on the same terminal.

```
var c : char;   /* shared variable */
```

Process keyboard	Process screen
begin	begin
char cr;	char cw;
while (1) {	while (1) {
read a character from keyboard into cr;	cw = c; /* instruction 2 */
c = cr; /* instruction 1 */	write the character cw to the screen;
}	}
end;	end;

The screen process must not execute instruction 2 before the keyboard process has executed instruction 1 (i.e., before a new character is available). Conversely, the keyboard process must not execute instruction 1 again (produce a new character) before the screen has finished executing instruction 2 for the previous character.

1. Using events, synchronize the two processes
2. Repeat the first question with Locks synchronization mechanism
3. If we have one keyboard and 3 screens, the keyboard process reads a character, and transmits it to the 3 screen processes, which display it at the same time. In this case what is the suitable synchronization mechanism to apply events or locks? Justify your answer.

Exercise 3

Two processes **P1** and **P2** share an integer variable **x**, which is initially equal to **zero (0)**. Each process increments the variable **x** **five (5) times**. The two processes execute their instructions **concurrently without any synchronization mechanisms**.

1. Explain how it is possible that, even though there are ten increment operations in total, the final value of **x** could be **3**.
2. What are all the possible final values of **x** after both processes have finished executing?
3. Give a solution to this problem when the variable **x** is protected using a **lock synchronization mechanism**.

Good Luck