Software Synchronization

Race Conditions

- A **race condition** is a situation where **several processes or threads** attempt to **manipulate the same data**, and the **outcome** of the execution **depends on the particular order the access takes place**.
- The **section of code** where a **process or thread** accesses a **shared resource** is referred to as the **critical section**.
- To **prevent race conditions**, we use **synchronization**. This ensures that no more than **one process** is able to execute it's **critical section** at a time.

Syncronization Solution Requirements

- Any **solution** to **synchronization** should satisfy the following requirements:
  1. The solution should have **mutual exclusion (mutex)**: Only **one process** can execute it's **critical section** at any given time.
  2. The solution should have **progress**: When **no process** is currently in a **critical section**, **any process** that **requests entry** into the **critical section** must be **permitted without delay**.
  3. The solution must **prevent starvation (bounded wait)**: There is an **upper bound** on the number of times a **process enters the critical section** while **another is waiting**.

  - Such a solution will prevent race conditions.

Synchronization Solutions

- A **simple** way to **achieve synchronization** is the use interrupts; when **interrupts are disabled, context switches will not happen**.
  - This is not a good solution, becuase user processes generally cannot disable interrupts.
  - This also does not work on a multi-core system.
- Another way to **achieve synchronization** is by using a **variable as a flag** to **determine** if any other processes are executing their **critical section**. When a process wants to **execute their critical section** and another process has already locked the resource, the process will use a **conditional loop until the lock is removed**.
  - This acheives mutual exclusion, but wastes a lot of processor time.
  - This also does not prevent starvation.
- **Modern computers** use **hardware solutions** to **synchronize processes**.

Hardware Syncronization