Three Control Problems with Competing Processes:

Mutual Exclusion: Ensures that only one process at a time can access a critical section (a part of the code that accesses shared resources) to prevent data inconsistency.

Deadlock: Occurs when two or more processes are stuck in a state where each is waiting for the other to release a resource, leading to a situation where none can proceed.

Starvation: When a process is perpetually denied access to resources or critical sections because other processes keep gaining access first, leaving it "starved" of resources.

Requirements for Mutual Exclusion:

Only one process can be in the critical section at any time.

Progress: If no process is in the critical section, one should be able to enter if it's waiting.

Bounded Waiting: A process should have a limit on the time it must wait to enter the critical section.

Operations on a Semaphore:

Wait (P): Decrements the semaphore. If the result is negative, the process is blocked until it becomes positive.

Signal (V): Increments the semaphore. If there are blocked processes, one is awakened.

Monitor: A synchronization construct that encapsulates shared variables, the procedures that operate on them, and the synchronization between those procedures. Only one process can execute a monitor procedure at a time, which ensures mutual exclusion within the monitor.

Busy Waiting vs. Blocking Wait:

Busy waiting can be less efficient than a blocking wait as it consumes processor time by repeatedly checking a condition in a loop. In contrast, a blocking wait releases the processor for other tasks until the condition is met. However, busy waiting can be useful if the wait time is very short or in a real-time system where responsiveness is crucial, as blocking might add overhead for waking the process back up.