# PL5 - Semaphores

Luís Nogueira, Orlando Sousa

{lmn, oms}@isep.ipp.pt

Sistemas de Computadores
2024/2025

**Note: All exercises must use semaphores for the correct synchronization between processes. The use of active (busy) waiting is discouraged.**

1. Develop a program that spawns 5 new processes with the following specifications:
   - Each process is tasked with writing 200 numbers into a text file.
   - Only one child process should have write access to the file at any given time.
   - Upon completion, the parent process should display the contents of the file.
2. Modify the previous program to ensure that each child process writes to the file sequentially, according to its process number ranging from 1 to 5.

   Note: Utilize five semaphores for synchronization among child processes.
3. Develop a program that takes the name of a file as a parameter and operates as outlined below:
   - Implement mutual exclusion for file access.
   - Append to the file a line of text containing the following information: "I am the process with PID `X`", replacing `X` with the PID of the process.
   - Pause execution for 2 seconds.
   - Release file access.

   Test your program by running multiple concurrent instances to ensure its functionality.

4. Modify the previous program by adding the following features:
   - Restrict the file to a maximum of 10 lines.
   - Integrate an option to delete a line from the file.
   - Implement a mechanism where the program waits a maximum of 10 seconds to access the file, and notify the user if this threshold is exceeded.

5. Develop a program that spawns a new process and utilizes semaphores to synchronize its actions with its parent.
   - The child process displays the message "I am the child" on the screen.
   - The parent process displays the message "I am the parent" on the screen.
   - Ensure that the parent process is the first to display its message on the screen.

6. Enhance the previous program to incorporate the writing of 10 messages, alternating between the parent and child processes.

7. Take into account the operations of the child processes illustrated below:

| Child 1 | Child 2 | Child 3 |
|---|---|---|
| `Sistemas` | `de` | `Computadores` |
| `is` | `the` | `best!` |

   Implement operations on semaphores such that the printed result is: "Sistemas de Computadores is the best!"

   Note: Use `fflush(stdout)` after every `printf()` so the output is not buffered.

8. Develop a program to simulate the ticket-selling operation, incorporating the following considerations:
   - Only the `seller` has access to the tickets, and thus, is aware of the next ticket number;
   - Clients must be attended to in the sequence of their arrival, waiting until they receive their ticket;
   - Upon receiving a ticket, a client should print its ticket number.

   Develop separate programs for the behaviors of clients and the seller.

   Ensure synchronization of actions using semaphores and facilitate data exchange through a shared memory area. Additionally, each client should experience a random service time ranging from 1 to 10 seconds.

   For an optimal simulation, attempt to execute multiple clients concurrently.

9. A financial institution aims to create an application enabling users to manage banking transactions such as deposits and withdrawals. To achieve this goal, the following programs need to be developed:

- `withdraw` - This program facilitates users in withdrawing money from their account, ensuring they have sufficient funds.
- `consult` - This program enables users to review the balance in their account.
- `server` - This program prioritizes client requests based on arrival time, ensuring each client exclusively accesses their respective bank account.

Note: The application should seamlessly support multiple concurrent clients.

10. Consider the following two processes:

| P1 | P2 |
|---|---|
| `buy_chips()` | `buy_beer()` |
| `eat_and_drink()` | `eat_and_drink()` |

Add semaphores to ensure that neither P1 nor P2 eat or drink until both the beer and chips are acquired.

11. Modify the previous program to:
    - Add 4 more processes that will also execute (randomly) `buy_chips()` or `buy_beer();`
    - The 6 processes can only execute `eat_and_drink()` when all other processes have finished acquiring the chips and the beer.

    Be careful to produce adequate output that clearly shows the execution of the processes.

12. Develop a program that begins by spawning two new processes. Each child process will function as a `producer`, while the parent process will serve as a `consumer`.

    These processes interact via a circular buffer housed in a shared memory area, capable of holding 10 integers. Each `producer` process inserts incrementing values into the buffer, which the `consumer` subsequently reads and prints.

    It is assumed that only 30 values are exchanged throughout the lifetime of the program, with access to the buffer being regulated by a semaphore:
    - `producer` processes will be blocked if the buffer reaches its capacity.
    - `producer` processes will write to the buffer only after ensuring mutual exclusion.
    - The `consumer` process will be blocked until new data becomes available to read.

13. Develop the following set of programs:
    - `reader` - responsible for reading a `string` into a shared memory area:
      - Readers refrain from modifying the shared memory area, enabling multiple readers to access it concurrently.
      - Readers are prioritized over writers.

- Each reader should print the retrieved `string` along with the current count of readers.

- `writer` - tasked with writing to the shared memory area:
  - Writes its Process ID (PID) and the current time.
  - Only one writer can access the shared memory area at any given time.
  - Writers can access the shared memory area only when there are no active readers.
  - Each writer prints the count of writers as well as the count of readers.

To verify the effectiveness of the software, execute multiple instances of readers and writers simultaneously.

14. Enhance the previous program to incorporate the following modifications:
    - Restrict the number of concurrent readers to 5.
    - Prevent readers from accessing the shared memory area when more than 2 writers are waiting.

15. Imagine a public service facility equipped with 3 desks, each represented by a server process, catering to clients. The system operates according to the following principles:
    - Whenever server processes are available, they should promptly attend to client processes. The duration of attending to a client ranges from 1 to 10 seconds.
    - If no client processes are present, the server processes should remain in a blocked state.
    - A maximum of 10 clients are allowed to wait at any given time.

16. Create a program to facilitate an application supporting multiple `producer` processes, each responsible for transmitting messages to a range of `consumer` processes, defined by the following characteristics:
    - The messages are placed in 4 queues, each associated with a priority level.
    - Assume that all messages are of the same size;
    - Assume that the message queues have space for 3 messages
    - Assume that the messages should be consumed in priority order.

17. Design a program to manage a showroom and a group of visitors, observing the following guidelines:
    - Each visitor is represented by an infinite loop, simulating their entry and eventual departure from the showroom.
    - Showroom events commence and conclude at predetermined times, allowing visitors to enter and exit during these intervals.

- The showroom accommodates a maximum of 5 visitors at any given time. If additional visitors attempt to enter, they must wait until space becomes available.

18. Implement a simulation reflecting the dynamics of a pool hall featuring $N$ tables. A table is considered occupied from the moment two players occupy it until both players depart. Upon arrival, a player can join any available table that isn't occupied, preferably selecting one with a player already prepared to commence a game.

19. Consider a bridge with only one lane that permits traffic in both East-West directions, adhering to the following rules:

    - A car may access the bridge only if it is unoccupied by vehicles traveling in the opposite direction.
    - If the bridge is occupied in the opposite direction, the car must wait until traffic permits.

    It is assumed that a car requires 1 minute to traverse the bridge. The permitted direction of traffic on the bridge shifts each time the last car completes its crossing.

20. Envision a parking lot with a maximum capacity of MAX cars, accommodating three distinct user types: VIP, Special, and Normal.

    - As long as there are vacant parking slots, cars enter the parking lot in the sequence of their arrival.
    - Once the parking lot reaches full capacity, the entry of a new car is contingent upon the departure of others. Priority is assigned to waiting VIPs, followed by Specials, and lastly Normals (in that precise order).