Khaula Molapo

Operating Systems Week 6

1.

In a database system, Inter-Process Communication (IPC) ensures different processes work together efficiently. For example, consider a client-server database setup. When a client submits a query, the request is handled by a client process, which must communicate with the database server process. IPC provides the mechanism to transfer this data safely and quickly. Shared memory may be used for storing query results, while semaphores control access, preventing data corruption when multiple clients request information simultaneously. Message queues or sockets can handle structured communication, allowing different machines to exchange data reliably. This approach ensures that multiple client processes can interact with the central database concurrently without interfering with each other's transactions. By implementing IPC mechanisms, the database system achieves faster response times, improved concurrency, and better resource utilization. Without IPC, processes would remain isolated, leading to inefficiency, duplication of effort, and delays in handling large numbers of database queries.

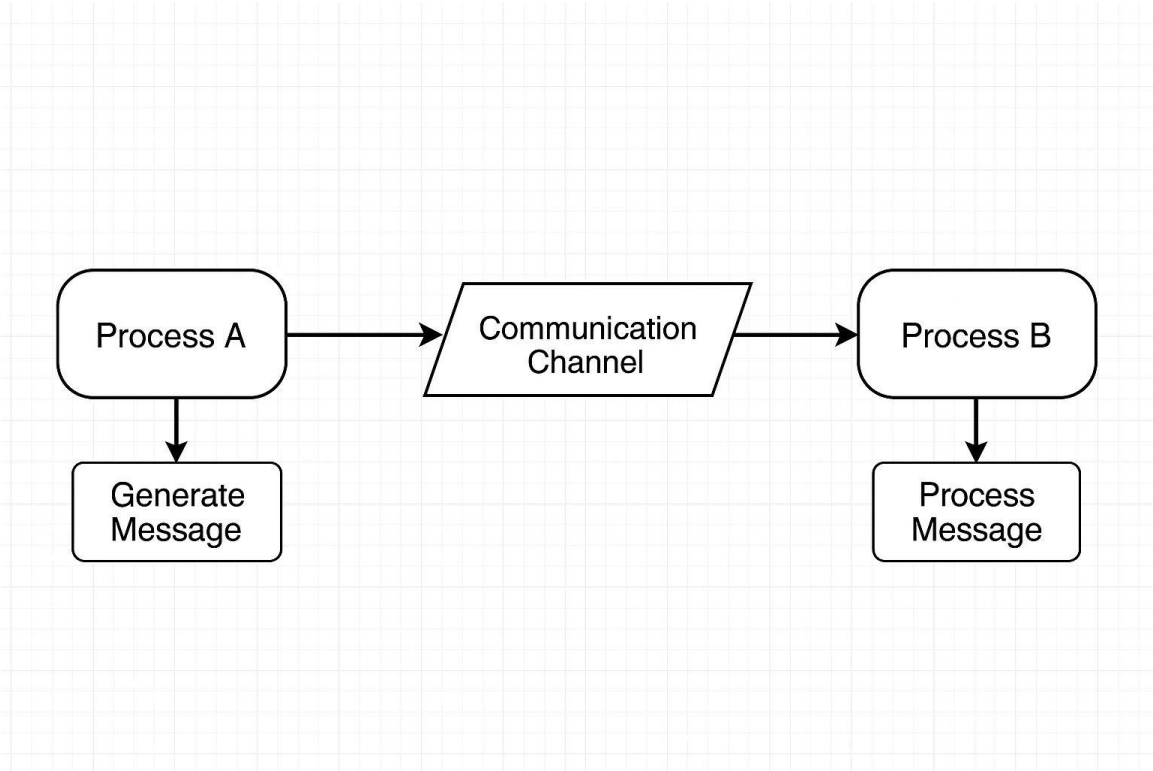
2.

Message passing is a key IPC technique where processes exchange data by sending and receiving messages, instead of sharing memory. Each process operates independently, and communication happens through a controlled channel such as sockets, message queues, or pipes. Its main role is to provide synchronization and coordination among processes, especially in distributed systems where shared memory is impractical. Message passing ensures data integrity by preventing direct memory conflicts and allows processes to work across networks. It simplifies communication by abstracting the underlying hardware details. Overall, message passing supports modular, secure, and scalable system design in both local and distributed environments.

3.

In Draw.io, I created a simple message passing flowchart. The flow begins with Process A generating a message, which is placed into a communication channel. The channel

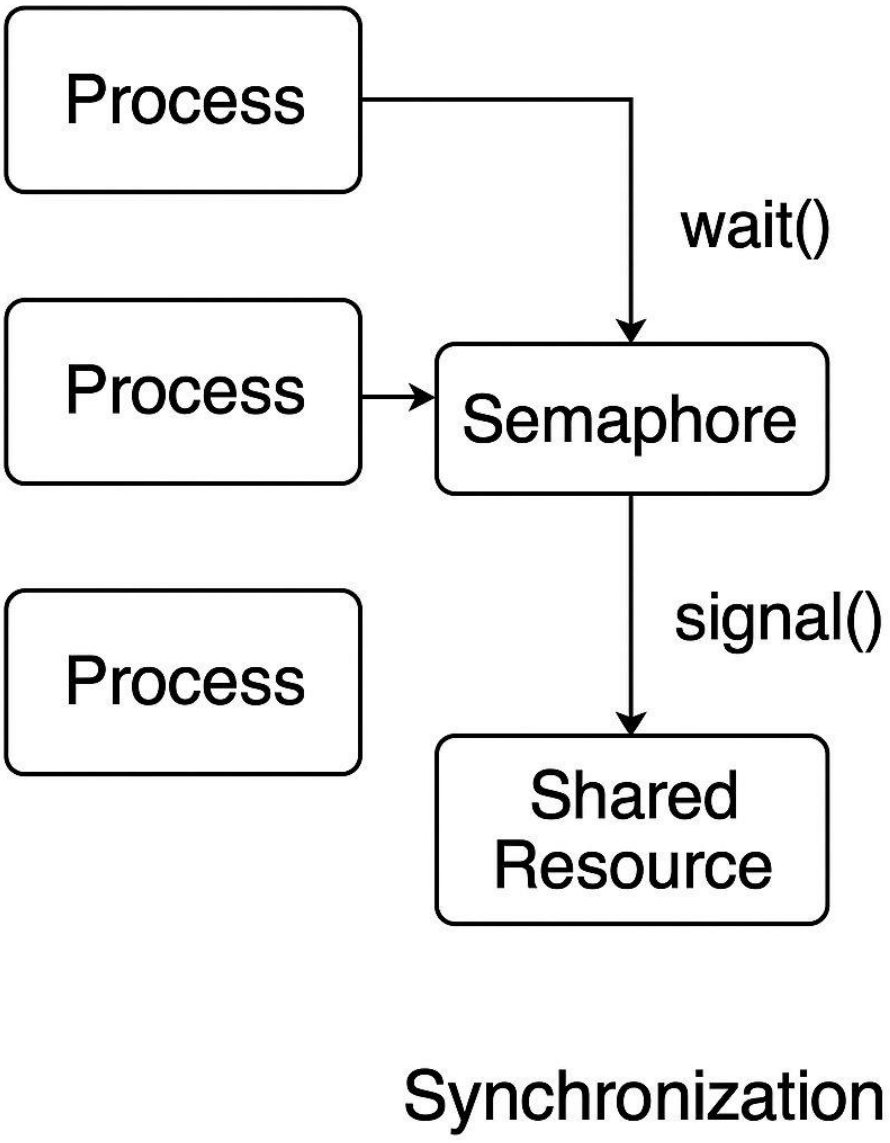
represents a queue or socket that ensures safe delivery. Next, Process B retrieves the message and processes it. This shows how IPC ensures data moves reliably between independent processes. While designing, I reflected on how the flow mirrors real-world distributed applications, like a chat system where one user sends a message and another receives it. This exercise demonstrated how message passing decouples processes, making systems easier to manage, debug, and scale effectively.



4.

In Draw.io, I designed a diagram of a synchronization mechanism using semaphores. The diagram shows multiple processes attempting to access a shared resource, such as a file. A semaphore acts as a control lock, with two operations: wait() and signal(). When a process wants access, it executes wait(); if the resource is available, access is granted. Once finished, the process calls signal(), releasing the lock for others. This ensures mutual exclusion, preventing race conditions where two processes might write simultaneously.

The diagram highlights the importance of synchronization in IPC, ensuring consistent results, fairness, and stability in concurrent system operations.



Synchronization