Esquemas de sincronización entre procesos

Dependiendo el modo en que se implementan las primitivas de comunicación vistas en la presentación, existen diversos esquemas de sincronización entre procesos

- **Primitivas bloqueantes:** el proceso que las invoca queda bloqueado hasta que se verifica una condición (es decir, cuando la primitiva se completa).
- **Primitivas no bloqueantes:** el proceso que las invoca continúa su ejecución.

Ejemplos:

- ENVIAR bloqueante: el proceso emisor queda en espera hasta que el receptor reciba correctamente el mensaje (receptor completa la correspondiente primitiva RECIBIR)
- ENVIAR no bloqueante: el proceso emisor “coloca” dos datos en una zona de envío (buffer) y continúa su ejecución.
- RECIBIR bloqueante: el proceso receptor queda en espera hasta que haya datos disponibles del emisor que corresponda (emisor completa la correspondiente primitiva RECIBIR)
- RECIBIR no bloqueante: el proceso receptor recoge los datos disponibles, si no los hay lo indica y continúa su ejecución.
- INICIAR CONEXIÓN bloqueante: el proceso que inicia la conexión queda a la espera hasta que el otro extremo invoque el correspondiente ACEPTAR CONEXIÓN.
- ACEPTAR CONEXIÓN bloqueante: hace que el proceso quede en espera hasta que desde el otro extremo se invoque un INICIAR CONEXIÓN.

Casos típicos:

- Comunicaciones síncronas: ENVIAR y RECIBIR bloqueantes
    - Se garantiza sincronización entre los mensajes enviados y recibidos
    - Simplifica la programación, pero penalizan el rendimiento
- Comunicaciones asíncronas: ENVIAR no bloqueante
    - Evitan bloqueos indefinidos y ofrecen mejor rendimiento
    - RECIBIR puede ser bloqueante (lo más usual) o no bloqueante
        - Implementación compleja con ENVIAR y RECIBIR no bloqueantes

- Diseño habitual: ubicar las operaciones bloqueantes en un hilo de ejecución (thread) propio
    - Ejemplo: servidores multihilo, un hilo para esperar y otro para atender cada petición.