Ejercicios de finales ordenados por su tipo, delante del ej. Está escrito de qué final salió.

1. Algoritmos de Sincronización y Modelos de Concurrencia

Fork-Join y Work Stealing

- **[2do-2C 2023] Ejercicio 2:** Explicar el modelo de concurrencia Fork-join. Explicar algoritmo de work stealing. Explicar y justificar por qué no es conveniente tener una única cola de tareas en dicho algoritmo.

- **[2do-2C 2024] Punto 4:** Explicar fork join, work stealing y por qué conviene usar work stealing en vez de una única cola de procesos.

- **[2do-1C 2025] Punto 1:** Fork-join: definición, definir work stealing y explicar por qué es mejor implementar work stealing a que tener una sola cola de tareas compartida.

Programación Asincrónica (Async)

- **[2do-2C 2024] Punto 2:** Explicar Async, comparar con threads y dar ventajas y desventajas de c/u, ejemplo de uso de c/u.

- **[5to-2C 2024]:** Explicar programación asincrónica, diferencias con threads y ejemplos en que convenga más uno y el otro.

Actores

- **[3ero-2C 2024] Punto 3:** a. Explicar modelo de actores y dar sus componentes. b. Explicar qué es el *arbiter*. c. Explicar qué es el contexto.
- **[2do-1C 2025] Punto 3:** Actores: Explicar cuál es la motivación del modelo, características detalladas de la entidad actor y ciclo de vida de un actor de Actix.

Semáforos y Monitores

- **[1er-2C 2022]:** Monitores (y volatile).
- **[5to-2C 2024]:** Explicar qué es un monitor y en qué se diferencia de un semáforo.
- **[3ero-1C 2025] Punto 1:** Semáforo, sus funciones y variables internas.

Comparativa General de Modelos

- **[3ero-2C 2024] Punto 2:** a. Comparar los modelos de exclusión mutua, fork-join, mensajes y actores. b. Dar ventajas y desventajas c/u. c. Dar ejemplos de uso de c/u y por qué convienen.

---

2. Problemas de Concurrencia (Deadlocks y Propiedades)

Propiedades de Corrección

- **[2do-2C 2024] Punto 1:** Explicar las propiedades Safety y Liveness, y ejemplos de propiedades asociadas a estas.
- **[3ero-2C 2024] Punto 1:** a. Definir el problema de la Sección Crítica (SC). b. Decir las 3 propiedades de corrección que debe cumplir.
- **[1er-1C 2025] Punto 2:** Describir y comparar los efectos negativos que tienen en los programas concurrentes: Deadlocks, Race conditions, Starvation, Busy wait.

Deadlocks (Definición, Detección y Prevención)

- **[1er-2C 2022]:** Detección de deadlock (exclusión mutua también).

- **[2do-2C 2023] Ejercicio 1:** Qué es un deadlock, explique sus consecuencias en un programa concurrente. Mencione qué complicaciones adicionales tienen los deadlocks en ambientes de concurrencia distribuida. Explique a detalle mecanismos de prevención de deadlocks en ambientes distribuidos.
- **[2do-2C 2024] Punto 3:** Explicar deadlocks, explicar algoritmos distribuidos de detección de deadlocks, y graficar un ejemplo del algoritmo distribuido.
- **[5to-2C 2024]:** Explicar el algoritmo de detección de deadlocks distribuido, y diferencias entre prevención y detección de deadlocks.
- **[1er-1C 2025] Punto 3:** a. Explicar el algoritmo de detección de deadlocks distribuido. b. Diagramar un ejemplo de detección de deadlock y explicarlo paso a paso. c. Explicar cómo se puede evitar esto (prevención: wait-die y wound-wait).

---

3. Sistemas Distribuidos

Conceptos Básicos y Entidades

- **[1er-2C 2022]:** Qué es entidad y sus capacidades. Conocimiento distribuido.
- **[3ero-1C 2025] Punto 4:** Regla, acción, comportamiento y conocimiento de entidades.
- **[1er-1C 2025] Punto 4:** a. Enumerar posibles eventos en un ambiente distribuido. b. Explicar cómo se calcula el costo y complejidad en un ambiente distribuido y comparar el cómputo que hay que hacer en un sistema centralizado.
- **[2do-1C 2025] Punto 4:** Entidades: Qué son, capacidades y restricciones temporales.

Exclusión Mutua Distribuida y Elección de Líder

- **[1er-2C 2022]:** Elección del líder (Bully y Ring).
- **[2do-2C 2023] Ejercicio 4:** Explicar el algoritmo Ring para elección de líder. Realizar diagramas que detallen paso a paso el algoritmo.
- **[3ero-1C 2025] Punto 3:** Exclusión mutua en ambiente distribuido, diagrama y problemas.

---

4. Transacciones y Control de Concurrencia

- **[1er-2C 2022]:** Precondiciones de transacciones (storage estable, tiempos de comunicación finitos).
- **[3ero-2C 2024] Punto 4:** a. Explicar el commit en 2 fases, dar ventajas y desventajas. b. Explicar concurrencia optimista, dar un caso de uso y por qué conviene.
- **[2do-1C 2025] Punto 5:** Control de concurrencia: Definir 2-phase locking, timestamps y concurrencia optimista. Situación de supermercado con mucha contención en hora pico: ventajas y desventajas para los 3 tipos de control, elegir uno y justificar.

---

5. Comunicación y Redes

- **[2do-2C 2023] Ejercicio 3:** Qué es un socket y qué servicio utilizaría para implementar una aplicación de streaming de películas. Comparar las características y propiedades de los sockets de Unix con los channels de Rust.
- **[1er-1C 2025] Punto 1:** a. Describir el modelo OSI y explicar objetivos y capacidades de cada capa. b. Explicar qué es un socket y qué modelo de concurrencia usan.

---

6. Redes de Petri

- **[5to-2C 2024]:** Explicar qué es la función de entrada y salida y el grafo de alcance de una red de Petri y hacer la red de una comunicación cliente-servidor.
- **[3ero-1C 2025] Punto 2:** Funciones de entrada y salida de una red de Petri, y diagrama del cliente servidor.
- **[2do-1C 2025] Punto 2:** a. Definición y para qué sirve. Explicar diferencia entre ordinaria vs general. b. Definición de grafo de alcance y ejemplo. c. Graficar una red de Petri de productor-consumidor con buffer acotado.

💻 PARTE DESARROLLO (Práctica)

Desarrollo con Actores (Pseudo-código Rust)

- **[1er-2C 2022]:** Un productor-consumidor con actores. El productor manda números aleatorios cada un intervalo. Se puede consultar al consumidor el acumulado.
- **[2do-2C 2023] Ejercicio 5:** Implementar en pseudo-código Rust (utilizando actores) un mecanismo para consultar en tiempo real el resultado parcial de una elección donde las mesas van emitiendo sus votos propios de manera concurrente.
- **[5to-2C 2024]:** Diseñar un sistema concurrente distribuido para manejar los pedidos de una tienda usando actores, detallar mensajes y mecanismos de resiliencia y persistencia de datos (similar al TP).

Desarrollo con Transacciones y Sincronización

- **[2do-2C 2024] Punto 5:** Planteo de una situación de compra de tickets de avión y sistema de pagos. Implementar una solución siguiendo un modelo de transacciones (2-phase locking).
- **[1er-1C 2025] Punto 5 (Condición de aprobación):** Sistema de compra de pasajes de micro.
    - a. Modelar un sistema de transaccionalidad (Two-phase commit), explicar mensajes y prevención de problemas (reserva sin pago).
    - b. Diagramar entidades, mensajes y flujo.
    - c. Explicar dos casos de éxito y dos de falla.
- **[3ero-1C 2025] Punto 5:** Problema de reserva de restaurante (congestión, timeout de 5 mins).
    - 5.1. Detectar problemas de concurrencia.
    - 5.2. Resolver problemas detallando método de sincronización (semáforos, actores, transacciones).
    - 5.3. Explicar manejo de 3 casos: dos usuarios reservan la última mesa, timeout de pago, falla servidor de lock.
- **[3ero-2C 2024] Punto 5:** Lo mismo que los otros finales (reservas/compras) pero aplicado a **habitaciones de hotel**.

Extras x tema

Introducción

- ¿Qué modelos de concurrencia se ven?
  1-Estado mutable compartido, 2-Paralelismo/Fork-Join, 3-Canales/Mensajes, 4-Programación asincrónica, 5-Actores
- ¿De qué formas se puede proteger el estado compartido?
  Se puede proteger mediante Exclusión Mutua (usando Mutex o locks), Semáforos, Monitores, o evitando el estado compartido mediante el Pasaje de Mensajes (canales) donde el estado "se mueve" de un hilo a otro.

Fork Join

- ¿Qué es fork join?
  Es un modelo para paralelizar tareas dividiéndolas en subtareas más pequeñas (fork) que se ejecutan en paralelo y luego se esperan sus resultados para combinarlos (join). Es ideal para algoritmos "divide y vencerás".
- ¿Cómo se implementa fork join?
  Se implementa mediante un thread pool y colas de tareas. Una tarea grande se descompone en pequeñas; si una tarea es lo suficientemente pequeña, se resuelve; si no, se hace "fork" de nuevas subtareas.
- ¿Cuál es la performance ideal de fork join?
  Es O(T1/p), donde T1 es el tiempo secuencial y p el número de procesadores, asumiendo que el trabajo está perfectamente balanceado.
- ¿Qué desventajas tiene fork join?
  Puede tener sobrecarga (overhead) por la creación excesiva de tareas pequeñas y puede sufrir desbalance de carga si algunas ramas del árbol de ejecución son mucho más pesadas que otras.
  Solo es aplicable a tareas que son divisibles en subtareas
- ¿Cómo funciona work stealing?
  Cada hilo tiene su propia cola de tareas. Cuando un hilo termina las suyas, "roba" tareas del final de la cola de otro hilo que esté ocupado. Esto mejora el balance de carga y reduce la contención en comparación con una única cola global.

Programación Asincrónica

- ¿Cuándo se usa programación asincrónica
- ¿Qué es un future
- ¿Qué es una expresión wait
- ¿Qué es una tarea
- ¿Qué son los tipos Pin

- ¿Qué es el modelo piñata
- ¿Qué hace async
- ¿Qué hace await
- ¿Qué hace block_on
- ¿Cuándo usar programación asíncrona

Corrección

- ¿Qué propiedades de corrección existen
- ¿Qué se debe cumplir para que suceda un deadlock
- ¿Qué es un deadlock
- ¿Cómo se implementan los locks en Unix
- ¿Cómo se implementan los locks en Rust
- ¿Qué tipos de locks tiene Rust
- ¿Qué es un lock envenenado

Sincronización

- ¿Cuáles son los estados de ejecución de un proceso
- ¿Qué es un semáforo y que operaciones tiene
- ¿Qué es un mutex y que operaciones tiene
- ¿Cómo es el problema del productor consumidor
- ¿Cómo se resuelve el problema del productor consumidor con semáforos
- ¿Qué es una barrera y que operaciones tiene
- ¿Qué es un monitor y qué operaciones tiene
- ¿Qué compone a un monitor
- ¿Qué es un condition variable y qué operaciones tiene

Redes de Petri

- ¿Qué es una red de Petri
- ¿Qué problemas clásicos de la materia se pueden modelar con redes de Petri

Actores

- ¿Qué es el modelo de actores
- ¿Cómo puede ser la comunicación entre actores
- ¿Qué posee cada actor
- ¿Qué es un mensaje
- ¿Qué posee un canal
- ¿Qué posee un actor
- ¿Qué no comparte un actor
- ¿De qué formas se puede enviar un mensaje
- ¿Qué es el arbitrer
- ¿Cuál es la diferencia entre actores y mensajes?

La diferencia con los canales/mensajes es que no hay una terminación formal de la entidad que cambia el estado, solo tengo mensajes para reemplazar el uso del estado compartido.

Exclusión Mutua Distribuida

- ¿Qué algoritmos de exclusión mutua distribuida existen
- ¿Qué es un algoritmo centralizado
- ¿Qué es un algoritmo distribuido
- ¿Qué es un algoritmo de token ring

Modelo Cliente-Servidor

- ¿Qué tipos de arquitectura existen?
- ¿Qué tipos de servidor existen?
- ¿Qué capas componen el modelo OSI?

Algoritmos de Elección

- ¿Qué algoritmos de elección existen?
- ¿Qué es un algoritmo bully?
- ¿Qué es un algoritmo anillo?

Transacciones

- ¿Qué es una transacción?
- ¿Qué primitivas tienen las transacciones?
- ¿Qué son las propiedades ACID?
- ¿Qué algoritmos de implementación de transacciones existen?
- ¿Qué es un private workspace?
- ¿Qué es un write ahead log?
- ¿Qué es un algoritmo de two phase commit
- ¿Qué es un algoritmo de two phase locking
- ¿Qué es un algoritmo de optimistic concurrency control
- ¿Qué es un algoritmo de timestamps
- ¿Qué algoritmos de detección de deadlocks distribuidos existen
- ¿Qué es un algoritmo centralizado
- ¿Qué es un algoritmo distribuido
- ¿Qué es un algoritmo de wait die
- ¿Qué es un algoritmo de wound wait

Ambientes Distribuidos

- ¿Qué es una entidad?
- ¿Qué capacidades tiene una entidad?
- Qué eventos externos puede recibir una entidad?

- ¿Qué son las acciones?
- ¿Cuáles son las reglas?
- ¿Qué es el comportamiento?
- ¿Qué restricciones siguen los mensajes?
- ¿Qué métricas se usan para comparar algoritmos distribuidos?
- ¿Qué tipos de eventos existen?
- ¿Cuál es el estado interno de una entidad?
- ¿Qué es el conocimiento de una entidad?
- ¿Qué tipos de conocimiento de una entidad existen?
- ¿Qué es la información métrica?
- ¿Qué son las propiedades topológicas?
- ¿Qué es un mapa topológico?