

Ejercicios de parciales ordenados por su tipo, delante del ej. Está escrito de qué parcial salió.

Modelar, justificar y dejar en claro lo que sabemos. Evaluar tradeoffs, escribimos struct tanto.

[Ejercicios de parciales ordenados por su tipo, delante del ej. Está escrito de qué parcial salió.](#)

[Modelo de Concurrencia](#)

[Verdadero o Falso](#)

[Es busy wait?](#)

[Corrección de código](#)

[Implementación de algún problema con algún modelo específico](#)

[Redes de petri](#)

Modelo de Concurrencia

[1c2025]

Ejercicio 4

Describa y justifique con que modelo de concurrencia modelaría la implementación para cada uno de los siguientes casos de uso

- a. Una aplicación que encripta múltiples archivos .PDF con una clave privada.
- b. El backend para una aplicación de un restaurante donde múltiples mozos pueden sumar pedidos y cobrar en todas las mesas.
- c. Una aplicación que consulta periódicamente las cotizaciones de Bitcoin en cientos de exchanges y brokers.
- d. Una aplicación que subtitula para videos, convirtiendo voz a texto con una red neuronal.

- a. Fork join
- b. ?
- c. ?
- d. Fork join

[1c2025-recu2]

Ejercicio 3. Describa y justifique con que modelo de concurrencia modelaría la implementación para cada uno de los siguientes casos de uso

- a. Un sistema de consultas en tiempo real sobre un conjunto de miles de tablas.
- b. El backend para un juego estilo MMOG (massive multiplayer online game)
- c. Un crawler que descarga contenidos de internet para entrenar una red neuronal.
- d. Una aplicación que dado un conjunto de videos de ejemplo, genera un nuevo video donde los mismos personajes dicen lo que se les indica con un prompt ("deepfake").

- a. ?
- b. ?
- c. ?
- d. ni idea

[1c2025-recu1]

Ejercicio 3

Describa y justifique con que modelo de concurrencia modelaría la implementación para cada uno de los siguientes casos de uso

- Un sistema de subastas en línea donde múltiples usuarios ofrecen por ítems en tiempo real, se debe actualizar el precio del producto y notificar a los usuarios involucrados.
- Una aplicación de una pinturería que aplica filtros en una imagen para ver cómo quedarían distintos colores sobre una pared.
- Una memoria cache utilizada por un modelo de Machine Learning usado para reconocimiento facial.
- Una herramienta que dado N imágenes (donde N es muy grande) las junta en grupos de 100 y las transforma a PDF.

- a. ?
- b.
- c.
- d. Fork join

[2c2023]

1. Describa cuál modelo utilizar para las siguientes situaciones:

- Cálculo de matrices para modelo de redes neuronales.
- Solicitudes a distintas APIs y combinar el resultado.
- Leer Log de una pagina que es muy visitada
- Acceder al BackEnd de un videojuego

- a.
- b.
- c.
- d.

[1c2024]

1. Revisando el diseño aplicado en algunos proyectos, se encontró el uso de las siguientes herramientas para resolver problemas de concurrencia. Para cada uno de los problemas enumere ventajas o desventajas de utilizar la solución propuesta y mencione cual utilizaría usted.

- Renderizado de videos 3D en alta resolución, utilizando programación asíncrona.
- Aplicación que arma una nube de palabras a partir de la API de Twitter, utilizando barriers y mutex.
- Una aplicación para realizar una votación en vivo para un concurso de televisión, optimizada con Vectorización.

- a.
- b.
- c.
- d.

[2c2024]

5. Describa y justifique con que modelo de concurrencia modelaría la implementación para cada uno de los siguientes casos de uso

- a. Convertir un conjunto extenso de archivos de .DOC a .PDF
- b. El backend para una aplicación de preguntas & respuestas competitiva al estilo Menti o Kahoot.
- c. Una memoria caché utilizada para reducir la cantidad de requests en un servidor web a una base de datos.
- d. Una API HTTP que ejecuta un modelo de procesamiento de lenguaje natural para clasificar el sentimiento de un mensaje.

- a.
- b.
- c.
- d.

[2c2024-recu1]

Ejercicio 1

Enumere y justifique qué modelo de concurrencia es más conveniente utilizar para cada uno de los siguientes casos de uso

- Calcular productos de matrices para entrenar redes neuronales
- Realizar llamadas a diferentes APIs y combinar sus resultados
- Procesar los archivos de log de acceso de un sitio web muy concurrido
- Backend de una app para editar documentos de forma colaborativa en línea, como un Google Doc

- a.
- b.
- c.
- d.

Verdadero o Falso

[1c2025]

Ejercicio 5

Verdadero o Falso. Justifique

- a. Procesos, hilos y tareas asincrónicas todos poseen espacios de memoria independientes entre sí.
- b. El scheduler del sistema operativo puede detener una tarea asincrónica puntual y habilitar la ejecución de otra para el mismo proceso.
- c. En vectorización, tanto las operaciones verticales como las operaciones horizontales devuelven resultados vectoriales
- d. Un hilo esperando sobre una condvar sólo puede despertarse cuando otro hilo hace signal de la misma.

- a.
- b.
- c.
- d.

[1c2025-recu2]

Ejercicio 2. Verdadero o Falso. Justifique

- a. Las tareas asincrónicas son especialmente útiles para procesos de cómputo intensivo.
- b. El estado interno de un actor puede accederse desde otros actores mediante locks
- c. Programando en el modelo fork-join se debe tener especial cuidado en evitar condiciones de carrera.
- d. En el estado mutable compartido los hilos deben verificar periódicamente por actualizaciones de dicho estado

- a.
- b.
- c.
- d.

[1c2025-recu1]

Ejercicio 2

Verdadero o Falso. Justifique

- a. En un ambiente de ejecución con una única CPU, un conjunto de hilos de procesamiento intensivo tomarán un tiempo de ejecución significativamente menor a un conjunto de tareas asíncronas que ejecuten el mismo procesamiento.
- b. Programando en el modelo fork-join se debe tener especial cuidado en evitar condiciones de carrera.
- c. En el estado mutable compartido los hilos deben verificar periódicamente por actualizaciones de dicho estado
- d. El estado interno de un actor se encuentra protegido por una exclusión mutua.

a.

b.

c.

d.

[1c2024]

2. Programación asíncrona. Elija verdadero o falso y explique brevemente por qué:

- El encargado de hacer poll es el thread principal del programa.
- El método poll es llamado únicamente cuando la función puede progresar.
- El modelo piñata es colaborativo.
- La operación asíncrona inicia cuando se llama a un método declarado con `async`.

a.

b.

c.

d.

[2c2024]

4. Verdadero o Falso. Justifique

- a. Procesos, hilos y tareas asincrónicas poseen espacios de memoria independientes.
 - b. El scheduler del sistema operativo puede detener una tarea asincrónica puntual y habilitar la ejecución de otra para el mismo proceso.
 - c. Tanto los threads, como las tareas asincrónicas disponen de un stack propio.
 - d. En un ambiente de ejecución con una única CPU, un conjunto de hilos de procesamiento intensivo tomarán un tiempo de ejecución significativamente menor a un conjunto de tareas asincrónicas que ejecuten el mismo procesamiento.
-
- a.
 - b.
 - c.
 - d.

[2c2024-recu1]

Ejercicio 4

Verdadero o Falso. Justifique

- a. El estado interno de un actor se encuentra protegido por una exclusión mutua.
 - b. La llamada al método poll de un Future es bloqueante.
 - c. Programando en el modelo fork-join se debe tener especial cuidado en evitar condiciones de carrera.
 - d. Un hilo esperando sobre una condvar sólo puede despertarse cuando otro hilo hace signal de la misma.
-
- a.
 - b.
 - c.
 - d.

Es busy wait?

-> dependiendo el contexto, si estás en un dispositivo, te quedas sin batería o quemas el hardware.

[1c2025]

```
1. for _ in 0..MINERS {
    let copper = Arc::clone(&resource);
    thread::spawn(move || loop {
        let mined_amount = rand::thread_rng().gen_range(1..10);
        *copper.write().expect("failed to mine") += mined_amount;
        let delay = rand::thread_rng().gen_range(3000..7000);
        thread::sleep(Duration::from_millis(delay));
    });
}
```

[1c2025]

```
2. fn philosopher(id: usize, first_chopstick: Arc<Semaphore>, second_chopstick
    Arc<Semaphore>) {
    loop {
        println!("Philosopher {}: thinking", id);
        thread::sleep(Duration::from_millis(rng.gen_range(500..1500)));

        loop {
            println!("Philosopher {}: taking first chopstick", id);
            let first_access = first_chopstick.acquire();

            println!("Philosopher {}: attempting second chopstick", id);
            match second_chopstick.try_acquire() {
                Ok(second_access) => {
                    println!("Philosopher {} eating", id);
                    let delay = rand::thread_rng().gen_range(3000..7000);
                    thread::sleep(Duration::from_millis(delay));
                    break;
                }
                Err(_) => {
                    println!("Philosopher {} was unsuccessful", id);
                    drop(first_access);
                }
            }
        }
    }
}
```

[1c2025]

```
3. struct Data {
    value: Option<i32>,
}

fn main() {
    let data = Arc::new(Mutex::new(Data { value: None }));

    let c1 = Arc::clone(&data);
    let t1 = thread::spawn(move || {
        // does some work
        let mut lock = c1.lock().unwrap();
        lock.value = Some(42);
        ...
    });

    let c2 = Arc::clone(&data);
```

```
let t2 = thread::spawn(move || {
    loop {
        let lock = c2.lock().unwrap();
        if lock.value.is_some() {
            println!("Valor obtenido: {}", lock.value.unwrap());
            break;
        }
    }
});

t1.join().unwrap();
t2.join().unwrap();
}
```

[1c2024]

```
for _ in 0..MINERS {
    let lithium = Arc::clone(&mineral);
    thread::spawn(move || loop {
        let mined = rand::thread_rng().gen();
        let random_result: f64 = rand::thread_rng().gen();

        *lithium.write().expect("failed to mine") += mined;
        thread::sleep(Duration::from_millis(5000 as f64 *
random_result) as u64));
    })
}
```

```
for _ in 0..MINERS {
    let lithium = Arc::clone(&mineral);
    let batteries_produced = Arc::clone(&resources);
    thread::spawn(move || loop {
        let mut lithium = lithium.write().expect("failed");
        if lithium >= 100 {
            lithium -= 100;
            batteries_produced.write().expect("failed to produce") += 1
        }
        thread::sleep(Duration::from_millis(500));
    })}
}
```

[2c2024]

1. Para cada uno de los siguientes fragmentos de código indique si es o no es un busy wait. Justifique en cada caso.

```
1. loop {
    match TcpStream::connect("127.0.0.1:8080") {
        Ok(mut stream) => {
            stream.write_all(message.as_bytes()).expect("error")
        }
        Err(_) => {
            let random_result: f64 = rand::thread_rng().gen();
            thread::sleep(
                Duration::from_millis((5000 as f64 * random_result) as u64)
            );
        }
    }
}
```

```
2. loop {
    let random_result: f64 = rand::thread_rng().gen();
    thread::sleep(Duration::from_millis(random_result as u64));
    let mut items = self.pending_acks.lock().unwrap();
    let now = Instant::now();
    let mut i = 0;
    while i < items.len() {
        if items[i].expiration <= now {
            if items[i].item_type == "ACK" {
                let _ = items.remove(i);
                drop(items);
                self.send_result_interfaces();
                break;
            }
        } else {
            i += 1;
        }
    }
}
```

```
3. for _ in 0..MINERS {
    let copper = Arc::clone(&resource);
    thread::spawn(move || loop {
        let mined_amount = rand::thread_rng().gen_range(1..10);
        *copper.write().expect("failed to mine") += mined_amount;
        let delay = rand::thread_rng().gen_range(3000..7000);
        thread::sleep(Duration::from_millis(delay));
    });
}
```

Corrección de código

-> ojo con mezclar actores con convars
[1c2025]

Ejercicio 2

Dado el siguiente fragmento de código, indique el nombre del problema que modela. Indique si la implementación es correcta o describa cómo mejorarla.

```
fn main() {
    const N: usize = 5;

    let producers_waiting = Arc::new(Semaphore::new(0));
    let consumer_done = Arc::new(Semaphore::new(0));
    let data = Arc::new(Mutex::new(None));

    let producers_waiting_clone = producers_waiting.clone();
    let consumer_done_clone = consumer_done.clone();

    let consumer = thread::spawn(move || loop {
        println!("[Consumer] Sleeping...");
        producers_waiting_clone.acquire();

        println!("Doing my job with data {}...", data.lock().expect("can read"));
        thread::sleep(Duration::from_secs(2));

        consumer_done_clone.release();
        println!("Job finished");
    });

    let producers: Vec<_> = (0..N).map(|id| {
        let producers_waiting_clone = producers_waiting.clone();
        let consumer_done_clone = consumer_done.clone();

        thread::spawn(move || loop {
            let delay = rand::thread_rng().gen_range(3000..7000);
            thread::sleep(Duration::from_millis(delay));
            println!("[Producer {}] Arrived", id);
            producers_waiting_clone.release();

            *data.lock().expect("can't set data") = Some(id)

            println!("[Producer {}] waiting to be consumed", id);
            consumer_done_clone.acquire();

            println!("[Producer {}] Leaving", id);
        })
    }).collect();

    consumer.join().unwrap();
    for p in producer { p.join().unwrap(); }
}
```

Modele una red de Petri para el problema del punto anterior. Si hubiera propuesto mejoras, inclúyalas.

[1c2025-recu2]

Ejercicio 1. Una IA generó los siguientes fragmentos de código. Identifique problemas y proponga soluciones.

```
a.
struct ResourceGuard {
    guard: Semaphore
}
impl Actor for ResourceGuard {
    type Context = Context<Self>;
}
impl Handler<Acquire> for ResourceGuard {
    type Result = ();
    fn handle(&mut self, msg: Acquire, _ctx: &mut Context<Self>) -> Self::Result {
        self.guard.acquire();
    }
}
impl Handler<Release> for ResourceGuard {
    type Result = ();
    fn handle(&mut self, msg: Release, _ctx: &mut Context<Self>) -> Self::Result {
        self.guard.release();
    }
}

b.
impl Handler<Process> for Resource {
    type Result = ();

    fn handle(&mut self, msg: Process,
              _ctx: &mut Context<Self>) -> Self::Result {
        if !self.resourceReady {
            _ctx.address().do_send(msg)
        }
        // ...
    }
}
```

[1c2025-recu1]

Ejercicio 1. A partir de los siguientes fragmentos de código, identifique problemas y proponga soluciones.

```
a.
struct Producer {
    buffer: Arc<(Condvar, Mutex<Vec<i32>>)>,
}
impl Actor for Producer {
    type Context = Context<Self>;
}
impl Handler<Produce> for Producer {
    type Result = ();
    fn handle(&mut self, msg: Produce, _ctx: &mut Context<Self>) -> Self::Result {
        let (cvar, lock) = &*self.buffer;
        let mut guard = cvar.wait_while(lock.lock().unwrap(), |buf| { buf.len() >= 10 })
        *guard.push(msg.value)
    }
}
impl Handler<Consume> for Producer {
    type Result = ();
    fn handle(&mut self, msg: Consume, _ctx: &mut Context<Self>) -> Self::Result {
        let (cvar, lock) = &*self.buffer;
        let mut guard = cvar.wait_while(lock.lock().unwrap(), |buf| { buf.len() == 0 })
        return *guard.pop();
    }
}
```

[1c2025-recu1]

```

b.
impl Handler<PrepararCafe> for Barista {
    fn handle(&mut self, _msg: PrepararCafe, _ctx: &mut Context<Self>) {
        loop {
            for maquina in self.maquinas.iter() {
                match maquina.send(EstasLibre).await {
                    Ok(true) => {
                        println!("Barista. Máquina libre, preparando café");
                        // Preparar café (sin esperar resultado)
                        maquina.do_send(PrepararCafe);
                        return;
                    },
                    Ok(false) => {
                        println!("Barista. Sigo buscando máquina libre");
                    },
                    Err(e) => {
                        println!("Barista. Error: {}", e);
                    }
                }
            }
        }
    }
}

```

[1c2024]

```

pub struct SynchronizationStruct {
    mutex: Mutex<i32>,
    cond_var: Condvar,
}

impl SynchronizationStruct {
    pub fn new(size: u16) -> SynchronizationStruct {
        SynchronizationStruct {
            mutex: Mutex::new(size),
            cond_var: Condvar::new(),
        }
    }

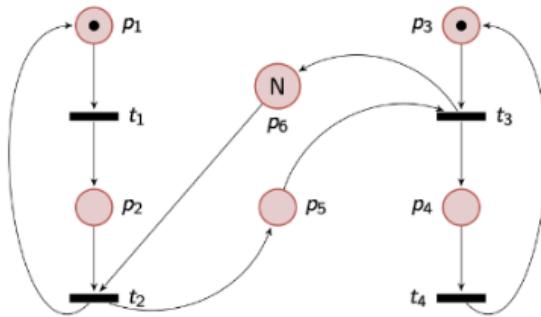
    pub fn function_1(&self) {
        let mut amount = self.mutex.lock().unwrap();
        if *amount <= 0 {
            amount = self.cond_var.wait(amount).unwrap();
        }
        *amount -= 1;
    }

    pub fn function_2(&self) {
        let mut amount = self.mutex.lock().unwrap();
        *amount += 1;
        self.cond_var.notify_all();
    }
}

```

[1c2024]

5. Dados la siguiente red de Petri y fragmento de código, indique el nombre del problema que modelan. Indique si la implementación es correcta o describa cómo mejorarla.



```
fn main() {
    let sem = Arc::new(Semaphore::new(0));
    let buffer = Arc::new(Mutex::new(Vec::with_capacity(N)));

    let sem_cloned = Arc::clone(&sem);
    let buf_cloned = Arc::clone(&buffer);
    let t1 = thread::spawn(move || {
        loop {
            // heavy computation
            let random_result: f64 = rand::thread_rng().gen();
            thread::sleep(Duration::from_millis((500 as f64 *
random_result) as u64));

            buf_cloned.lock().expect("").push(random_result);
            sem_cloned.release()
        }
    });

    let sem_cloned = Arc::clone(&sem);
    let buf_cloned = Arc::clone(&buffer);
    let t2 = thread::spawn(move || {
        loop {
            sem_cloned.acquire();
            println!("{}", buf_cloned.lock().expect("").pop());
        }
    });

    t1.join().unwrap();
    t2.join().unwrap();
}
```

[2c2024-recu1]

Ejercicio 2

En una revisión de código se encontró el siguiente fragmento

```
const N:usize = 10;
fn main() {
    let buffer = Arc::new(Mutex::new(Vec::new().with_capacity(N)));
    let buffer_local = buffer.clone();
    let handle = thread::spawn(move || {
        loop {
            let mut buf = buffer_local.lock().unwrap();
            if buf.len() < N {
                buf.push(rand::thread_rng().gen());
            } else {
                drop(buf);
                thread::sleep(Duration::from_secs(1));
            }
        });
        loop {
            let mut buf = buffer.lock().unwrap();
            if !buf.is_empty() {
                println!("{} ", buf.pop().unwrap());
            } else {
                drop(buf);
                thread::sleep(Duration::from_secs(1));
            }
        }
    });
    handle.join().unwrap();
}
```

a. Describa qué problemas presenta este código y proponga una solución

b. Describa por qué son necesarias las llamadas a drop en esta implementación.

Implementación de algún problema con algún modelo específico

[1c2025]

Ejercicio 3

Se quiere abrir una estación de servicio YPF con 10 surtidores. Por el momento solamente se logró la habilitación de 3 tanques internos de nafta, el objetivo es a futuro poder habilitar más. Los surtidores atienden a los autos que vienen a cargar combustible, pero deben acceder a los tanques internos de un surtidor a la vez. Además, cuando llega el camión cisterna de YPF a recargar el combustible disponible en la estación de servicio, este debe asegurarse de que ningún tanque esté siendo usado por un surtidor ya que eso representa un riesgo de seguridad.

Diseñe el sistema utilizando el modelo de actores, y para cada entidad defina cuáles son los estados internos y los mensajes que intercambian.

[1c2025-recu2]

Ejercicio 4. Dado el siguiente enunciado

En un salón de videojuegos, los mismos se pagan con una tarjeta recargable. En el mismo salón se festejan cumpleaños. En el caso de estos, se le da a cada invitado una tarjeta que descuenta dinero de un pool compartido entre todos.

- a. Modelarlo con actores, describiendo sus estados y mensajes en pseudocódigo de Rust, y cómo reaccionan.
- b. Modelarlo con una red de Petri.

[1c2025-recu1]

Ejercicio 4

En una oficina sobre la 9 de Julio trabaja una empresa que posee una serie de máquinas de café automáticas distribuidas en todas las áreas, a lo largo de los 10 pisos que alquila.

Para comprar café en las máquinas, cada empleado tiene una tarjeta cargada con créditos. Sin embargo, los créditos son compartidos por equipo para mejor control de cada gerencia.

Los créditos se cargan el primer día hábil de cada mes, y además cada gerente tiene la posibilidad de agregar montos extra a lo largo del mes como bonus cuando considera que sus empleados deben ser recompensados.

Modele el sistema con actores, describiendo sus estados y mensajes en pseudocódigo de Rust. Describa cómo reaccionan ante los mensajes que reciben (en prosa).

Especifique cómo asegura su modelo el caso del "doble gasto" si dos tarjetas asociadas al mismo equipo se utilizan en dos máquinas al mismo tiempo.

[2c2023]

2. Desarrollar un pseudocódigo en Rust que solucione

- Se tiene 100 links de distintas páginas.
- Simular el tiempo de espera usando sleep y rand.
- No se pueden tener más de N threads activos a la vez (para tener latencia baja).
- Se desea calcular el tiempo promedio total.

[2c2023]

3. Armar un Semáforo usando Monitores.

[2c2024]

3. Se quiere abrir un restaurante en el barrio de San Telmo. Se espera que los clientes lleguen y sean atendidos por alguno de los mozos de turno, cada uno de los cuales tomará los pedidos de cada mesa, los notificará a la cocina y luego seguirá tomando otros pedidos. Como la cocina es chica los cocineros pueden entrar a buscar los ingredientes al depósito de a uno a la vez, y buscar entre los distintos alimentos les puede llevar un tiempo variable. Cuando los cocineros hayan terminado de preparar un pedido deben notificar a los mozos para que lo lleven a la mesa. Además, los mozos deben estar disponibles para cobrarle a los clientes. Diseñe el sistema utilizando el modelo de actores, y para cada entidad defina cuáles son los estados internos y los mensajes que intercambian.

[2c2024-recu1]

Ejercicio 5

Nos encontramos implementando un sistema para una cafetería de especialidad. Los clientes llegarán al mostrador y serán atendidos por algún cajero para tomar su pedido y cobrarlo. Los baristas irán tomando los pedidos ingresados y los prepararán. Por el momento se disponen únicamente de dos máquinas de café, las cuales solo puede utilizar un barista por vez. Finalmente, al tener el pedido listo, el barista llamará al cliente por su nombre para entregárselo.

Diseñe el sistema utilizando el modelo de actores. Para cada uno de ellos especifique su estado interno en pseudocódigo de Rust. Detalle los mensajes que reciben, sus payloads y cómo los procesan también en pseudocódigo.

Redes de petri

-> Modelar la atención de los servidores

[2c2023]

4. Dibujar la red de Petri para el problema de productor-consumidor con buffer acotado.

[2c2024]

2. Modelar una Red de Petri para el problema del Lector-Escritor sin preferencia. Luego, modele una solución que contemple preferencia de escritura.

[2c2024-recu1]

Ejercicio 3

Modelar una Red de Petri para la solución propuesta en el punto anterior

En una revisión de código se encontró el siguiente fragmento

```
const N:usize = 10;
fn main() {
    let buffer = Arc::new(Mutex::new(Vec::new().with_capacity(N)));
    let buffer_local = buffer.clone();
    let handle = thread::spawn(move || {
        loop {
            let mut buf = buffer_local.lock().unwrap();
            if buf.len() < N {
                buf.push(rand::thread_rng().gen());
            } else {
                drop(buf);
                thread::sleep(Duration::from_secs(1));
            }
        }
    });
    loop {
        let mut buf = buffer.lock().unwrap();
        if !buf.is_empty() {
            println!("{}: {}", buf.pop().unwrap());
        } else {
            drop(buf);
            thread::sleep(Duration::from_secs(1));
        }
    }
    handle.join().unwrap();
}
```

- a. Describa qué problemas presenta este código y proponga una solución
- b. Describa por qué son necesarias las llamadas a drop en esta implementación.