1° Cuatrimestre 2024

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 enuncie ventajas o desventajas de utilizar la solución propuesta y menciona cual utilizaría usted.

- Renderizado de videos 3D en alta resolución, utilizando programación asincrónica.
- 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.

2. Programación asincrónica. 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 asincrónica inicia cuando se llama a un método declarado con async.

3. Para cada uno de los siguientes fragmentos de código indique si es o no es un busy wait. Justifique en cada caso (Nota: mineral y batteries_produced son locks).

```
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));
    })}
```

4. Dada la siguiente estructura, nombre si conoce una estructura de sincronización con el mismo comportamiento. Indique posibles errores en la implementación.

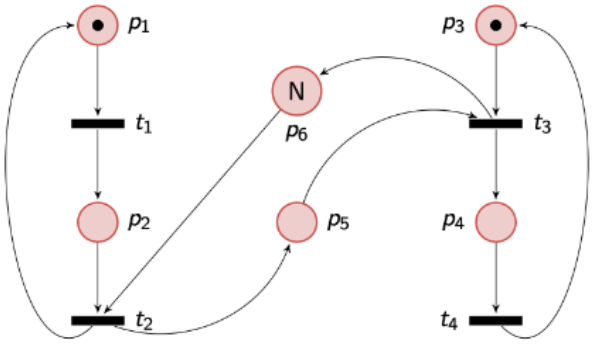
```
pub struct SynchronizationStruct {
    mutex: Mutex<i32>,
    cond_var: Condvar,
}

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();
    }
}
```

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();
}
```

2° Cuatrimestre 2024

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));
    });
}
```

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

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.

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.

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.

2° Cuatrimestre 2023

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

2. Desarrollar un pseudocodigo 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.

3. Armar un Semáforo usando Monitores.

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