TAREA 1

Problemas de sincronización

Problema de Santa Claus

Sistemas operativos

Profesor: ING. Gunnar Eyal Wolf Iszaevich

Semestre 2025-1
Grupo 6

Autores:

- ***Pérez Osorio Luis Eduardo***
- ***Flores García Claudio***

Problema Resuelto.- Santa Claus

- **Descripción del problema**

Santa Claus duerme en el Polo Norte mientras sus elfos trabajan frenéticamente en la construcción de millones de nuevos juguetes, a veces se topan con un problema y pueden pedir ayuda a Santa Claus pero solo de 3 en 3 para no "darle tanta lata", si no existe ningún problema Santa Claus sigue durmiendo.

Además, sus nueve renos pasan todo el año de vacaciones en las playas del Caribe. Santa Claus debe de despertar a tiempo para iniciar su viaje anual, cuando sus 9 renos están de vuelta, es hora de despertar a Santa Claus para que inicie su recorrido.

- **Reglas**

1. Santa Claus está dormido, solo despertará si:
    a. Sus 9 renos han vuelto de sus vacaciones.
    b. Existen 3 elfos que necesitan ayuda.
2. Puede haber cualquier cantidad de elfos.

Lenguaje y Entorno de Desarrollo

Desarrollamos esta solución al problema usando python con la biblioteca "threading" para poder hacer uso de los ejemplos vistos en clase como referencia, además de la facilidad de lectura y escritura del lenguaje.
Para ejecutar el programa es necesario tener instalado python y ejecutarlo desde la terminal, debido al uso de emojis para ilustrar el flujo del programa es necesario que la terminal esté configurada correctamente para mostrar utf-8.
Para ejecutar el programa se debe posicionar en el directorio que contiene el archivo santa.py y correr el comando.

python santa.py

Estrategia de sincronización.

Antes de explicar los mecanismos, en nuestro programa creamos a 10 elfos.

1. **Mecanismos.**

   - Semáforos
     - `santa_sleep` : Controla cuando Santa debe de despertar.
     - `ayuda_elfos` : Coordina la ayuda a los elfos.
     - `elfos_ayudados` : Confirma que los elfos fueron atendidos.
     - `renos_listos` : Es la autorización para preparar la entrega (trineo).
     - `renos_pueden_irse` : Controla cuando pueden partir para la entrega.

   - Mutex
     - `elfos_mutex` : Protege el acceso a la lista de elfos esperando por ayuda.
     - `renos_mutex` : Protege el contador de renos.

2. **Patrones de Sincronización.**

   - Patrón para los elfos (Grupos de espera: 3 en 3)

```
with self.elfos_mutex:
    if len(self.elfos_esperando) < 3:
        print(f"🧝 Elfo {id} necesita ayuda")
        self.elfos_esperando.append(threading.current_thread())

        if len(self.elfos_esperando) == 3:
            print("👥 Grupo de 3 elfos formado, despertando a Santa")
            self.santa_sleep.release()
```

Los elfos forman grupos de 3, cuando un elfo necesita ayuda se verifica si la lista aun tiene espacio, caso positivo, se agregá al elfo en la lista de ayuda y sea el caso en que ya completaron un grupo se despierta a Santa, cuando esto sucede se señaliza con un mensaje.

Pero, ¿qué pasa con los elfos que necesitan ayuda pero hay un grupo completo siendo atendido?

```
        # Esperar fuera del mutex para no bloquear a otros elfos
        self.elfos_mutex.release()
        self.ayuda_elfos.acquire()  # Esperar ayuda de Santa
        self.elfos_ayudados.release()  # Señalar que fue ayudado
        self.elfos_mutex.acquire()

        if threading.current_thread() in self.elfos_esperando:
            self.elfos_esperando.remove(threading.current_thread())
    else:
        print(f"🧝 Elfo {id} continúa trabajando solo")
```

Bueno, a los elfos se les manda a una “lista de espera” y continuarán trabajando solos pero cuando el grupo anterior ya fue atendido por Santa la lista que contiene el grupo de elfos ya atendido los remueve para que entren los que estaban en lista de espera.

- Patrón para los renos (Patrón de barrera).

```
with self.renos_mutex:
    self.renos_regresados += 1
    print(f"🦌 Reno {id} ha regresado ({self.renos_regresados}/9)")

    if self.renos_regresados == 9:
        print("🦌 Todos los renos han regresado")
        self.santa_sleep.release()

# Esperar a que Santa prepare el trineo
self.renos_listos.acquire()
# Esperar a completar la entrega
self.renos_pueden_irse.acquire()

with self.renos_mutex:
    self.renos_regresados -= 1
```

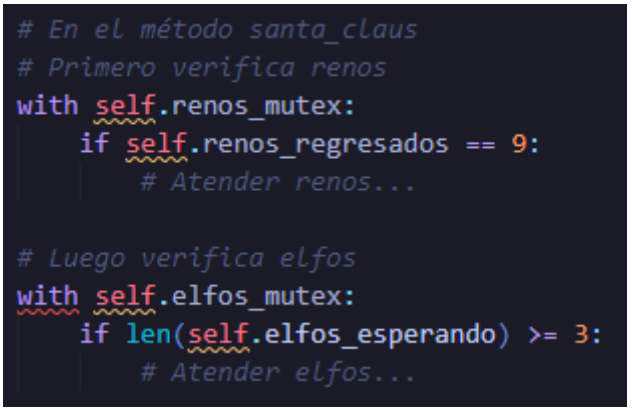
El patrón de barrera se refiere a un mecanismo dónde los hilos (renos) deben esperar a un punto específico hasta cumplir una condición (que lleguen todos (9)) para poder continuar.
En el programa, es como si existieran 5 fases:

1. Llegada de los renos: Es el mutex que protege el contador de posibles modificaciones simultáneas.
2. Esperar a todos (9): Esta es la barrera principal, cuando los renos se acompletan se despierta a Santa para preparar la entrega.
3. Preparación: Esta es la primer barrera que la definimos como si Santa estuviera preparando el trineo para partir.
4. Entrega: Esta es la segunda barrera, que se definió como todo el proceso de entrega de regalos hasta terminar.
5. Reset: Que es el reinicio del contador, que se puede interpretar como si los renos se fueran de vacaciones.

3. Estrategia general de implementación.

Como vimos en clase, este problema se puede manejar como un sistema PRODUCTOR - CONSUMIDOR, donde los renos y los elfos actúan como productores de solicitudes y Santa como un consumidor de solicitudes, pero siempre teniendo en cuenta la prioridad de los renos sobre los elfos.

4. **Sistema de prioridades.**



```
# En el método santa_claus
# Primero verifica renos
with self.renos_mutex:
    if self.renos_regresados == 9:
        # Atender renos...

# Luego verifica elfos
with self.elfos_mutex:
    if len(self.elfos_esperando) >= 3:
        # Atender elfos...
```

Ignorando el error del código ya es para facilitar la explicación, en el método que nos permite simular las actividades que realiza Santa Claus, para las prioridades primero verifica la condición de los renos y posteriormente verifica las solicitudes de los elfos, de esta manera aseguramos mantener la importancia de los renos sobre los elfos.

5. **Comentarios.**

En el planteamiento del problema no quedaba claro si se debía dar prioridad a una clase de hilo (renos o elfos), nosotros interpretamos que se le debía dar prioridad a los renos para que se iniciara a repartir regalos tan pronto todos los renos estuvieran listos, en base a las pruebas que hemos podido hacer parece que no se presentan condiciones de carrera o bloqueos mutuos en la ejecución paralela de los hilos y consideramos que se resolvió el problema exitosamente.