



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**  
**FACULTAD DE INGENIERÍA**

Sistemas Operativos

**PROFESOR(A):**

Gunnar Eyal Wolf Iszaevich

**NOMBRES(S) ALUMNOS(s):**

Hernández Ortiz Jonathan Emmanuel

Pérez Avin Paola Celina de Jesús

**PROYECTO 02**

Situación Cotidiana Paralelizable

**PROBLEMA ABORDADO:**

“EL CINE”

**SEMESTRE 2024-1**

# ÍNDICE

PLANTEAMIENTO DE LA SITUACIÓN.....	3
MECANISMOS DE SINCRONIZACIÓN EMPLEADOS.....	3
LÓGICA DE SINCRONIZACIÓN.....	6
ESTADO COMPARTIDO.....	7
DESCRIPCIÓN ALGORÍTMICA.....	8
DESCRIPCIÓN DE LA INTERACCIÓN.....	8
ENTORNO DE DESARROLLO.....	9
LENGUAJE DE PROGRAMACIÓN Y VERSIÓN.....	9
SISTEMA OPERATIVO.....	9
EJECUCIONES.....	10

## PLANTEAMIENTO DE LA SITUACIÓN

Imagina que estamos organizando una emocionante noche de cine en una sala con capacidad para 10 espectadores. La película que proyectaremos es extremadamente popular y ha generado un gran entusiasmo, con 100 clientes deseando ingresar a la sala para disfrutarla. Para garantizar que esta experiencia sea fluida y justa para todos, necesitamos encontrar una manera de coordinar el acceso de estos 100 clientes a una sala con una capacidad máxima de 10 personas a la vez. El objetivo principal de esta coordinación es asegurarnos de que todos los clientes tengan la oportunidad de disfrutar de la película sin enfrentar problemas ni aglomeraciones.

Para lograr esto, hemos desarrollado un programa que actúa como un “amable conserje virtual”. Utiliza hilos, como si fueran asistentes virtuales, para controlar el flujo de entrada y salida de clientes de la sala de cine. ¡Esto nos permite asegurar que en ningún momento más de 10 personas estén dentro de la sala! Y que todos tengan la misma oportunidad de ver la película de manera cómoda y ordenada. En resumen, nuestro objetivo es hacer que esta noche de cine sea una experiencia excepcional para todos los asistentes, donde todos puedan disfrutar de la película sin problemas. ¡Vamos a hacer que esta noche de cine sea genial! Juntos.

## MECANISMOS DE SINCRONIZACIÓN EMPLEADOS

### SEMÁFORO(S):

Un semáforo es un mecanismo utilizado en programación concurrente y multihilo para controlar el acceso a recursos compartidos. En términos generales, un semáforo funciona como una señal o indicador que permite a los hilos o procesos coordinar sus actividades y evitar conflictos al acceder a recursos compartidos. ¡Esta situación no es la excepción! Pues el uso del semáforo “semaforo\_cine” ejemplifica esta función al controlar la entrada y salida de espectadores en el cine, asegurando que no más de un número específico de personas acceda a la sala al mismo tiempo.

- En este código, se utiliza un semáforo llamado “semaforo\_cine” para regular el acceso a los asientos de la sala de cine. En su configuración inicial, se establece un contador en 10, lo que indica que hay 10 asientos disponibles en la sala. Este semáforo se emplea para garantizar que no más de 10 personas puedan ocupar asientos simultáneamente.

```
# Se crea un semáforo para controlar el acceso a los asientos de la sala del cine.  
# Suponiendo, como ejemplo, que hay 10 asientos disponibles.  
semaforo_cine = threading.Semaphore(10)
```

- Cuando un espectador desea ingresar a la sala de cine, debe adquirir el semáforo utilizando “semaforo\_cine.acquire()”. Esto evita que más de 10 personas entren en la sala al mismo tiempo, ya que si el semáforo está en 0, bloquea el acceso y espera a que primeramente se libere.

```
# Intenta adquirir un asiento si hay disponibilidad.
semaforo_cine.acquire()
print(mi_color + "\nEspectador %d ha ocupado un asiento y está viendo la película." % espectador_id)
print("¡A disfrutar!")
```

- Una vez que un espectador obtiene el semáforo y su asiento, lo libera usando “semaforo\_cine.release()”, permitiendo así que otro espectador entre a la sala.

```
# Libera el asiento para el siguiente cliente del hilo.
semaforo_cine.release()
print(mi_color + "\nEspectador %d ha terminado de ver la película y ha dejado un asiento." % espectador_id)
print("Muy bonita la película... :)")
```

## HILOS (THREADS):

Los hilos son una parte fundamental para la simulación de la situación que planteamos. Pues cada espectador se representa mediante un hilo independiente, lo que permite que múltiples espectadores interactúen simultáneamente en la sala de cine.

### Creación de hilos:

- Se crea un bucle “for” para generar hilos para cada espectador. La variable “rango” se utiliza para asignar un identificador único a cada espectador, desde 1 hasta “num\_espectadores” (100 en este caso).

```
# Variable asignada con un rango para mejorar la visualización del bucle FOR.
rango = range(1, num_espectadores + 1)

# Creamos un bucle FOR para enviar a los espectadores y visualizar los hilos.
for espectador_id in rango:
    # Aplicamos un retraso para dar tiempo a que los espectadores pasen.
    time.sleep(3)
    # Hilo para los espectadores que se les pasa como argumentos a la función 'ver_pelicula'.
    # La variable espectador_id.
    hilo = threading.Thread(target=ver_pelicula, args=(espectador_id,))
    # Se añade una lista los hilos.
    hilos.append(hilo)
    # Se inicializan los hilos.
    hilo.start()
```

- Dentro del bucle “for”, se crea un hilo para cada espectador utilizando “threading.Thread”. Se especifica la función objetivo que el hilo ejecutará, que en este caso es la función “ver\_pelicula”. Además, se pasa el argumento “espectador\_id”, que identifica a cada espectador de manera única.

```
# Hilo para los espectadores que se les pasa como argumentos a la función 'ver_pelicula'.
# La variable espectador_id.
hilo = threading.Thread(target=ver_pelicula, args=(espectador_id,))
# Se añade una lista los hilos.
hilos.append(hilo)
# Se inicializan los hilos.
hilo.start()
```

Inicio de hilos:

- Después de crear un hilo para un espectador, se agrega el hilo a una lista llamada "hilos" para mantener un seguimiento de todos los hilos.

```
# Se añade una lista los hilos.
hilos.append(hilo)
# Se inicializan los hilos.
hilo.start()
```

- Luego, el hilo se inicia utilizando el método "start()". Cuando se inicia un hilo, comienza a ejecutar la función objetivo ("ver\_pelicula") en paralelo con otros hilos. Esto permite que múltiples espectadores vean la película simultáneamente.

Espera a la finalización:

- Una vez que se han creado y comenzado todos los hilos para los espectadores, el código utiliza un bucle "for" para esperar a que cada hilo termine. Esto se hace llamando al método "join()" en cada hilo. La función "join()" bloquea la ejecución del programa principal hasta que el hilo haya finalizado.
- Esta espera garantiza que todos los espectadores vean la película antes de que el programa finalice.

```
# Al igual que abrimos los hilos, debemos cerrarlos para optimizar los procesos.
# Y asimismo, no consumir recursos que ya no se ocupan.
for hilo in hilos:
    hilo.join()
```

Hilo de deserción aleatoria:

- Además de los hilos para los espectadores que ven la película, el código también crea un hilo adicional llamado "desercion\_thread" que simula la deserción aleatoria de espectadores. Este hilo se inicia y se ejecuta en paralelo con los hilos de los espectadores que ven la película.

```
# Simula la deserción aleatoria de espectadores.
def desercion_aleatoria():
    time.sleep(10) # Introduce un retardo antes de que comience la deserción.
    while True:
        espectador_id = random.randint(1, num_espectadores)
        semaforo_cine.release()
        print(Fore.YELLOW + f"Espectador {espectador_id} ha abandonado el cine.")
        time.sleep(2) # Espacio de tiempo antes de que otro espectador abandone.
```

## LÓGICA DE SINCRONIZACIÓN

La lógica de operación en este código se basa en la gestión de hilos y el uso de un semáforo para controlar el acceso a los asientos de la sala de cine, como ya se mencionó anteriormente. ¡Esta lógica debe asegurarse de que no haya más de 10 espectadores en la sala al mismo tiempo! Y asimismo, proporcionar una simulación realista de la experiencia de ver una película en un cine.

Semáforo para control de asientos:

- Se utiliza un semáforo llamado "semaforo\_cine" con una capacidad máxima de 10 asientos. Cuando un espectador intenta ingresar, adquiere el semáforo. Si hay asientos disponibles (el semáforo no está en su capacidad máxima), el espectador obtiene un asiento. Si la sala está llena, el espectador debe esperar hasta que un asiento se libere.

Hilos para espectadores:

- Cada espectador es representado por un hilo individual. Estos hilos ejecutan la función "ver\_pelicula". Cuando un espectador ingresa al cine, intenta adquirir el semáforo. Si el semáforo está disponible, el espectador obtiene un asiento. Si no, espera hasta que haya asientos disponibles.

Simulación de la experiencia de los espectadores:

- Cada hilo de espectador simula su experiencia de ver la película, incluyendo tiempos de espera aleatorios y acciones como comprar palomitas o ir al baño. Estos elementos aleatorios añaden realismo a la simulación.

Hilo de deserción aleatoria:

- Además de los hilos de los espectadores, se crea un hilo llamado "desercion\_thread" que simula que algunos espectadores abandonan la película en momentos aleatorios al liberar el semáforo.

Espera de finalización:

- El programa principal espera a que todos los hilos de los espectadores hayan terminado de ver la película antes de finalizar. Esto asegura que todos los espectadores tengan la oportunidad de ver la película.

## ESTADO COMPARTIDO

Variables o estructuras que son accesibles mediante múltiples hilos y que deben ser compartidas o sincronizadas adecuadamente para evitar problemas de concurrencia. ¡Y sí! Vaya que existen algunas variables y estructuras globales así en este código.

Semáforo: "semaforo\_cine" es un semáforo global que se utiliza para controlar el acceso a los asientos de la sala de cine. Este semáforo es compartido por todos los hilos y garantiza que no haya más de 10 espectadores en la sala al mismo tiempo. Los hilos lo adquieren y lo liberan para obtener asientos y dejar la sala.

```
# Se crea un semáforo para controlar el acceso a los asientos de la sala del cine.  
# Suponiendo, como ejemplo, que hay 10 asientos disponibles.  
semaforo_cine = threading.Semaphore(10)
```

Lista de colores: La lista "color" contiene colores que se asignan aleatoriamente a cada espectador. Es una variable global que se comparte entre los hilos para dar color a la salida en la consola. Aunque no es crítico para la sincronización, se utiliza para la visualización y es compartido entre los hilos.

```
# Se utiliza una biblioteca para agregar colores a los textos en la consola del terminal.  
from colorama import Fore  
  
# Colores disponibles para asignar a cada cliente y aplicar en la terminal.  
# Rojo      Blanco      Amarillo      Verde      Azul      Magenta      Cyan.  
color =[Fore.RED,Fore.WHITE,Fore.YELLOW,Fore.GREEN,Fore.BLUE,Fore.MAGENTA,Fore.CYAN]
```

Número de espectadores: "num\_espectadores" es una variable que representa la cantidad de espectadores que desean ver la película. Es una variable global utilizada para determinar cuántos hilos (espectadores) se crearán y se compartirá entre ellos.

```
# Cantidad de personas que desean disfrutar de la película.  
num_espectadores = 100
```

Lista de hilos: La lista "hilos" se utiliza para mantener un seguimiento de todos los hilos de espectadores que se crean. Es una variable global que se comparte entre los hilos para que el programa principal pueda esperar a que todos los hilos de espectadores finalicen antes de terminar la simulación.

```
# Se añade una lista los hilos.  
hilos.append(hilo)  
# Se inicializan los hilos.  
hilo.start()
```

Hilo de deserción aleatoria: El hilo "desercion\_thread" es un hilo adicional que simula la deserción aleatoria de espectadores. Es una variable global que se inicia y se ejecuta en paralelo con los hilos de espectadores. Este hilo también libera el semáforo global ("semaforo\_cine") al simular la deserción.

### **DESCRIPCIÓN ALGORÍTMICA**

- Cada espectador llega al cine de forma simulada con un retraso inicial y se le asigna un color aleatorio.
- Los espectadores esperan en una fila simulada antes de entrar a la sala de cine.
- Intentan adquirir un asiento utilizando el semáforo "semaforo\_cine". Si hay asientos disponibles, obtienen un asiento y el semáforo disminuye en 1. Si no hay asientos disponibles, deben esperar.
- Los espectadores simulan la experiencia de ver la película, con acciones aleatorias como comprar palomitas o ir al baño.
- Permanecen en la sala de cine durante un tiempo simulado que representa la duración de la película.
- Al finalizar la película, liberan el asiento utilizando el semáforo, lo que permite que otros espectadores entren.
- Finalmente, dejan la sala del cine y muestran un mensaje de salida en la consola.
- Además, el hilo "desercion\_thread" simula la deserción aleatoria de espectadores, lo que significa que algunos espectadores pueden abandonar la sala en momentos aleatorios, liberando asientos para otros espectadores.

### **DESCRIPCIÓN DE LA INTERACCIÓN**

Pues todo se da principalmente a través del uso del semáforo "semaforo\_cine" y la simulación de la experiencia de ver la película.

Semáforo para control de asientos:

- Los hilos de los espectadores interactúan con el semáforo "semaforo\_cine" para adquirir asientos en la sala de cine. Cuando un espectador intenta adquirir un asiento, se bloquea si el semáforo está en su capacidad máxima (0), lo que significa que la sala está llena. Esto evita que más de 10 espectadores ingresen simultáneamente, asegurando que la capacidad máxima de la sala se respete. Cuando un espectador libera un asiento al finalizar la película, el semáforo se incrementa en 1, permitiendo que otro espectador ingrese.

Experiencia en el cine:

- Los hilos de los espectadores simulan la experiencia de ver la película, y esta simulación incluye acciones aleatorias, como comprar palomitas o ir al baño. Aunque estas acciones no tienen una interacción directa entre los hilos, añaden variabilidad a la simulación y hacen que la experiencia sea más realista.

Hilo de deserción aleatoria:



- El hilo "desercion\_thread" interactúa con los hilos de los espectadores al liberar el semáforo de forma aleatoria. Esto simula la deserción de espectadores en momentos inesperados. Cuando el hilo de deserción libera el semáforo, un espectador puede abandonar la sala de cine, lo que permite que otro espectador entre. Esta interacción agrega un elemento aleatorio y realista a la simulación.

## ENTORNO DE DESARROLLO

El entorno de desarrollo necesario para ejecutar con éxito este código en Python en la terminal de Git Bash se puede describir en términos generales de la siguiente manera:

### ENTORNO DE DESARROLLO:

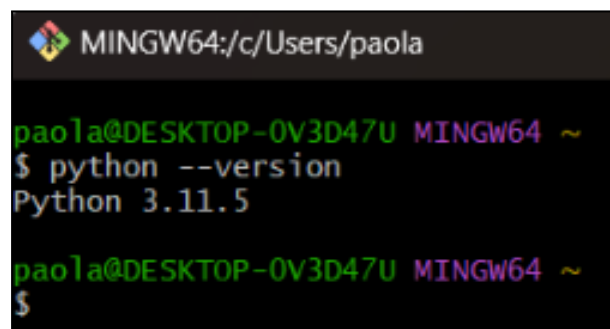
- Se requiere tener Python instalado en el sistema. Puede descargarse desde el sitio web oficial y seguir las instrucciones de instalación.
- La terminal de Git Bash se utiliza como el entorno desde el cual se ejecutará el código. Git Bash proporciona un entorno similar al de Unix en sistemas Windows.

### EJECUCIÓN DEL CÓDIGO:

- Descargar el archivo de código fuente en Python y guardarlo con la extensión ".py".
- Abrir Git Bash en el sistema.
- Ir al directorio donde se encuentra el archivo de código utilizando el comando "cd".
- Ejecutar el código utilizando el comando "python" seguido del nombre del archivo.
- Observar la salida en la terminal de Git Bash mientras se ejecuta la simulación de la experiencia de los espectadores en la sala de cine.

## LENGUAJE DE PROGRAMACIÓN Y VERSIÓN

¡El lenguaje de programación en el que se escribió este código es Python! específicamente en su versión 3.11.5. Python es un lenguaje de programación interpretado, lo que significa que se requiere tener Python 3.11.5 o una versión posterior de Python 3 instalada en tu sistema para ejecutar el código con éxito. Puedes descargar Python 3.11.5 desde el sitio web oficial de Python (<https://www.python.org/downloads/>) o mediante un administrador de paquetes si está disponible en tu sistema.



```
MINGW64:/c/Users/paola
paola@DESKTOP-0V3D47U MINGW64 ~
$ python --version
Python 3.11.5
paola@DESKTOP-0V3D47U MINGW64 ~
$
```

## SISTEMA OPERATIVO

El código se desarrolló en un entorno de sistema operativo Windows y se ejecutó utilizando la terminal Git Bash. Sobre todo porque Git Bash proporciona una emulación de terminal que permite utilizar comandos similares a los de Unix en sistemas Windows.

Para ejecutar el código con éxito en tu entorno, necesitas:

1. Asegúrate de que tienes Python 3.11.5 o una versión posterior de Python 3 instalada en tu sistema Windows. Puedes descargar Python desde el sitio web oficial (<https://www.python.org/downloads/>).
2. Utilizar Git Bash como tu terminal. Puedes descargar e instalar Git Bash desde el sitio web oficial de Git (<https://gitforwindows.org/>).
3. Navegar a la ubicación donde se encuentra el archivo de código Python utilizando el comando "cd" en Git Bash.
4. Ejecutar el código Python utilizando el comando "python" seguido del nombre del archivo Python en Git Bash.

## EJECUCIONES

```
cine_bueno.py > ...
1 #####Proyecto 2#####
2 ###Sistemas Operativos Semestre 2024-1
3 #####Integrantes:
4 #####Hernandez Ortiz Jonathan Emmanuel
5 #####Pérez Avin Paola Celina de Jesús.
6
7
8 #####Codigo: Programa desarrollado en Python para una solución mediante Proceso
9 ##### de Sincronización de una sala en Cine, donde tenemos una capacidad de 10 personas
10 ##### pero tenemos 100 clientes, queriendo ingresar a ver X película, por lo que, no se
11 ##### puede tener mas esas personas en la sala, por eso mediante hilos controlamos
12 ##### la entrada y salida de personas para poder disfrutar de la película.
13
14 #Importamos las bibliotecas a ocupar en el programa
15 import threading
16 import os,time,random
17 from colorama import Fore #Biblioteca para darle color a los textos en Terminal Console
18 #Colores posibles para cada cliente y darle color a la terminal
19 # Rojo Blanco Amarillo Verde Azul Magenta Cyan
20 color =[Fore.RED,Fore.WHITE,Fore.YELLOW,Fore.GREEN,Fore.BLUE,Fore.MAGENTA,Fore.CYAN]
21 # Creamos un semáforo para controlar el acceso a los asientos de la sala del cine.
22 semaforo_cine = threading.Semaphore(10) # Supongamos que hay 10 asientos disponibles.
23
24 # Función que simula el proceso de un espectador viendo una película.
25 def ver_pelicula(espectador_id):
26     #hacemos un pequeño retardo para que no imprima la info de "golpe" y no podamos apreciar la asignación de asi
27     time.sleep(5)
28     #Generamos un color aleatoriamente para cada Cliente
29     mi_color = color[random.randint(0, 6)]
30     #Imprimimos su color asignado y al principio todos están en la fila para disfrutar
```

Espectador 1 está en la fila del cine.

Espectador 2 está en la fila del cine.

Espectador 1 ha ocupado un asiento y está viendo la película.  
¡A disfrutar!

Espectador 3 está en la fila del cine.

Espectador 2 ha ocupado un asiento y está viendo la película.  
¡A disfrutar!

Espectador 4 está en la fila del cine.

Espectador 3 ha ocupado un asiento y está viendo la película.  
¡A disfrutar!

Espectador 5 está en la fila del cine.

Espectador 1 ha terminado de ver la película y ha dejado un asiento.  
Muy bonita la película... :)

Espectador 4 ha ocupado un asiento y está viendo la película.  
¡A disfrutar!

Espectador 2 ha terminado de ver la película y ha dejado un asiento.  
Muy bonita la película... :)

Espectador 6 está en la fila del cine.

Espectador 5 ha ocupado un asiento y está viendo la película.  
¡A disfrutar!

Espectador 3 ha terminado de ver la película y ha dejado un asiento.  
Muy bonita la película... :)

Espectador 7 está en la fila del cine.

```
MINGW64:/c/Users/paola/Downloads
Espectador 46 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 50 está en la fila del cine.

Espectador 49 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 47 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 51 está en la fila del cine.

Espectador 50 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 50 fue al baño.

Espectador 48 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 52 está en la fila del cine.

Espectador 51 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 49 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 53 está en la fila del cine.

Espectador 52 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 54 está en la fila del cine.

Espectador 53 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 50 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 51 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 55 está en la fila del cine.
```

```
MINGW64:/c/Users/paola/Downloads

Muy bonita la película... :)

Espectador 86 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 84 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 88 está en la fila del cine.

Espectador 87 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 85 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 89 está en la fila del cine.

Espectador 88 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 86 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 90 está en la fila del cine.

Espectador 89 ha ocupado un asiento y está viendo la película.
¡A disfrutar!

Espectador 87 ha terminado de ver la película y ha dejado un asiento.
Muy bonita la película... :)

Espectador 91 está en la fila del cine.
```