

Sistemas Operativos  
**Proyecto 2:** Una situación cotidiana  
paralelizable

02/10/2019

Equipo:  
**Moreno Morua Luis Alberto**  
**Ramírez Ceseña Angel Orlando**

## ***Manejando un restaurante.***

Para este proyecto creamos un programa que simula un restaurante con 6 mesas, dos meseros, un cajero y un cocinero.

Puede haber hasta 6 clientes dentro del restaurante, uno por cada mesa.

Originalmente el programa se planeo para mostrar dibujitos en consola, como la imagen de la derecha, pero sólo quedo en un prototipo que igualmente anexamos al proyecto.

El prototipo genera algunos dibujos de monitos en las mesas, pero antes de terminar se atora. Así que la versión final del programa termino siendo texto escrito en consola, que narra lo que esta pasando en el restaurante.



**Lenguaje y entorno:** C++ 2011 para implementar hilos y mecanismos de sincronización, y C para todo lo demás.

Para compilarlo en MinGW necesitamos indicarle que vamos a usar C++11 e hilos, pasándole `"-std=c++11 -pthread"` como argumentos.

Nuestro programa sólo corre en Windows debido a la biblioteca `<Windows.h>`

## **Estrategia de sincronización:**

Empezando por el restaurante, usamos un semáforo para evitar que haya más de 6 clientes dentro, y tenemos un mutex en la puerta para que pasen de uno por uno.

Afuera del restaurante también tenemos una cola de personas esperando, esta también usa un semáforo, para evitar que haya más de 5 personas esperando fuera. Cuando hay 5 personas esperando afuera, nuestro hilo creador de clientes se

detiene, y sólo se reactiva al salir un cliente del restaurante. Esto evita que consumamos mucha memoria, aunque originalmente fue pensado porque máximo podíamos dibujar 5 personitas esperando fuera del restaurante.

Los meseros están protegidos por un mutex para evitar que el cocinero o los clientes los interrumpen mientras están realizando una tarea. También están protegidos por una variable condicional, que usan para evitar que los interrumpen al encargar un pedido al cocinero.

El cocinero está protegido por un mutex, para evitar que los dos meseros le encarguen un pedido al mismo tiempo. Cuando ya hizo un pedido y quiere mandárselo al mesero, usa dos mutex para comunicarse con el mesero, de una forma similar a *Rendezvous* (o que a menos nos pareció similar).

El cajero también usa dos mutex de forma similar para comunicarse con el cliente al pagar.

Por último, los clientes, al igual que los meseros, el cocinero y el cajero, usan mutex y variables condicionales para pausar su ejecución cuando esperan por algo o no tienen nada que hacer.

Los hilos también usan banderas para comunicarse entre ellos el estado de las cosas. Por ejemplo, cuando el mesero sirve comida en la mesa levanta la bandera de “comidaEnMesas”, para que los clientes puedan revisar si hay comida en su mesa.

## **El programa:**

Casi al inicio del código definimos la constante “*maxClientes*”, que nos permite modificar el número de clientes que creará y atenderá nuestra simulación. Por default está puesto en 10.

Lo primero que hacemos es lanzar los hilos principales: creamos dos meseros, un cocinero, un cajero y a mi hilo generador de clientes.

Los clientes son los que van a activar a mis demás hilos, y van a pasar por varios “estados” antes de desaparecer:

- Entrar al restaurante
- Pedir comida
- Comer
- Pagar
- Salir del Restaurante

Cuando entran al restaurante, revisan arreglo de banderas "*mesas[6]*", que nos dice si una mesa esta ocupada (1) o libre (0). Depende del numero de mesa, es el mesero que los atiende. Las primeras 3 es mesero A y mesero B los 3 restantes.

Cuando quieren pedir comida, revisan la bandera del mesero que les corresponde (*flagMeseroA* o *flagMeseroB*), si esta en 'n', intentan agarrar su mutex para decirle su pedido. Si esta en otra letra entonces el hilo del cliente se bloquea, y espera a que el mesero los despierte usando una variable condicional que todos los clientes tienen en común (*clienteEsperaPedir*).

Una vez pidió su comida, el hilo del cliente se vuelve a bloquear, y espera que el mesero lo despierte usando la variable condicional "*esperandoComida*". Cuando el mesero pone comida en alguna mesa, despierta a todos los clientes para que revisen si hay comida en su mesa usando el arreglo de banderas "*comidaEnMesas[6]*". Si no había comida en su mesa vuelven a dormirse.

Cuando termina de comer, vuelve a poner en 0 la bandera de comida que corresponde a su mesa.

Ahora va a intentar pagar, así que intenta hacerse del mutex del cajero. Luego modifica la bandera del cajero de no ocupado (n), a ocupado (o), y lo despierta usando su variable condicional (*despiertaCajero*), y se ve obligado a esperarlo, porque intenta hacerse con el mutex "*dinero*", pero el cajero lo tiene, y sólo lo suelta una vez que hizo el cobro. En este paso el Cliente también pone en 0 la bandera que corresponde a la mesa en la que estaba.

Por último, el cliente sale del restaurante, para hacerlo sólo disminuye en uno el contador de "*genteSentada*", y libera la memoria que estaba ocupando su estructura.

La comunicación de los meseros con los clientes o el cocinero se da gracias a su bandera *flagMeseroA* o *flagMeseroB*. Cuando alguno quiere llamarlo, se hace del mutex del mesero (*meseroAtiendeA* / *meseroAtiendeB*), le pone algún valor a su bandera y lo despierta usando su variable condicional "*meseroVenA* / *meseroVenB*". Una vez despierto, el hilo mesero revisa el parámetro que le dejaron en su bandera para saber que hacer. Siendo "t" tomar pedido del cliente, "r" recoger pedido hecho por el cocinero, y "m" terminar la ejecución de su hilo.

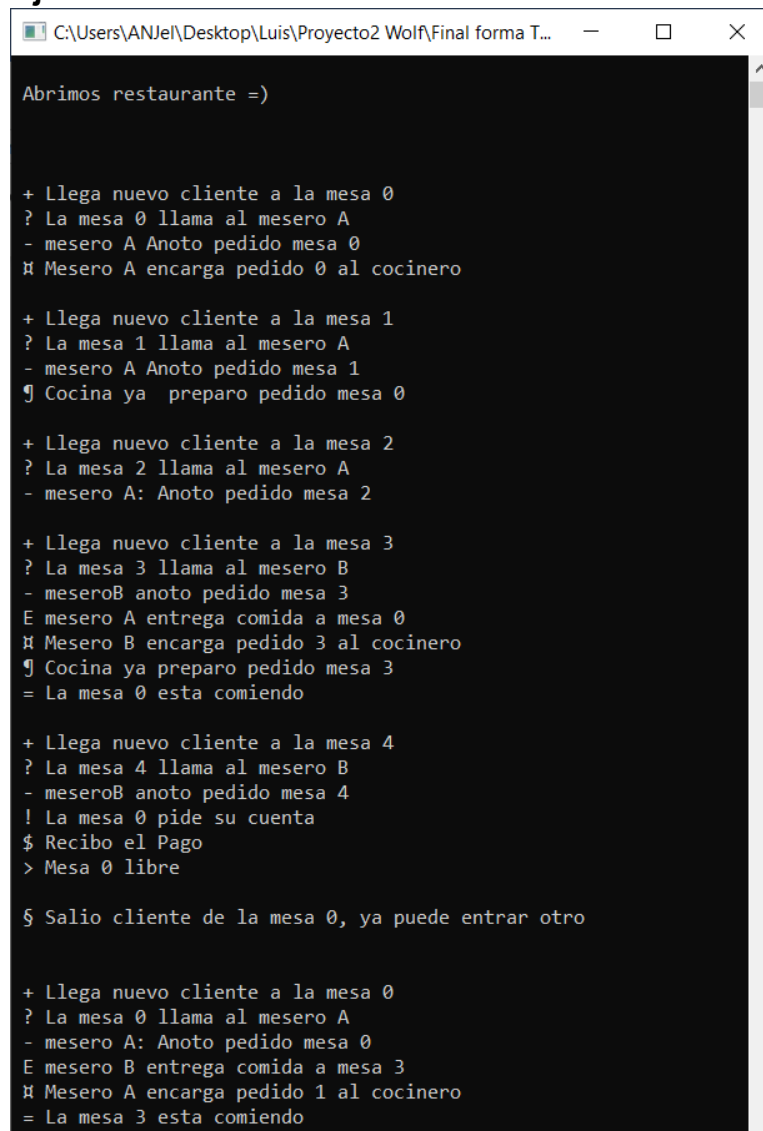
En el caso de que tenga que tomar un pedido, el mesero lanza otro hilo que va a estar al pendiente del cocinero para hacerle el pedido cuando se desocupe, de esa forma el mesero puede seguir libre para tomar pedidos de los demás clientes.

El cocinero funciona de una forma muy similar. Cuando el mesero lo quiere usar toma su mutex “cocineroA”, y cambia su bandera. Pero aquí, el cocinero usa valor del parámetro para saber que mesero le hablo, de esa forma cuando acabe de hacer el pedido sabrá a que mesero entregárselo.

**Problemas no resueltos:** Originalmente se pensaba poner 3 cocineros en vez de uno, y hacer que mostrara monitos en vez de texto. Por el momento eso quedo en un prototipo que no dibuja a los meseros ni al cocinero, pero intenta dibujar a los clientes, aunque no dibuja la fila afuera del restaurante y después de dibujar algunos clientes se atora.

Sin embargo, las pruebas hechas a la versión del proyecto usando texto han funcionado bien.

### Ejecución:



```
Abrimos restaurante =)

+ Llega nuevo cliente a la mesa 0
? La mesa 0 llama al mesero A
- mesero A Anoto pedido mesa 0
M Mesero A encarga pedido 0 al cocinero

+ Llega nuevo cliente a la mesa 1
? La mesa 1 llama al mesero A
- mesero A Anoto pedido mesa 1
M Cocina ya preparo pedido mesa 0

+ Llega nuevo cliente a la mesa 2
? La mesa 2 llama al mesero A
- mesero A: Anoto pedido mesa 2

+ Llega nuevo cliente a la mesa 3
? La mesa 3 llama al mesero B
- meseroB anoto pedido mesa 3
E mesero A entrega comida a mesa 0
M Mesero B encarga pedido 3 al cocinero
M Cocina ya preparo pedido mesa 3
= La mesa 0 esta comiendo

+ Llega nuevo cliente a la mesa 4
? La mesa 4 llama al mesero B
- meseroB anoto pedido mesa 4
! La mesa 0 pide su cuenta
$ Recibo el Pago
> Mesa 0 libre

$ Salio cliente de la mesa 0, ya puede entrar otro

+ Llega nuevo cliente a la mesa 0
? La mesa 0 llama al mesero A
- mesero A: Anoto pedido mesa 0
E mesero B entrega comida a mesa 3
M Mesero A encarga pedido 1 al cocinero
= La mesa 3 esta comiendo
```

+ Llega nuevo cliente a la mesa 5  
? La mesa 5 llama al mesero B  
- meseroB anoto pedido mesa 5  
👩🍳 Cocina ya preparo pedido mesa 1  
! La mesa 3 pide su cuenta  
\$ Recibo el Pago  
> Mesa 3 libre

§ Salio cliente de la mesa 3, ya puede entrar otro

+ Llega nuevo cliente a la mesa 3  
? La mesa 3 llama al mesero B  
- meseroB anoto pedido mesa 3  
E mesero A entrega comida a mesa 1  
👨🍳 Mesero B encarga pedido 4 al cocinero  
👩🍳 Cocina ya preparo pedido mesa 4  
= La mesa 1 esta comiendo  
! La mesa 1 pide su cuenta  
\$ Recibo el Pago  
> Mesa 1 libre

§ Salio cliente de la mesa 1, ya puede entrar otro

+ Llega nuevo cliente a la mesa 1  
? La mesa 1 llama al mesero A  
- mesero A: Anoto pedido mesa 1  
E mesero B entrega comida a mesa 4  
👨🍳 Mesero A encarga pedido 2 al cocinero  
= La mesa 4 esta comiendo  
👩🍳 Cocina ya preparo pedido mesa 2  
! La mesa 4 pide su cuenta  
\$ Recibo el Pago  
> Mesa 4 libre

§ Salio cliente de la mesa 4, ya puede entrar otro

+ Llega nuevo cliente a la mesa 4  
? La mesa 4 llama al mesero B  
- meseroB anoto pedido mesa 4  
E mesero A entrega comida a mesa 2  
👨🍳 Mesero B encarga pedido 5 al cocinero  
👩🍳 Cocina ya preparo pedido mesa 5  
= La mesa 2 esta comiendo  
! La mesa 2 pide su cuenta  
\$ Recibo el Pago  
> Mesa 2 libre

§ Salio cliente de la mesa 2, ya puede entrar otro

E mesero B entrega comida a mesa 5  
👨🍳 Mesero B encarga pedido 3 al cocinero  
👩🍳 Cocina ya preparo pedido mesa 3  
= La mesa 5 esta comiendo  
! La mesa 5 pide su cuenta  
\$ Recibo el Pago  
> Mesa 5 libre

§ Salio cliente de la mesa 5, ya puede entrar otro

```
E mesero B entrega comida a mesa 3
M Mesero B encarga pedido 4 al cocinero
C Cocina ya preparo pedido mesa 4
= La mesa 3 esta comiendo
! La mesa 3 pide su cuenta
$ Recibo el Pago
> Mesa 3 libre

$ Salio cliente de la mesa 3, ya puede entrar otro

E mesero B entrega comida a mesa 4
M Mesero A encarga pedido 1 al cocinero
= La mesa 4 esta comiendo
C Cocina ya preparo pedido mesa 1
! La mesa 4 pide su cuenta
$ Recibo el Pago
> Mesa 4 libre

$ Salio cliente de la mesa 4, ya puede entrar otro

E mesero A entrega comida a mesa 1
M Mesero A encarga pedido 0 al cocinero
= La mesa 1 esta comiendo
C Cocina ya preparo pedido mesa 0
! La mesa 1 pide su cuenta
$ Recibo el Pago
> Mesa 1 libre

$ Salio cliente de la mesa 1, ya puede entrar otro

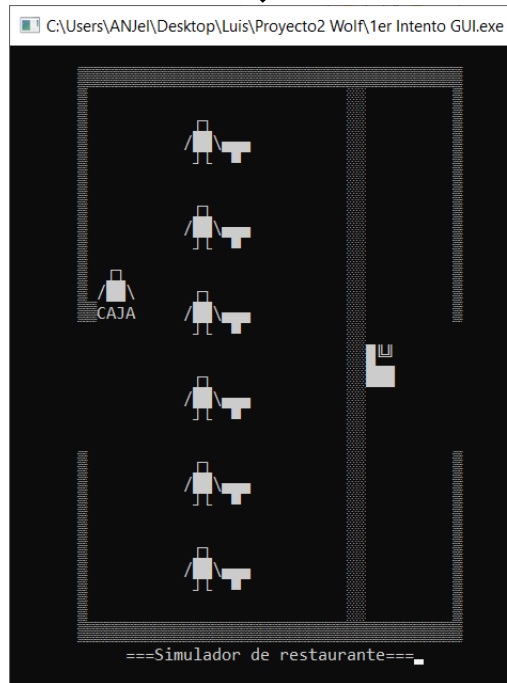
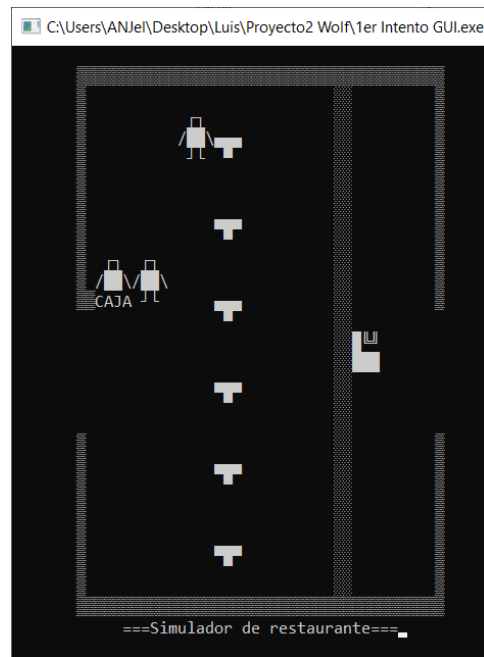
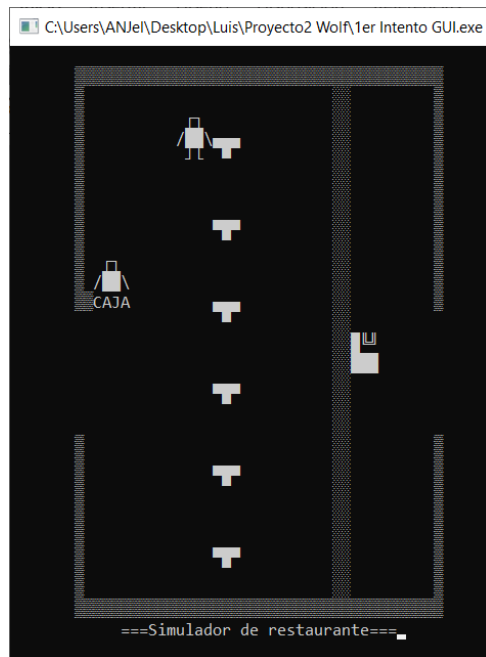
E mesero A entrega comida a mesa 0
= La mesa 0 esta comiendo
! La mesa 0 pide su cuenta
$ Recibo el Pago
> Mesa 0 libre

$ Salio cliente de la mesa 0, ya puede entrar otro

>> Ya atendimos a todos los clientes. Cerrando restaurante

Listo, presiona ENTER para terminar el programa_
```

***Ejecución del prototipo:***



Una vez llena las 6 mesas se queda atorado, lo cual no tiene sentido, porque, aunque no está dibujando a los meseros ni al cocinero, siguen estando en el código y siguen funcionando.