Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Sistemas Operativos

Proyecto 2: : Una situación cotidiana paralelizable

Profesor: Gunnar Eyal Wolf Iszaevich

Alumnos:

Bárcenas Avelar, Jorge Octavio

Reza Chavarria, Sergio Gabriel

Contenido del Proyecto 2: Una situación cotidiana paralelizable

[Proyecto 2: Una situación cotidiana paralelizable 3](#_Toc37692267)

[Objetivos 3](#_Toc37692268)

[Identificación y Descripción del problema 3](#_Toc37692269)

[Mecanismos utilizados 4](#_Toc37692270)

[Lógica de Operación 4](#_Toc37692271)

[Entorno de desarrollo del programa 5](#_Toc37692272)

[Notas de Uso 5](#_Toc37692273)

[Pruebas de escritorio 5](#_Toc37692274)

# Proyecto 2: Una situación cotidiana paralelizable

# Objetivos

* Identificar algún evento de la vida cotidiana, en la cual este evento sea paralelizable y utilizarlo como caso de estudio para el proyecto,
* Comprender en donde se genera las concurrencias en el caso de estudio elegido.
* Modelar el problema en un lenguaje de programación deseado y utilizar los mecanismos de sincronización estudiados en el curso.

# Identificación y Descripción del problema

En el caso a desarrollar se planteó la situación de un restaurante. El estado del restaurante consta de una cantidad de meseros, una cantidad de mesas, una cantidad de chefs, una fila de espera y clientes. Imaginamos que para que hubiera un poquito de “competencia” la cantidad de mesas debería ser mayor a la cantidad de meseros, por lo que los meseros deben de atender varias mesas. Los procesos o hilos planteados en el trabajo son los puestos de clientes (e invitados), meseros y chefs.

Los clientes son los que serán acomodados en la fila hasta que llegue su turno. El mesero llevará al cliente. El cliente llevará a los invitados. Los invitados son personas que no esperaron en la fila, sino ya venían con el cliente.

El mesero solo podrá tomar la orden cuando el cliente y los invitados ya decidieron que ordenar, para no tener varios viajes. El mesero llevará la orden al chef, los cuales se encargarán de realizar la orden, mientras que el mesero, si es necesario, atenderá más mesas. Cuando el chef tenga toda la orden completa, llamará al mesero para entregarla y continuará con las ordenes pendientes. Al entregarla a los comensales, estos tomarán su tiempo para la degustación de su comida. Cuando todos los comensales de la mesa terminen su comida, pedirán y pagarán la cuenta. Cuando terminen se liberará esta mesa para su uso. Todo el proceso se repetirá hasta que ya no haya comensales esperando en la fila.

Los casos de concurrencia se encuentran en los casos de los chefs y las ordenes, el chef debe de tomar su tiempo para preparar la orden, por lo que solo puede tomar una orden a la vez. Al finalizar y entregar la orden continuará con la siguiente orden pendiente. Además, que se encuentran varios chefs en la cocina, así se puede controlar la cantidad de ordenes pendientes.

# Mecanismos utilizados

Los mecanismos utilizados para la sincronización de procesos fueron

* Mutex: Utilizado para las funciones que tienen los meseros, y en la fila de clientes del restaurante.
* Semaforos: Utilizado para los meseros disponibles, en las ordenes, en el estado del cliente ya comiendo.
* Multiplex: Utilizado para la cantidad de chefs, meseros, y mesas disponibles para ser utilizados.

# Lógica de Operación

Las variables globales utilizadas serán:

* La cantidad de meseros, mesas, chefs y clientes.
* Multiplex para los limitantes
* Lista de meseros disponibles, clientes esperando en la fila del restaurante y de Chefs disponibles. Estas listas son utilizadas para consulta. Así tenemos registro de que proceso puede ser utilizado.

A partir de la asignación del número de entidades (chefs, meseros, mesas, clientes) se crearán las instancias correspondientes a estos valores.

Cuando se dé la declaración de clientes, el programa iniciará. Los meseros interactuarán con el cliente para la asignación de mesa, pedido y entrega de orden y pago del servicio. Estas acciones utilizaran el apoyo de mutex para la atención individual de los meseros con la mesa de los clientes.

Los clientes involucrados tendrán asignados varios procesos que serán los invitados de los clientes. Estos clientes interactuarán en conjunto con el cliente principal para las ordenes a los meseros.

Los chefs utilizarán mutex para la obtención, preparación y entrega de las ordenes de las mesas que el mesero lleva.

La cantidad de mesas, los meseros, y los chefs actuarán como multiplex para el uso limitado de los hilos correspondientes, así evitando atención a clientes sin mesa.

# Entorno de desarrollo del programa

En lenguaje de desarrollo utilizado fue Python (Versión 3.8.2). Ya que ya habíamos trabajado con anterioridad los conceptos de los sincronizadores de procesos. Se trabajaron en 2 sistemas operativos.

* Linux Mint 19.1
* Windows 10

Para el proyecto se utilizó la biblioteca Tkinter, el cual es un módulo estándar para interfaces gráficas para Python.

# Notas de Uso

Para ejecutar el programa debe primero haber instalado Tkinter. Para Windows ya está descargado. Para distribuciones de Linux, se debe ingresar el siguiente comando para instalar

Para esto fue necesario descargarlo. Se debe ingresar el comando en las distribuciones Linux.

* sudo apt-get install python3-tk (Python 3)
* sudo apt-get install python-tk (Python 2.7)

Para la ejecución del archivo se utiliza el comando, en la ubicación en donde se encuentre el archivo

* python3 proyecto\_2.py

Al poner el comando se desplegará la ventana del proyecto. En esta ventana se tendrá que ingresar la cantidad de meseros, mesas, chefs y clientes en los espacios correspondientes. Al tener los valores deseados se deberá oprimir el botón de Enviar. En caso de ingresar valores incorrectos la ventana derecha desplegará el mensaje de “Valores Incorrectos”

Al enviar las cantidades, la caja de texto registrará todos los procesos realizados por las entidades y se podrá tener el registro total del restaurante.

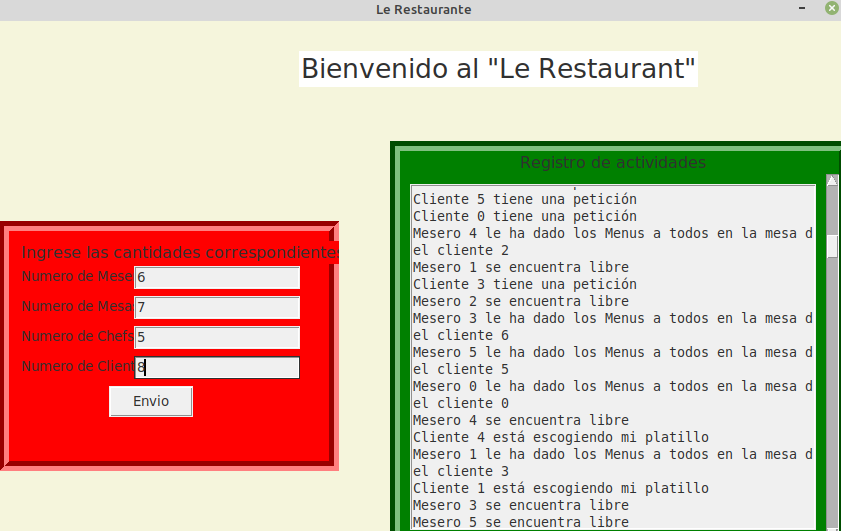
Para realizar diferentes pruebas se deberá realizar el procedimiento de ejecución de nuevo, esto cerrando la ventana de la interfaz.

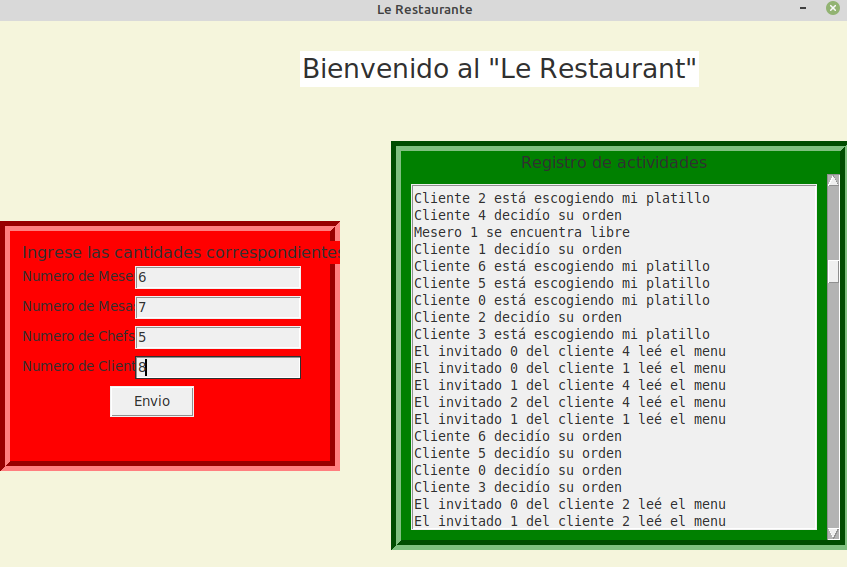
# Pruebas de escritorio

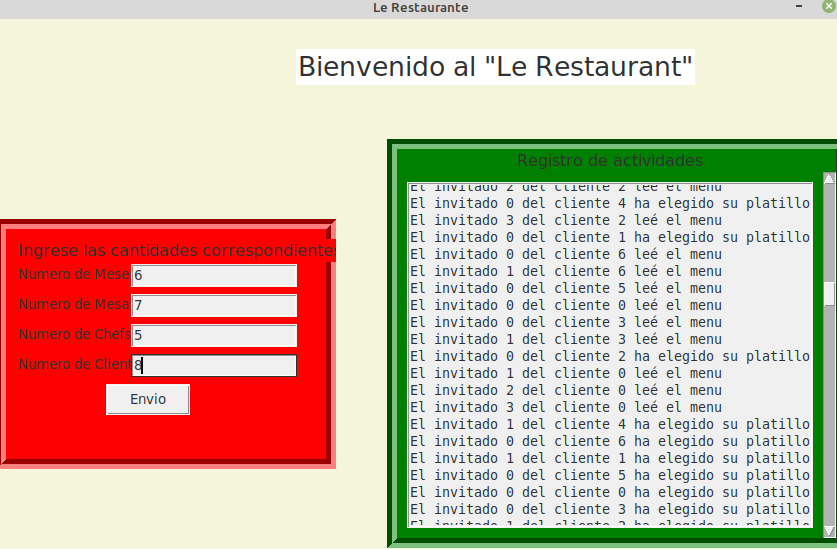
Corrida uno:

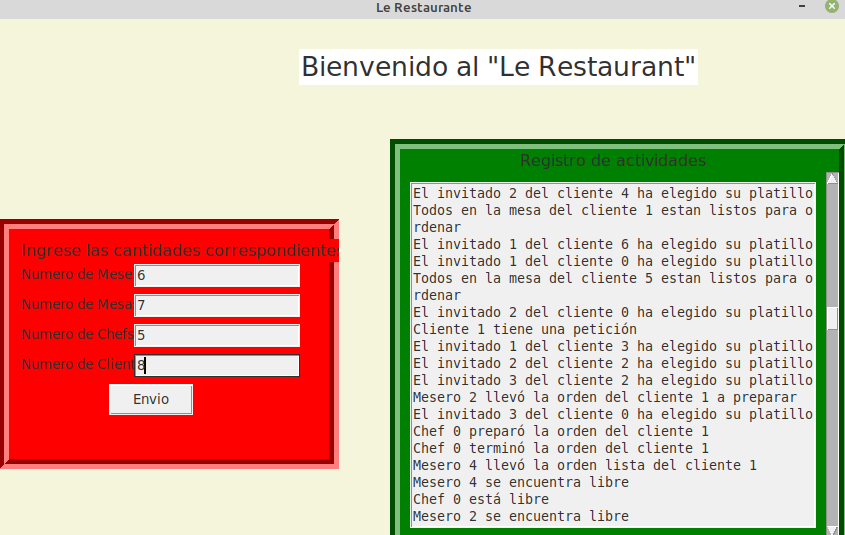


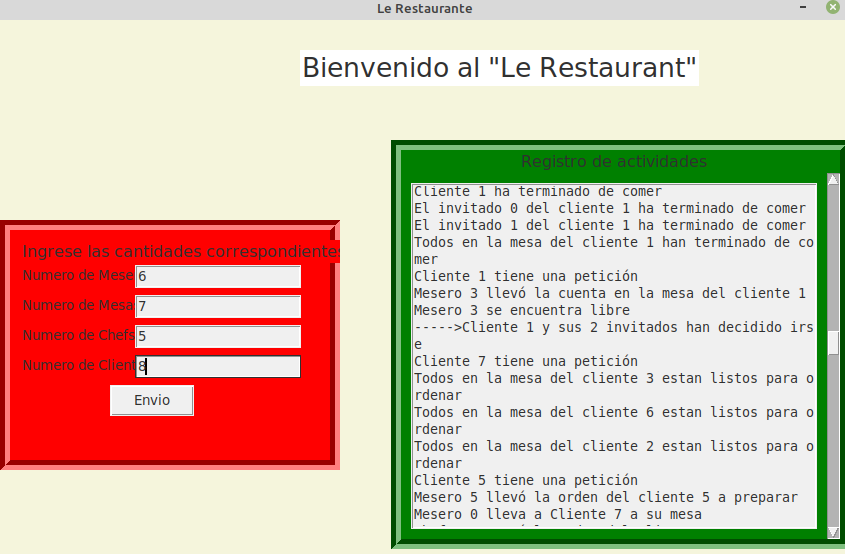


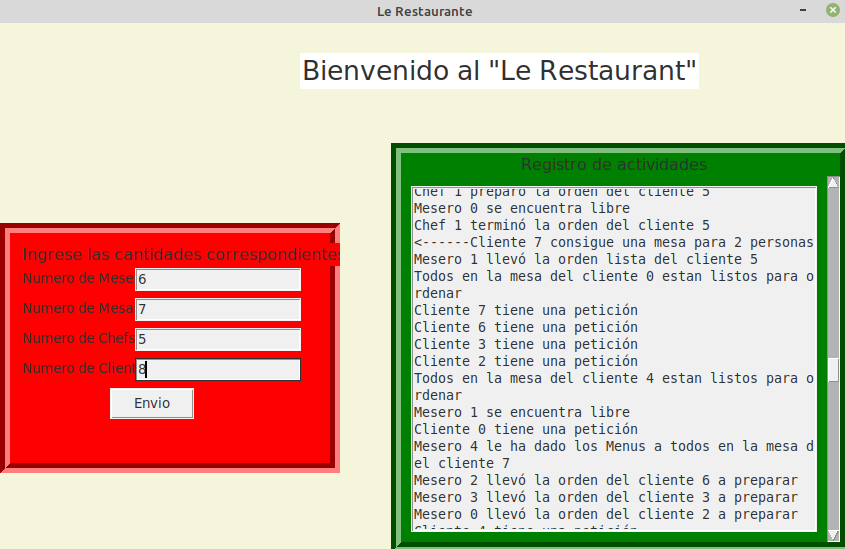


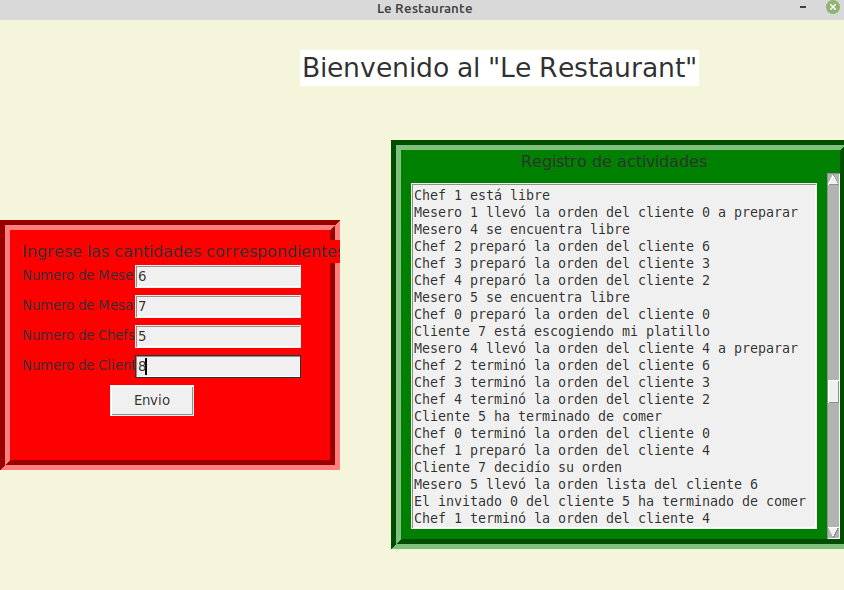


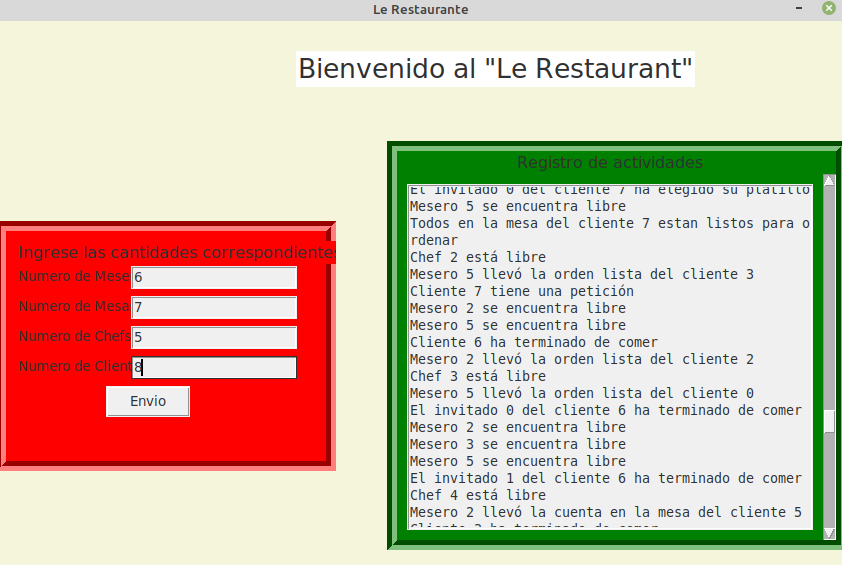


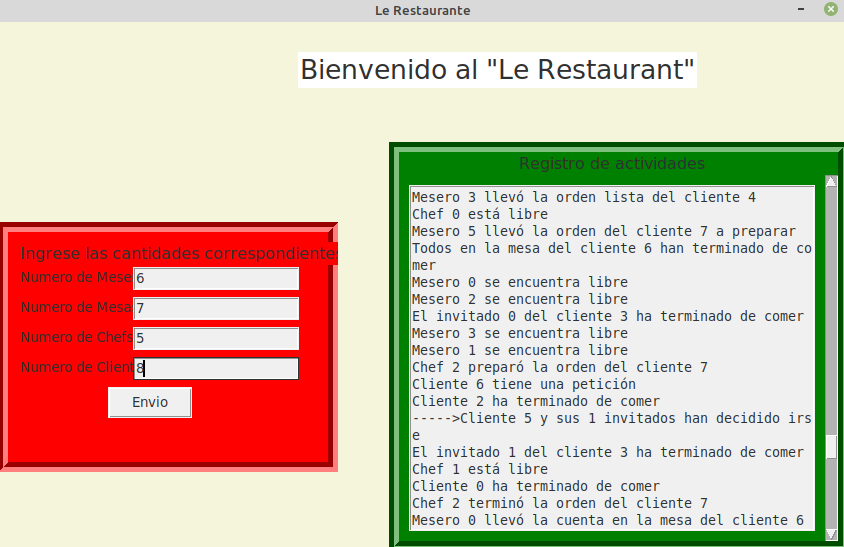


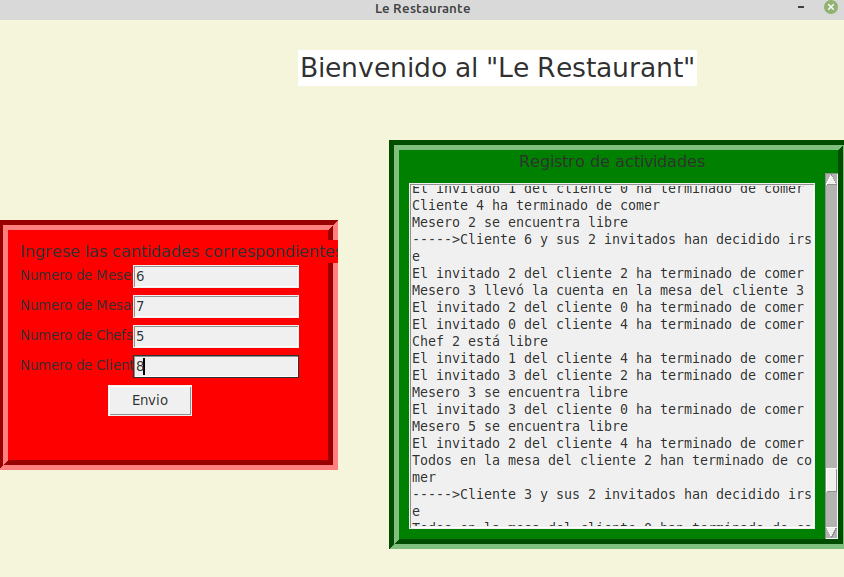


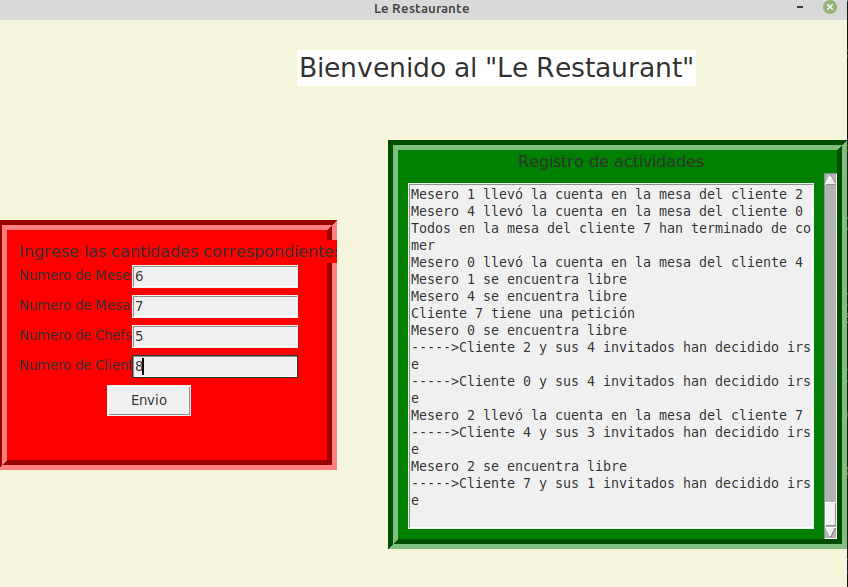












Corrida dos:



