# Modelado y resolucion de Problemas del proyeto de Sistemas Operativos

Valentino Russo et al

# Problemas de comunicación y procesos

## 1 Pumper Nic

Para este problema consideramos que las siguientes resticciones.

1. Una cantidad limitada de clientes en el restaurante a la vez.
2. Todas las ordenes se hacen al empleado de las ordenes
3. Los empleados solo pueden realizar una operacion a la vez.
4. Cada cliente debe recibir su comida en una cantidad finita de tiempo.
5. Los clientes deben recibir lo que pidieron
6. Los clientes VIP tienen prioridad en recibir su orden en cuanto a los normales.
7. No utilizar semaforos.

### 1.1 Resolución con pipes

En el primer insciso del enunciado, dadas las resticciones planteamos la solución de una manera tal que al tomar pedir una orden un cliente el proximo no puede pedir hasta que el anterior no haya recibido su pedido, en otras palabras, exclusión mutua para pedir y recibir un pedido. Esto no significa que si un cliente normal hace su pedido, un cliente VIP no pueda hacer la suya pues funcionan con pipes separadas.

Para garantizar que haya una cantidad limitada de clientes dentro del restaurante se utilizo un pipe no bloqueante iniciado con la una cantidad de mensajes equivalente a la cantidad máxima de personas permitidas dentro del restaurante. Entonces para poder entrar al restaurante un cliente debe leer un mensaje de ese pipe y cuando sale debe devolverlo al pipe para que otros clientes puedan entrar.Este pipe es compartido entre clientes VIPs y normales pues el restaurante tiene la misma cantidad de espacio para todo tipo de clientes.

Para implementar la exclusión mutua se hizo algo similar al lo que hizo para implementar la cantidad limitada de clientes en el restaurante y se utilizo un pipe que inicia con solo un mensaje y para pedir la orden un cliente debe leer de este pipe y cuando recibe su orden debe devolverlo al pipe. En este caso el pipe del turno no es compartido entre los tipos de clientes, es decir, existe un pipe que le permite hacer una orden a un cliente normal y otro que se la permite hacer a un cliente VIP.

Con este sistema se garantizan todas las restricciones dadas, pero tiene un problema y es que no aprovecha el paralelismo al máximo para pedir ordenes pues solo 2 clientes pueden pedir ordenes y esperar a la vez, además el empleado de ordenes provoca un cuello de botella a la hora de tomar ordenes pues solo toma de a una. Una mejor manera de resolverlo seria utilizar un buffer con las ordenes de varios clientes y enviar un conjunto de $M$ de ordenes al empleado de ordenes y que este se encargue de repartirla entre los empleados. Y luego los clientes esperen en otro buffer su orden, así se maximiza la cantidad de ordenes pedidas a la vez y se minimiza la cantidad de llamadas al empleado de ordenes.

Otro detalle a aclarar es que a primera vista parece redundante usar una cantidad limitada de personas dentro del restaurante si solo dos pueden ser atendidas y esperar a la vez. Pero esto es adrede pues en el enunciado se menciona un comportamiente de los clientes que dice *"Cada cliente espera ser atendido en una cola, aunque si hay mucha gente puede decidir marcharse y volver más tarde"*, esto lo interpretamos como que un cliente al llegar al restaurante intenta entrar y si no puede hacerlo, se va a hacer otra cosa y luego de un tiempo vuelve a intentar entrar.Por este motivo es que el pipe que se utiliza para entrar al restaurante es no bloqueante y por ese motivo es que esta la restricción, sino simplemente se permitiria que cualquier cantidad de gente entre al restaurante y espere a ser atendida.

Los empleados se comunican a través de pipes con el empleado de ordenes y además los empleados que no son el de ordenes despachan la comida hecha en un mismo pipe. En total se utilizaron más de 10 pipes en todo el sistema, una cantidad bastante alta pero que fue necesaria para resolver el problema.

Por último se hizo que cuando un cliente haga una orden ya sea VIP o no, este le envie una señal al empleado de ordenes indicanto que hay una orden por atender. Del mismo modo cuando el empleado de ordenes, exitosamente envía a los otros empleados el pedido, se le envia una confirmación al cliente que este espera por recibir.

## 1.2 Resolución con cola de mensajes

Para esta parte del insciso se utilizo un sistema similar al de los pipes donde hay una cantidad limitada de personas en un momento dado dentro del restaurante, además cada cliente debe esperar a la confirmación del empleado que toma las ordenes.

La diferencia con la implementación de los pipes es que el próximo cliente puede hacer su pedido cuando el anterior hizo el suyo y recibió su confirmación, mejorando así la concurrencia.

Para implementar el restaurante con cola de mensajes se utilizaron 4 colas, si bien se podría haber hecho con una sola, consideramos que era más claro utilizar 4, la primera

es para los pedidos que hacen los clientes ya sean VIP o no, la segunda es para las señales que se envían (confirmación, hay pedidos,lleno), la tercera es para la comunicación entre el empleado de ordenes los demás empleados y la cuarta para despachar los pedidos completados a los clientes esperando.

Este problema se resuelve muy facilmente con cola de mensajes y mas elegantemente ya que las restricciones se modelan mejor. Además se mejoro la concurrencia pues ahora un cliente puede hacer su orden sin tener que esperar a que el anterior haya sido atendido.

## 2 Mini Shell

Para implementar la versión minimalista de cada comando se utilizaron llamadas al sistema que proveen las distintas librerias de C.

Para implementar el comando "help" y el "exit" directamente se hizo en el archivo principal de la shell. Para identificar los datos ingresados por el usuario, se usa la función strtok primero buscando un salto de linea y luego espacios. Luego de eso se le da el formato correcto para ejecutar el archivo correspondiente.

Para ejecutar los comandos se usa la función *excecvp* con el comando identificado, su ruta fue obtenida anteriormente al generar el formato, y de la misma manera los parametros.

# Problemas de sincronización

## 3 Problema Motos

Marcos escribe este.

## 4 Problema Santa Claus

Este problema presenta una gran similitud con el problema del barbero dormilón planteado por Dijkstra. En este caso en lugar de haber un solo tipo de entidad para despertar a Santa hay dos, los renos y los elfos. Se utilizo una estrategia que hace que Santa espere a los 9 renos primero dentro de un bucle y luego los 3 elfos también dentro de otro bucle. Luego de salir de los bucles vuelve a entrar en un bucle nuevo atendiendo de a un reno a la vez en lo que la literatura lo llaman el *rendezvous* y luego atiende a los elfos, tal cual lo dice el enunciado.

Al resolver este problema notamos que la cantidad de semaforos que se utilizaron era bastante alta por lo que había un alto grado de contención y poca concurrencia. Luego de consultarlo en clase y con nuestros compañeros, mencionaron que también notaron esto por lo que concluímos que debe ser algo parte de la solución.

# Fallo de disponibilidad