Informe Trabajo Práctico 2 - Técnicas de Programación Concurrente (75.59)

Agustín Mezzina (89637) Pablo Rodríguez (93970)

26 de junio de 2014

Índice

1. Introducción

1.1. Descripción del problema

El presente trabajo práctico consiste en una simulación de la dinámica de operación de una estación de servicio. En ella, los operarios deben atender a los clientes entrantes utilizando los recursos de los cuales dispone la estación. El eje del problema es la comunicación y la sincronización de los operarios en el uso de dichos recursos para el correcto funcionamiento del negocio. Esta segunda entrega agrega la problemática del manejo de prioridades en la atención de autos entrantes y el acceso a la caja del lugar.

1.2. Modo de uso

Adjunto a este informe se encuentra el código fuente de la aplicación y un archivo Makefile para realizar la compilación. El sistema debe ser compilado utilizando la directiva "make all", el cuál generará seis ejecutables con los programas que van a interactuar en el sistema. Para comenzar la simulación, debe ejecutarse en una consola el comando "ConcuStation". El sistema pedirá al usuario que ingrese el número de empleados y de surtidores que posee la estación. Luego de ingresados, dará inicio a la simulación. Durante el transcurso de la misma, el usuario puede ingresar dos comandos por entrada estándar:

1. "c": Consultar Caja. Imprime por salida estándar la recaudación actual de la Caja.
2. "q": Finalizar simulación.

2. Procesos

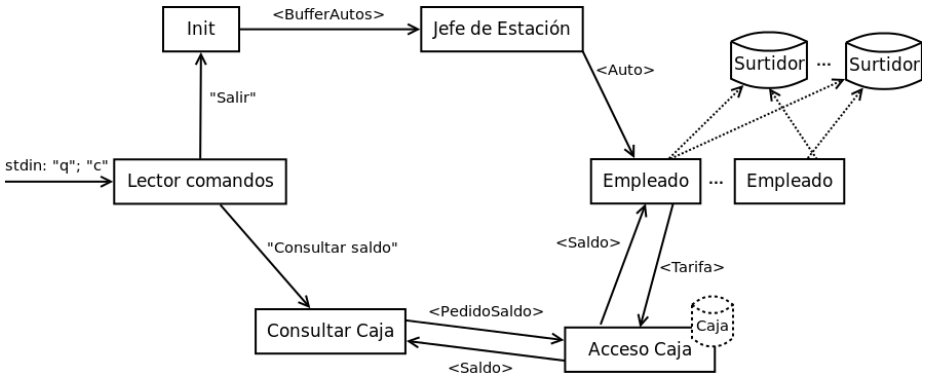
La aplicación se divide en los siguientes procesos:

1. Init: genera un auto cada cierta cantidad aleatoria de tiempo transcurrido y los envía al jefe de estación para su procesamiento. El auto puede ser, también de manera aleatoria, regular o VIP.
2. Jefe de estación: recibe los autos entrantes y los asigna a empleados libres. En caso de no haber empleados libres, los despacha. Prioriza la atención de los autos VIP sobre los regulares.
3. Empleado: asigna autos a surtidores libres. Si no hay surtidores libres, espera a que se libere alguno. Realiza el servicio. Cobra y guarda la recaudación en la caja.
4. Consulta caja registradora: consulta asincrónica sobre el valor de la recaudación en caja.
5. Lector de comandos: escucha comandos entrantes por entrada estándar y toma las medidas correspondientes para responder a ellos.

6. Acceso caja: posee la caja registradora de la estación. Recibe pedidos de los Empleados y de Consultar Caja para operar sobre ella y los atiende dando prioridad a los emitidos por Consultar Caja.

3. Comunicación entre procesos

En el siguiente diagrama, se muestra el esquema de comunicación entre los procesos del sistema. Incluye los datos que intercambian los procesos que se comunican entre sí y los recursos a los que deben acceder de manera concurrente.



4. Problemas típicos de concurrencia

1. Init - Jefe de estación: Productor - Consumidor. El proceso base hace las veces de Productor, ingresando autos nuevos a un buffer compartido donde posteriormente el Jefe de Estación, en este caso Consumidor, los retira según su orden de llegada y los procesa.

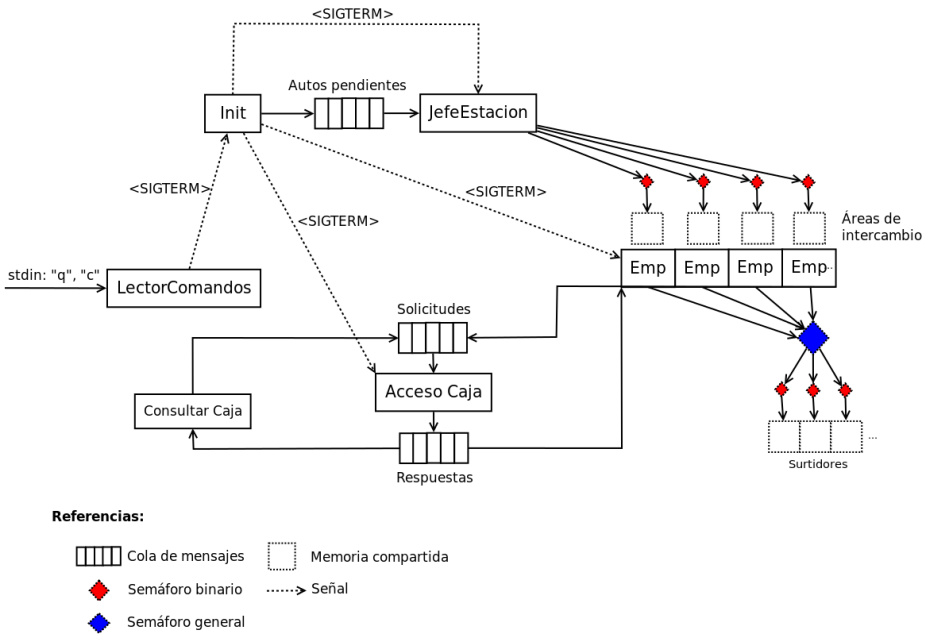
2. Jefe de estación - Empleado(s): Productor - Consumidor. Se presenta un caso similar al anterior, con la variante de que el Productor (Jefe de estación) debe determinar a cuál de los Consumidores (Empleados) enviar el recurso generado.

3. Empleado - Empleado: Sección Crítica. Los procesos deben excluirse mutuamente al ejecutar código que acceda a los recursos Surtidores, ya que el mismo Surtidor no puede ser utilizado por dos Empleados al mismo tiempo.

4. Empleado(s) - Consultar Caja: Sección Crítica. Los procesos deben excluirse mutuamente al acceder a la Caja, ya sea para leer como para escribir.

5. Mecanismos de concurrencia a utilizar

En el siguiente diagrama se muestran los mecanismos de concurrencia utilizados para lograr la comunicación y sincronización de procesos:



5.1. Init - Jefe de estación

El buffer de autos se implementó utilizando una cola de mensajes, de manera que el proceso inicial ingresa los autos al canal a medida que los va produciendo y el jefe de estación los toma para procesarlos. Cada auto se guarda con un valor de mtype diferente, según sea regular o VIP. Esto permite al jefe de estación tomar primero los autos VIP y, en caso de no haber ninguno, atender uno regular.

5.2. Jefe de estación - Empleado(s)

Cada Empleado mantiene con el Jefe de Estación un espacio de intercambio en memoria compartida, donde se guarda información sobre el estado del Empleado (libre u ocupado) y el auto que debe ser atendido. Cada uno de estos espacios de intercambio está protegido con un semáforo binario para lograr exclusión mutua entre el Jefe y el Empleado correspondiente en el acceso al mismo, ya sea para lectura o para escritura.

5.3. Empleado - Empleado

Los Surtidores están guardados en bloques de memoria compartida accesibles a todos los Empleados. Dicho bloque está protegido por un semáforo general que permite el acceso al área de memoria cuando haya algún surtidor libre. Cada espacio de memoria donde se encuentra un surtidor está, a su vez, protegido por un semáforo binario.

5.4. Empleado(s) - Consultar Caja - Acceso Caja

El acceso a la caja es coordinado mediante la utilización de dos colas de mensajes. En la primera, los procesos Consultar Caja y Empleado escriben una solicitud con la operación que quiere realizarse sobre la caja registradora, utilizando cada proceso un número de mtype diferente como identificación. El proceso Acceso Caja toma esas solicitudes, dando prioridad a las emitidas por Consultar Caja y pasando a las emitidas por los Empleados en caso de no haber ninguna, y realiza el cómputo necesario sobre la caja. Acto seguido, escribe la respuesta (que en todos los casos es el saldo actual de la caja) en la segunda cola de mensajes, dirigida al proceso en particular que realizó la solicitud. Los Empleados y Consultar Caja esperan esta respuesta para continuar con su operatoria.

6. Otras consideraciones

6.1. Inicio y finalización

Al iniciar la simulación, el proceso Init crea todas las memorias compartidas, colas de mensajes y semáforos (estos últimos también los inicializa) necesarios para el funcionamiento del sistema. Luego lanza los procesos Jefe de Empleados, Lector de Comandos, Acceso Caja y todos los Empleados. El proceso Consultar Caja es lanzado por el Lector de Comandos cada vez que el usuario ingresa el comando “c”. Este proceso finaliza luego de imprimir el valor de la Caja en pantalla. Cuando el Lector de Comandos recibe el comando “q”, el sistema debe finalizar. Para esto, se envía la señal SIGTERM al proceso Init y este, a su vez, envía una señal similar a todos los Empleados, al Jefe de Estación y al proceso Acceso Caja. Estos procesos manejarán la señal eliminando todos sus recursos y finalizando correctamente. El proceso Init espera la finalización de los demás procesos y luego procede a elminar todos los canales de comunicación creados inicialmente, para luego finalizar.

6.2. Log

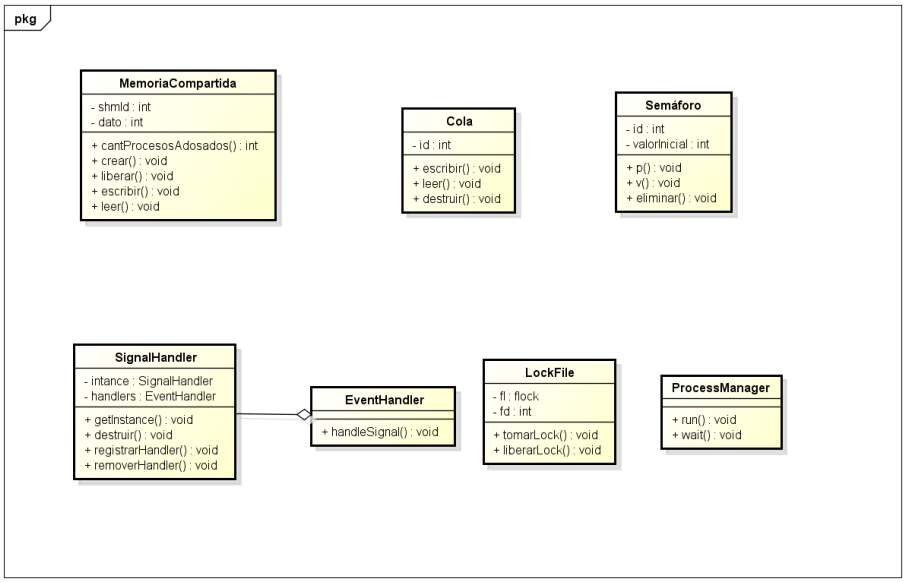
En caso de funcionar en modo Debug, todos los procesos de la simulación informan las acciones que realizan mediante la escritura a un único archivo denominado “ConcuStation.log”. El mismo puede utilizarse para verificar el

funcionamiento del sistema y el orden en el que se realizan las actividades. Para coordinar el acceso al archivo, se utilizan locks de escritura.

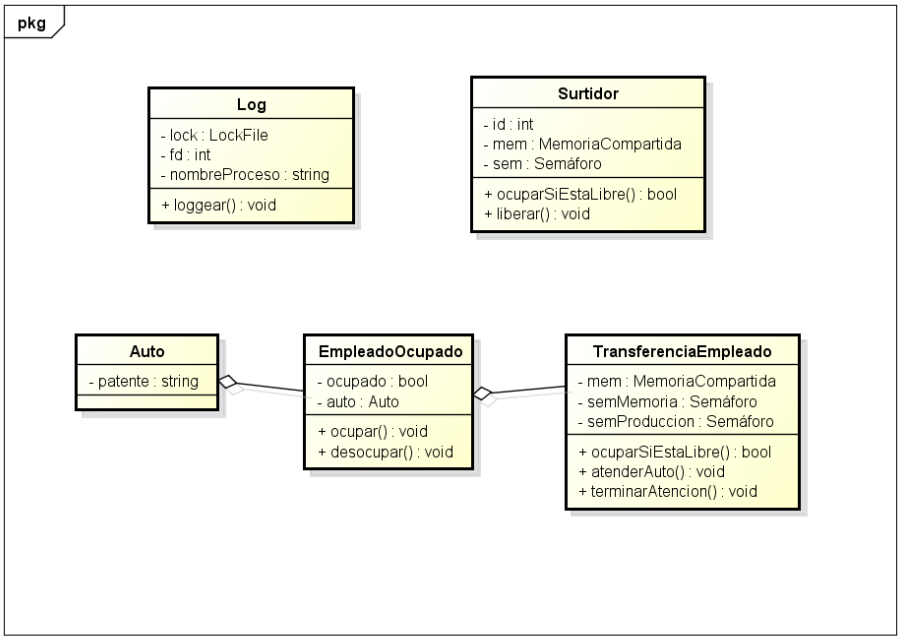
7. Diagramas

7.1. Diagrama de clases

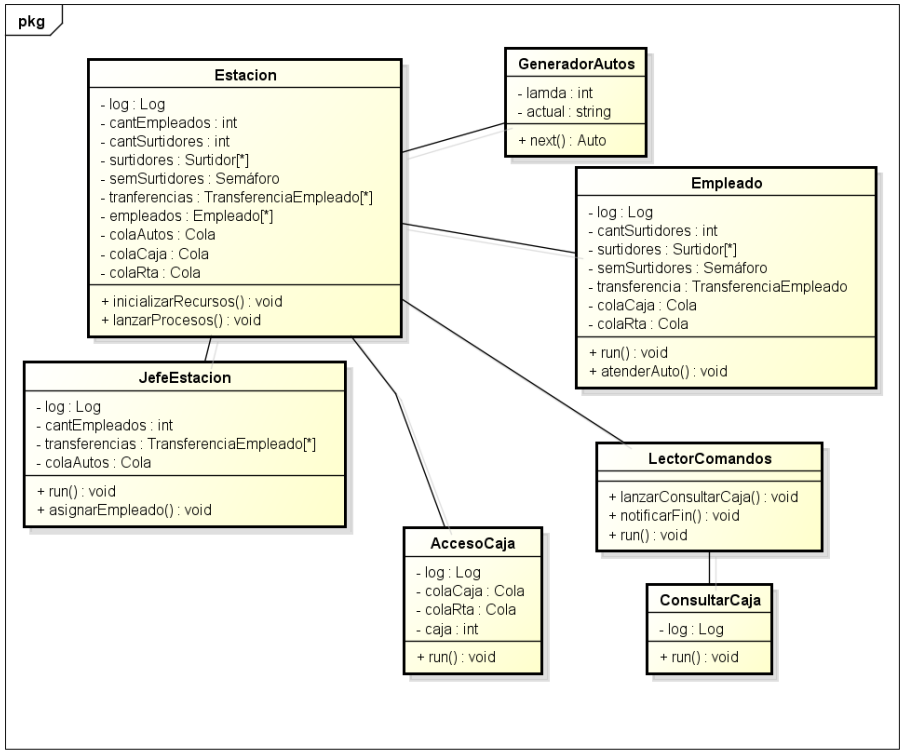
Mecanismos de concurrencia:



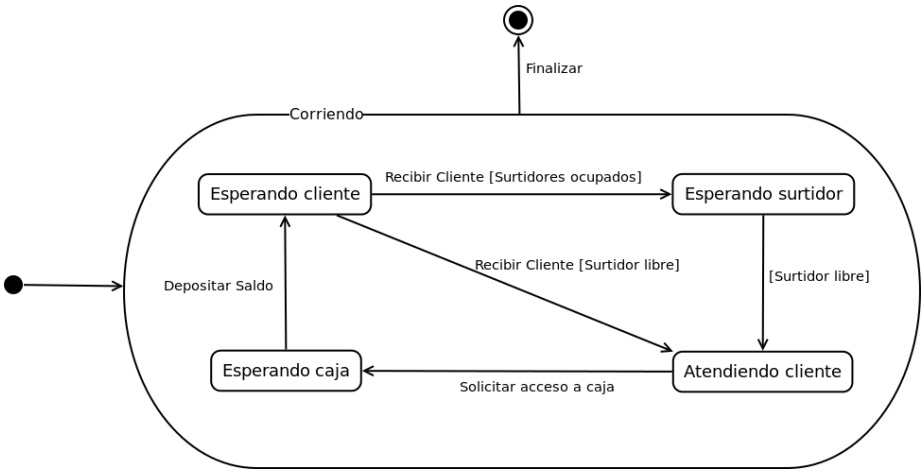
Recursos:


pkg
Log
- lock : LockFile
- fd : int
- nombreProceso : string
+ loggear() : void
Surtidor
- id : int
- mem : MemoriaCompartida
- sem : Semáforo
+ ocuparSiEstaLibre() : bool
+ liberar() : void
Auto
- patente : string
EmpleadoOcupado
- ocupado : bool
- auto : Auto
+ ocupar() : void
+ desocupar() : void
TransferenciaEmpleado
- mem : MemoriaCompartida
- semMemoria : Semáforo
- semProduccion : Semáforo
+ ocuparSiEstaLibre() : bool
+ atenderAuto() : void
+ terminarAtencion() : void
powered by Astah

Procesos:


pkg
Estacion
- log : Log
- cantEmpleados : int
- cantSurtidores : int
- surtidores : Surtidor[*]
- semSurtidores : Semáforo
- tranferencias : TransferenciaEmpleado[*]
- empleados : Empleado[*]
- colaAutos : Cola
- colaCaja : Cola
- colaRta : Cola
+ inicializarRecursos() : void
+ lanzarProcesos() : void
GeneradorAutos
- lamda : int
- actual : string
+ next() : Auto
Empleado
- log : Log
- cantSurtidores : int
- surtidores : Surtidor[*]
- semSurtidores : Semáforo
- transferencia : TransferenciaEmpleado
- colaCaja : Cola
- colaRta : Cola
+ run() : void
+ atenderAuto() : void
JefeEstacion
- log : Log
- cantEmpleados : int
- transferencias : TransferenciaEmpleado[*]
- colaAutos : Cola
+ run() : void
+ asignarEmpleado() : void
LectorComandos
+ lanzarConsultarCaja() : void
+ notificarFin() : void
+ run() : void
AccesoCaja
- log : Log
- colaCaja : Cola
- colaRta : Cola
- caja : int
+ run() : void
ConsultarCaja
- log : Log
+ run() : void
powered by Astah

7.2. Diagrama de transición de estados: Empleado



8. Casos de Uso

Caso de Uso: Consultar saldo de caja	
Descripción: Se obtiene del sistema el valor de la caja registradora al momento de la solicitud	
Actores participantes: Mánager	
Pre-condiciones: La estación debe haber recibido los parámetros de cantidad de empleados y surtidores	
Flujos	
Flujo Principal	
1	El mánager ingresa el comando “c” por entrada estándar.
2	El sistema imprime por salida estándar el monto recaudado en la caja al momento de la solicitud.
3	Si el programa se encuentra corriendo en modo debug (A1)
4	Finaliza el caso de uso
Flujos Alternativos	
A1	1. El sistema registra en el archivo de Log el monto recaudado en la caja al momento de la solicitud
	2. Continúa en (4)
Post-condiciones: -	

Caso de Uso: Iniciar simulación	
Descripción: Inicia la simulación de la estación con los parámetros determinados	
Actores participantes: Mánager	
Pre-condiciones: -	
Flujos	
Flujo Principal	
1	El sistema solicita la cantidad de empleados que tendrá la estación
2	El usuario ingresa la cantidad de empleados
3	El sistema solicita la cantidad de surtidores de los que dispondrá la estación
4	El usuario ingresa la cantidad de surtidores
5	El sistema pregunta si la simulación correrá en modo Debug
6	El usuario responde si la simulación correrá en modo Debug
7	El sistema inicia la simulación con los parámetros establecidos
8	Finaliza el caso de uso
Post-condiciones: La simulación se encuentra en ejecución	

Caso de Uso: Finalizar simulación	
Descripción: Finaliza la simulación de la estación liberando los recursos del sistema creados	
Actores participantes: Mánager	
Pre-condiciones: La simulación se encuentra en ejecución	
Flujos	
Flujo Principal	
1	El usuario ingresa el comando “q” por entrada estándar
2	El sistema libera los recursos utilizados
3	Si el programa se encuentra corriendo en modo debug (A1)
4	El sistema finaliza la simulación
5	Finaliza el caso de uso
Flujos alternativos	
A1	El sistema registra en el archivo de Log información sobre los procesos que finalizan
	Continúa en (4)
Post-condiciones: Los recursos del sistema creados para la simulación quedan liberados	