Programación Concurrente

Práctica especial: Simulacro del 2do parcial

Ejercicio 1. [**Monitores**] Se desea implementar un monitor para coordinar múltiples threads que participan de un proceso de encoding de video. En este escenario un thread actua como productor de cuadros de video crudos (sin compresión), mientras que otros threads consumen paquetes de cuadros para realizar el proceso de compresión (encoding). Dado que los cuadros de video crudo son pesados, se desea que en ningún momento se almacenen más de M cuadros en el monitor (i.e., bloqueando al productor de cuadros).

Resuelva los siguientes puntos:

a) Implemente un monitor `Encoder` con métodos `putRawFrame(frame)` (invocado por el thread productor de cuadros crudos) y `getPack()` (invocado por los threads encargados de la compresión). En este punto considere que el método `getPack` retorna una lista de P cuadros crudos (con P constante y menor o igual que M). Es decir, la ejecución de `getPack` debe bloquearse hasta que haya P cuadros disponibles.

b) Modifique el punto anterior de forma tal que los consumidores de cuadros puedan parametrizar la cantidad de cuadros con la que desean trabajar. Es decir, considere el método `getPack(int p)` que toma por parámetro la cantidad de cuadros a incluir en un paquete e ignore la constante P del punto anterior.

c) Extienda el punto anterior agregando un método adicional que almacena en el monitor un paquete de cuadros ya encodeados, `putEncodedPack(encodedPack)`. Este método tiene como precondición que sólo puede llamarse luego de una invocación a `getPack`, e indica que se concluyó la compresión del paquete de cuadros. En este punto se desea que se limite la cantidad de threads realizando tareas de encoding concurrentemente a K (con K constante). Es decir, que luego de K ejecuciones de `getPack` las siguientes se bloqueen hasta que se ejecute alguno de los `putEncodedPack` correspondientes.

Ejercicio 2. [**Mensajes**] Una agencia de viajes desea automatizar la consulta de paquetes turísticos. Dada la fecha de un viaje la agencia provee un paquete con tres componentes, un vuelo ida y vuelta al destino, un hotel donde hospedarse y el alquiler de un automóvil. La agencia trabaja con un proveedor distinto para cada servicio. Los proveedores ofrecen una interfaz de consulta y contratación a través de web-services (modelados como procesos que admiten comunicación mediante intercambio de mensajes). El servicio de vuelos acepta consultas en el canal `vuelo`, el servicio de hoteles utiliza el canal `hotel`, y el servicio de autos usa el canal `auto`. La agencia debe atender pedidos en el canal `agencia`. Todos los pedidos tienen la forma de una tupla con la fecha del viaje y el canal por donde se espera la respuesta. Las respuestas constan de un booleano que indica la disponibilidad del servicio en la fecha dada.

Resuelva los siguientes puntos utilizando intercambio de mensajes asincrónicos:

a) Implemente el proceso de la agencia de viajes que consulta simultáneamente a cada uno de los proveedores de servicios de forma de maximizar la concurrencia, e indica si un paquete puede armarse en la fecha dada por el cliente.

b) Modifique el punto anterior de forma tal que la agencia de viajes acepte múltiples consultas concurrentes.

c) Extienda el punto anterior considerando que en lugar de un único proveedor para cada servicio la agencia trabaja con múltiples proveedores. Para ello considere las variables `vuelo`, `hotel` y `auto` arreglos de canales. La agencia debe consultar a todos sus proveedores concurrentemente y reportar si puede armar un paquete completo. Tenga en cuenta que los proveedores pueden tener distintos tiempos de respuesta y que la agencia debe responder tan rápido como le sea posible.

Ejercicio 3. [**Monitores**] La observación del espacio es una actividad muy dependiente de la latitud y las condiciones atmosféricas del lugar donde se realiza. Por ello, no es raro que se construyan telescopios en ubicaciones óptimas para ser compartidos por diversos grupos de astrónomos. Se desea programar un monitor que gestione el acceso compartido a un telescopio. Para esto se deben proveer los métodos `iniciarObservacion()` y `finalizarObservacion()`, que permitan dar uso al instrumento. Adicionalmente, el telescopio admite la posibilidad de ser calibrado remotamente, por lo que el monitor también debe ser provisto de los métodos `iniciarCalibracion()` y `finalizarCalibracion()`. El telescopio no puede ser utilizado para observaciones mientras está siendo calibrado, ni puede calibrarse mientras haya otra calibración en curso o alguien realizando una observación.

Resuelva los siguientes puntos:

a) Implemente el monitor propuesto, tomando en cuenta que el telescopio no tiene un máximo de observadores concurrentes.

b) Modifique el ítem anterior para considerar que se puede controlar la dirección en la que está apuntando el telescopio (por simplicidad, diremos que las posiciones posibles son 1, 2, 3 y 4). Modifique el método `iniciarObservacion(int posicion)` para que tome por parámetro la posición que se quiere observar. Si el telescopio no está siendo calibrado o utilizado para observar en otra dirección, se inicia la observación y se interpreta que el telescopio en ese momento se mueve de manera instantánea a la posición nueva. Si por el contrario el telescopio está en uso, se debe esperar a que se libere.

c) Modifique el ítem anterior para considerar que el telescopio no debe moverse de posición si hay algún pedido de calibración pendiente (efectivamente priorizando la calibración por sobre las observaciones en posiciones distintas a la actual).

Ejercicio 4. [**Mensajes**] Un sistema de monitoreo de equipos IT funciona mediante la ejecución coordinada de múltiples servicios. En los equipos a monitorear se ejecuta un Agente (con un ID único) que una vez por minuto reporta que esta funcionando. Un Proxy por red local actúa de intermediario entre los agentes y un Servidor central, recibiendo los reportes de los Agentes y reenvíandolos al Servidor. El Servidor almacena un log con la actividad reportada (por simplicidad imprimiendo por pantalla los mensajes a medida que los va recibiendo).

a) Modele el escenario descrito utilizando intercambio de mensajes por canales (sin utilizar memoria compartida).

b) Modifique la solución anterior para que el Servidor central responda cada mensaje con un número aleatorio que debe ser sumado al ID del agente en futuros reportes (i.e., manteniendo el esquema de que el Servidor no se comunica directamente con los agentes, sino que pasa a través del Proxy).

c) Extienda la solución anterior para que el Servidor notifique (i.e., imprimiendo por pantalla) cuando no haya recibido comunicación de algún Agente cualquiera en los últimos 2 minutos (aproximadamente).