Monitores

Computación concurrente

Dante Bermúdez Marbán

16 de octubre del 2019

En este documento se hablará un poco acerca de un paper llamado *Monitors: An Operating System Structuring Concept* por C.A.R Hoare.

Se inicia explicando que el objetivo principal de un sistema operativo es gestionar los recursos de la instalación ante impredecibles demandas por los diferentes programas. Por lo tanto, se deben crear algoritmos que asignen y planifique tiempos de distintos tipos.

Cada planificador tendrá cierta información local así como procedimientos y funciones que son usados por los programas que desean adquirir y poner a disposición recursos.

Los monitores se definen como clases, pues es conveniente que en un sistema operativo existan varios monitores con una estructura similar (mismas funciones, etc).

El problema estriba en que, a pesar de que los monitores están disponibles para todos los programas en ejecución, solamene uno (a la vez) puede usarlo, haciendo que las llamadas a éste se mantengan en espera.

Los procedimientos locales a un monitor no deberían usar variables que no se ubiquen dentro del monitor.

Los planificadores necesitan una operación de espera (*wait*) para aquel programa que desea usar un recurso no está disponible y una operación de notificación (*signal*) que reanudará la ejecución del programa que estuviese esperando (si no hay programas esperando, pues no pasa algo).

También se nos habla de las variables tipo condición. La persona que diseña el monitor debe definir una variable de este tipo por cada motivo el programa debería esperarse. Estas variables no tienen como valor ni falso ni verdadero. Si más de un programa está esperando, la notificación reactivará

al programa que lleve más tiempo esperando (en implementaciones, se puede encontrar una cola de procesos que están esperando).

Posteriormente, se propone otra implementación de los monitores usando semáforos. A los monitores se les implmenta un segundo semáforo *urgent* (urgente) el cual está inicializado en cero y se suspende en cuanto se realiza la espera de urgente. Esto se debe a que, mientras se hace el proceso de notificación, se debe esperar un cambio de contexto para proceder.

Mas adelante, se propone que los monitores son candidatos para asociar invariantes resèctp a la información local. En este caso, las invariantes consiste en una proposición g, la cual es verdadera después de la inicialización y antes de la operación *wait*.

Para cada variable de condición b, se le asocia un predicado B que describe la condición en el que un programa que escribe en b desea reanudar su actividad.

Luego, el autor nos proporciona algunos ejemplos como el del buffer, con sus variables, operaciones de *append* y *remove* así como sus invariantes para probar que los monitores son correctos. Otro ejemplo es el de un planificador en el que se añade un parámetro para indicar prioridad (y así, la cola se convierte en una cola de prioridad). De igual manera, se incluye el problema de *readers and writers*.

En conclusión, los monitores son planificadores que gestionan los recursos para distintos procesos paralelos.