Linkia FP

Formación Profesional Oficial a Distancia

Linkia FP

DAM – M09 – Clase 04

Programación de servicios y procesos.

UF2 – Procesos y Hilos

Sincronización de Hilos.

M09: Programación de Servicios y Procesos.

99 horas en total.

UF2: Procesos y Hilos (37 horas)
UF3: Sockets y Servicios (37 horas)
UF1: Seguridad y Criptografía (25 horas)

4 Actividades – 3 Tests – 10 Clases Virtuales.

Semáforos

Los semáforos se pueden utilizar para controlar el acceso a un determinado recurso formado por un número finito de instancias. Se representa como una variable entera donde su valor representa el número de instancias libres o disponibles en el recurso compartido y una cola donde se almacenan los procesos o hilos bloqueados esperando para usar el recurso. En la inicialización se proporciona un valor inicial igual al número de recursos disponibles.

Operaciones:

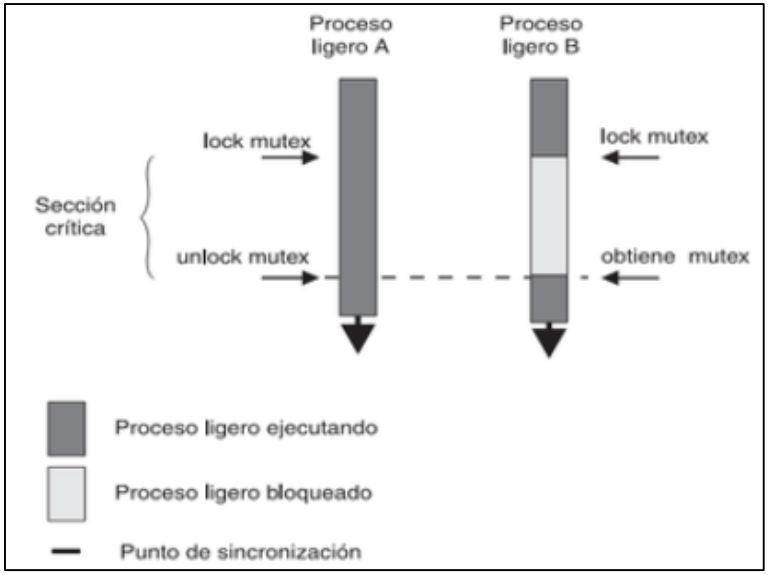
- **wait (espera):** Un proceso disminuye el número de instancias disponibles en uno, ya que se supone que lo va a utilizar.

- **Signal (señal):** Un proceso, cuando termina de usar la instancia del recurso compartido, avisa de su liberación mediante esta operación. Aumenta el valor de instancias disponibles en el semáforo. Si hay varios procesos esperando uno de ellos pasará a Runnable.

Semáforos binarios

- Un semáforo binario o mutex (MUTual EXclusion, Exclusión mutua) es un indicador de condición que registra si un único recurso está disponible o no.

- Un mutex solo puede tomar los valores 0 y 1. Si el semáforo vale 1, entonces el recurso esta disponible, y se puede acceder a la zona de compartición del recurso. Si el valor del semáforo es 0, el hilo debe esperar.

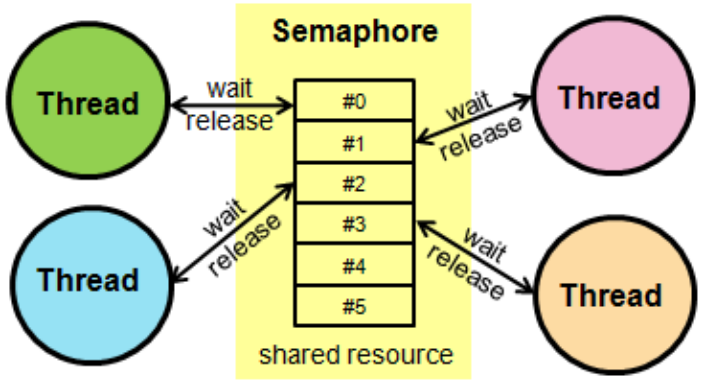


Semáforos

En java la utilización de los semáforos se realiza mediante el paquete java.util.concurrent y su clase Semaphore correspondiente.

Los métodos más importantes de esta clase son:

- **Semaphore(int valor)** Indica el valor inicial del semáforo antes de comenzar su ejecución.
- **acquire()** implementa la operación wait.
- **release()** implementa la operación signal. Desbloquea un hilo que esté esperando en el método wait().



Ejemplo Semaphore

En el siguiente ejemplo: Se ejecutan los hilos de dos en dos utilizando semáforos.

```
El proceso [ Proceso #0 ] dormira  5 segundos
El proceso [ Proceso #1 ] dormira  5 segundos
Finaliza el proceso [ Proceso #1 ]
Finaliza el proceso [ Proceso #0 ]
El proceso [ Proceso #3 ] dormira  5 segundos
El proceso [ Proceso #6 ] dormira  5 segundos
Finaliza el proceso [ Proceso #3 ]
El proceso [ Proceso #5 ] dormira  5 segundos
Finaliza el proceso [ Proceso #6 ]
El proceso [ Proceso #7 ] dormira  5 segundos
Finaliza el proceso [ Proceso #5 ]
El proceso [ Proceso #2 ] dormira  5 segundos
Finaliza el proceso [ Proceso #7 ]
El proceso [ Proceso #9 ] dormira  5 segundos
Finaliza el proceso [ Proceso #9 ]
Finaliza el proceso [ Proceso #2 ]
```

```java
import java.util.concurrent.Semaphore;

 class Semaforo implements Runnable {

    //  Número de  procesos que se pueden ejecutar al tiempo.
    private static final Semaphore DISPONIBILIDAD = new Semaphore(2);
    //nombre del proceso
    private final String nombre;

    public Semaforo(String nombre) {
        this.nombre = nombre;
    }

    public void run() {
        try {
            // Solicita disponibilidad.
            DISPONIBILIDAD.acquire();
             System.out.println("El proceso [ " + this.nombre + " ] dormira " +
                "5 segundos");
             Thread.sleep(5000);
             System.out.println("Finaliza el proceso [ " + this.nombre + " ]");
             // Libera disponibilidad.
            DISPONIBILIDAD.release();
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}
```

Linkia FP

Pregunta de la 4a clase virtual.

¿Qué hace el método acquire() de los semáforos?.

CLASE

Linkia FP

Formación Profesional Oficial a Distancia