Programación multihilo

Programación de servicios y procesos

gmq.psp2019@gmail.com

https://github.com/GMQPSP

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Recursos compartidos por hilos

- Recursos compartidos entre los hilos:
  - Código (instrucciones).
  - Variables globales.
  - Ficheros y dispositivos abiertos
  - Direcciones de memoria

Recursos compartidos por hilos

- Todos los hilos de un proceso comparten los recursos del proceso.
- Residen en el mismo espacio de direcciones y tienen acceso a los mismos datos.
- Cuando un hilo modifica un dato en la memoria, los otros hilos utilizan el resultado cuando acceden al dato
- Los valores comunes se guardan en el bloque de control de proceso (PCB), y los valores propios en el bloque de control de hilo (TCB).

Recursos compartidos por hilos

- Recursos **NO** compartidos entre los hilos:
  - Contador del programa (cada hilo puede ejecutar una sección distinta de código).
  - Registros de CPU.
  - Pila para las variables locales de los procedimientos a las que se invoca después de crear un hilo.
  - Estado: distintos hilos pueden estar en ejecución, listos o bloqueados esperando un evento.

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos.Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Estados de un hilo

- **Preparado (o nuevo)** : tras su creación espera a pasar a *en ejecución*
- **En ejecución**: realiza las tareas que tiene encomendadas
- **Bloqueado o dormido**: deja de ejecutarse por algún suceso

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Cambios de estado

- **Creación**:
  - Un proceso se crea su hilo.
  - Este hilo puede crear otros hilos dentro del mismo proceso proporcionando:
    - un puntero de instrucción y
    - argumentos del nuevo hilo.
  - El hilo tendrá su propio contexto y pasará al final de los hilos en estado *Preparado*.
  - Estado inicial: *preparado*

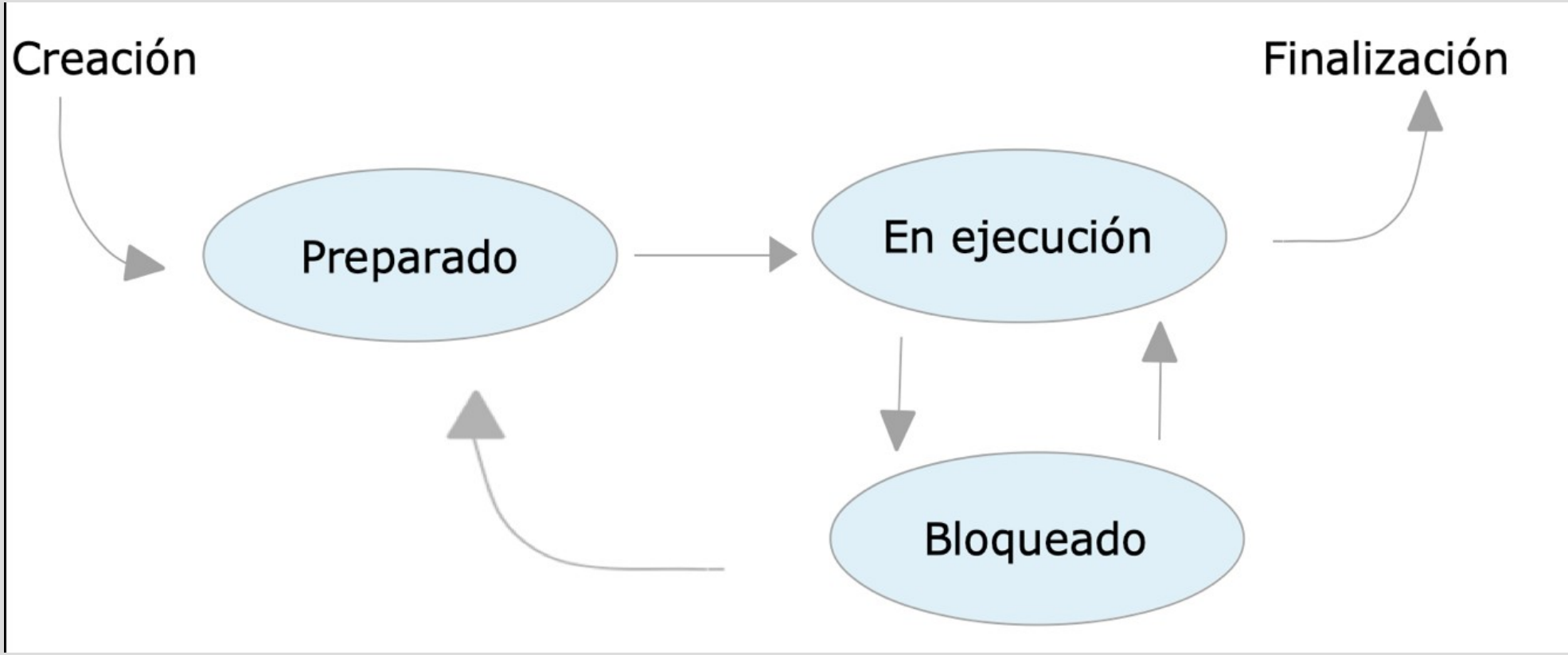
Cambios de estado

- **Bloqueo**: Cuando un hilo necesita esperar por un suceso, se bloquea (guardando sus registros de usuario, contador de programa y punteros de pila).
  - El procesador podrá pasar a ejecutar otro hilo que esté al principio de los hilos en estado *Preparado*.
  - De *en ejecución* a *bloqueado*

Cambios de estado

- **Desbloqueo**: cuando el suceso por el que el hilo se bloqueó se produce, el mismo pasa a la final de los hilos en estado *Preparado*.
  - De *bloqueado* a *preparado*
- **Terminación**: Cuando un hilo finaliza se liberan tanto su contexto como sus columnas.
  - Finaliza, normalmente, tras estar *en ejecución*

Estados de un hilo



Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Programación de hilos

- Las ventajas de los hilos se da cuando un procesos tiene múltiples hilos de ejecución los cuales realizan actividades distintas, que **pueden o no ser cooperativas** entre si.
- Los beneficios de los hilos se derivan de las implicaciones de rendimiento

Programación de hilos

- Se tarda mucho menos tiempo
  - en crear un hilo nuevo en un proceso existente que en crear un proceso (en orden de 10)
  - En terminar un hilo que un proceso
  - En cambiar de estado un hilo que un proceso
- Es más eficaz la comunicación entre hilos que entre procesos

Programación de hilos

- Cuándo usar multihilo
  - Trabajo interactivo y en segundo plano
  - Procesamiento asíncrono
    - Ej: Programación reactiva (RxJava)
  - Aceleración de la ejecución
  - Estructura modular de los programas

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

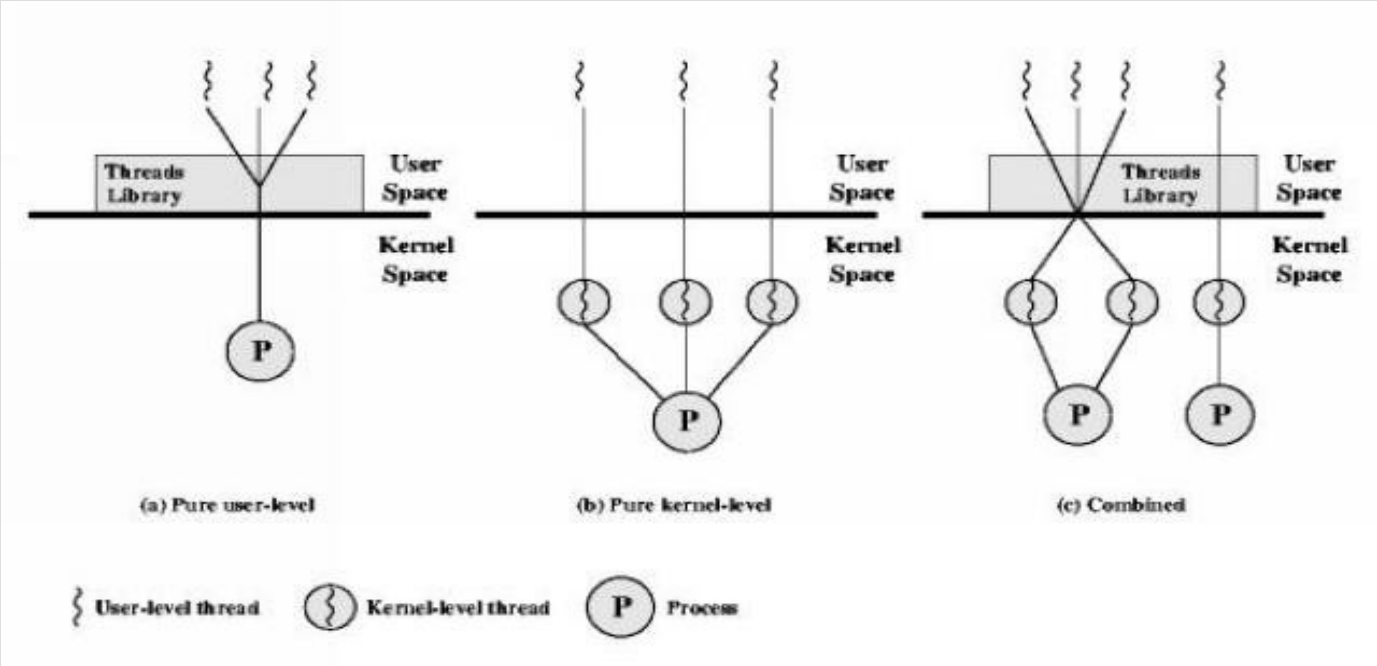
Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Gestión de hilos

- Existen dos categorías para la implementación de hilos: hilos a nivel de usuario (ULT) e hilos a nivel de núcleo/kernel (KLT)



Gestión de hilos

- ULT:
  - Son gestionados por las aplicaciones (el kernel no los conoce)
  - El cambio de ejecución entre hilos es más ligero al no necesitar acceso al kernel
  - La aplicación puede especificar una planificación de los hilos

Gestión de hilos

- KLT
  - Utilizado por la mayoría de sistemas operativos
  - Si se bloquea un hilo, puede planificar otro del mismo proceso.
  - Las propias funciones del kernel pueden ser multihilo.
  - El cambio entre hilos es costoso

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Creación, inicio y finalización

- Creación en Java
  - Interfaz Runnable (preferible)

```java
public class MyThread implements Runnable {

    public MyThread() {

    }

    @Override
    public void run() {
        //do stuff
    }
}
```

```java
public class ThreadsApplication {

    public static void main (String ... args) {
        Runnable thread = () -> { /*do stuff*/ };

        new Thread(thread).start();
    }
}
```

  - extends Thread:

```java
public class MyThread extends Thread {

    public MyThread() { }

    public void run() {
        //do stuff
    }
}
```

Creación, inicio y finalización

- Inicio de hilos en Java
  - Crear: Instancia clase
  - Init: Llamada a *run()*

```
public class MyThread implements Runnable {


    public MyThread() {

    }

    @Override
    public void run() {
        //do stuff
    }
}
```

  - Crear: Instancia *Thread*
  - Init: Llamar a *start()*

```
public class ThreadsApplication {

    public static void main (String ... args) {
        Runnable thread = () -> { /*do stuff*/ };

        new Thread(thread).start();
    }
}
```

Creación, inicio y finalización

- Finalización de hilos en java
  - No hay ninguna llamada que ordene finalizar un hilo
  - El hilo finaliza al cuando termina su ejecución
  - Hay que asegurarse de que los hilos tienen un fin

Creación, inicio y finalización

- Ejercicio
  - Crea una aplicación en Java que lance 5 hilos y que cada uno de ellos haga lo siguiente:
    - Escriba un mensaje: “Hola, soy “+id
    - Espere un tiempo aleatorio entre 1 y 2 segundos
    - Termine su ejecución con un mensaje:

      “Adiós, soy “+id

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Sincronización entre hilos

- Palabra reservada *synchronized*
- Utilizado en:
  - Métodos de clase
  - Métodos estáticos
  - Bloques de código
- Se debe sincronizar la menor parte de código posible: penaliza el rendimiento

Sincronización entre hilos

- Métodos synchronized

```
private synchronized void instanceMethod() {
    //do stuff
}

private static synchronized void staticMethod() {
    //do stuff
}
```

Sincronización entre hilos

- Java: synchonized block
  - synchronized(this): sincroniza el acceso al código para la instancia actual

```
private void mehod() {
    //do stuff
    synchronized (this) {
        //do more stuff
        //...
    }
}
```

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

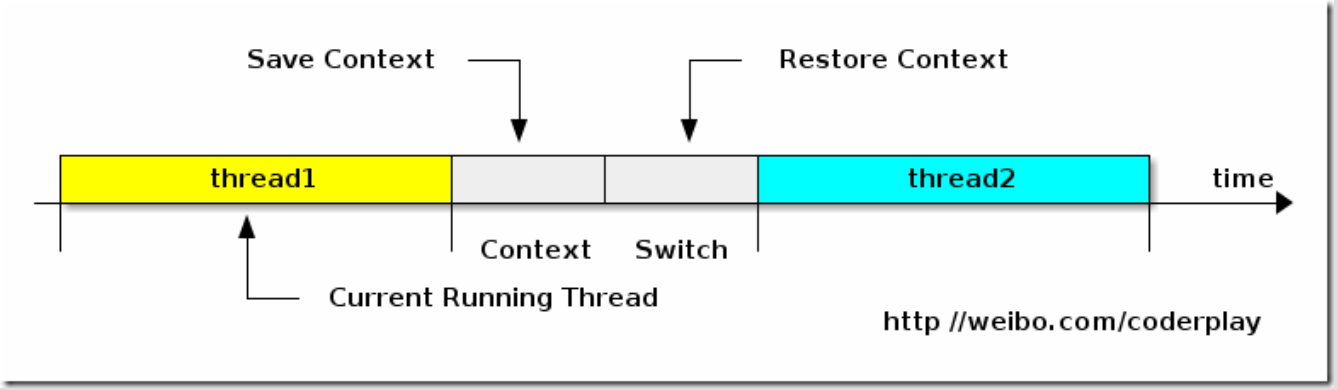
Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Prioridad de hilos

- Operación thread switching
  - en la ejecución de una rutina cuyo propósito es parar la ejecución de un hilo o proceso para dar paso a la ejecución de otro distinto.



Intercambio

- Pasos:
  - Salvar el estado del programa, el contexto, que se estaba ejecutando
  - Seleccionar otro programa para ejecutar
  - Restaurar el estado del programa seleccionado
  - Ejecutar el programa seleccionado.

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Prioridades de los hilos

- Cada hilo tiene una prioridad.
- Los de prioridad alta se ejecutan preferentemente a los de baja priodidad.
  - MAX < 10
  - MIN > 1
- API/Javadoc
  - setPriority(int)
    - IllegalArgumentException

Prioridades de los hilos

- Ejercicio:
  - Crear una aplicación que lance

    100 hilos con la misma prioridad

    - **Cada hilo se ejecutará 10 veces escribiendo “Hola soy +”id**
  - Crear una aplicación que lance

    100 con diferentes prioridades

    - **Cada hilo se ejecutará escribiendo “Hola soy +”id**

Programación multihilo

Recursos compartidos por los hilos.

Estados de un hilo.

Cambios de estado.

Elementos relacionados con la programación de hilos.

Gestión de hilos.

Creación, inicio y finalización.

Sincronización de hilos. Información entre hilos.

Intercambio.

Prioridades de los hilos. Gestión de prioridades.

Java8: Threads, Executors

Java 8: objectos Atomic y Concurrent

Java 8: Threads y Executors

- Interfaz Executor: ejecuta las tareas de un objeto *Runnable*.

```
Executor executor = anExecutor;
executor.execute(new RunnableTask1());
executor.execute(new RunnableTask2());
...
```

Java 8: Threads y Executors

- Algunas implementaciones de *Executor* imponen algún tipo de limitación en cómo y cuándo se ejecutan las tareas.
- Las implementaciones de *Executor* del paquete *java.util.concurrent* implementan la interfaz *ExecutorService*.
  - La clase *ThreadPoolExecutor* class provee una implementación de un pool de hilos.

Java 8: Threads y Executors

- *Executors* implementa métodos fábrica (factory methods) para *ExecutorService*

```
ExecutorService executor = Executors.newSingleThreadExecutor();
executor.submit(() -> {
    String threadName = Thread.currentThread().getName();
    System.out.println("Hello " + threadName);
});
```

Java 8: Threads y Executors

- Finalización de Executor
  - La interfaz *ExecutorService* define dos métodos para finalizar los ExecutorService:
    - shutdown(): esper a que las tareas actuales terminen
    - shutdownNow() interrumpe todas las tareas y tira el executor inmediatamente

Java 8: Threads y Executors

```
try {
    System.out.println("attempt to shutdown executor");
    executor.shutdown();
    executor.awaitTermination(5, TimeUnit.SECONDS);
}
catch (InterruptedException e) {
    System.err.println("tasks interrupted");
}
finally {
    if (!executor.isTerminated()) {
        System.err.println("cancel non-finished tasks");
    }
    executor.shutdownNow();
    System.out.println("shutdown finished");
}
```

Java 8: Threads y Executors

- ExecutorCompletionService
  - Se le provee de un *Executor* para correr las tareas
  - Retorna *Future:* resultado de una tarea asíncrona
  - La clase es lo suficientemente ligera para ser adecuada para un uso efímero cuando se procesan un grupo de tareas.

Java 8: Threads y Executors

- ExecutorCompletionService

```
final ExecutorService pool = Executors.newFixedThreadPool(10);
pool.submit(new Callable<String>() {
    @Override
    public String call() throws Exception {
        try (InputStream input = url.openStream()) {
            return IOUtils.toString(input, StandardCharsets.UTF_8);
        }
    }
});
```

Java 8: Threads y Executors

- *Callable*
  - Los *Executor a*además de implementaciones de *Runnable t*ambién soporta los Callable.
  - Callable: interfaz funcional similares a Runnable pero que retornan un valor.

Java 8: Threads y Executors

- *Timeouts*
- *InvokeAll*
- *InvokeAny*
- *Executor programados*

Java 8: extra

- *Locks and synchronized*
- *Paquete java.concurrent.atomic*

Java 8: práctica

- *Executors*
  - *Crea una aplicación basándose en Executors que dado un número de entrada cada hilo haga lo siguiente*
    - *Calcular su cuadrado (i^2)*
    - *Calcular su cubo (i^3)*
    - *Calcular su raíz cuadrada ($\sqrt{i}$)*
    - *Calcular su factorial (i!)*

Documentación

- Wikipedia
- Análisis arquitectural y funcional de la máquina virtual en la plataforma J2ME (Universidad. de Sevilla)
- Universidad de Valladolid
- Blog de Benjamin Winterberg (winterbe.com)