Innovación en Formación Profesional

Módulo Profesional 09:
Programación de servicios y procesos

Actividad UF2

CICLO FORMATIVO DE GRADO SUPERIOR EN

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

MODALIDAD ONLINE

Enrique Verea

Nombre de la actividad

Objetivos

Conocer los conceptos de procesos e hilos. Aprender multithread.

Competencias asociadas:

- Multithread

Metodología

- Preparación individual

Entrega

EN PDF el día 06/11/23

Dedicación estimada

600 minutos

Documentos de referencia

Recursos de la UF2.

Resultados de aprendizaje

- RA1. Desarrolla aplicaciones compuestas por varios procesos reconociendo y aplicando principios de programación paralela
-
- RA2.Desarrolla aplicaciones compuestas por varios hilos de ejecución analizando y aplicando librerías específicas del lenguaje de programación

Criterios de evaluación

Criterio 1. Identifica de las diferencias entre los paradigmas de programación paralela y distribuida.
Criterio 2. Identifica de los estados de un proceso.
Criterio 4. Sabe programar hilos y relación con los procesos.
Criterio 5. Aplica programación de aplicaciones multiproceso.
Criterio 6. Aplica sincronización y comunicación entre procesos.
Criterio 7. Gestión de procesos y desarrollo de aplicaciones con fines de computación paralela.
Criterio 8. Depura y documenta aplicaciones
Criterio 10. Gestiona hilos.
Criterio 11. Aplica programación de aplicaciones multihilo.
Criterio 15. Gestión de hilos por parte del sistema operativo. Planificación y acceso a su prioridad.

Desarrollo de la actividad

Ejercicio1. [0,65puntos]
Señala cuáles serán las posibles evoluciones de un proceso que se encuentra:

- En ejecución Bloqueado, listo (después de una interrupción) y terminado
- Bloqueado. Listo
- Acabado. (ninguno) el proceso ha terminado
- Nuevo. Listo
- Listo. En ejecución

Ejercicio2. [0,65puntos]
Explica con tus palabras qué es el concepto "*Cambio de contexto*".

Es el cambio de ejecución de un proceso a otro, en el cual el 'contexto' (memoria, estado del proceso, estado del procesador…) del proceso que se estaba ejecutando es guardado en registros del procesador para luego poder seguir siendo ejecutado, y un nuevo 'contexto' es creado para ejecutar el nuevo proceso.

Ejercicio3. [0,65puntos]
Cuál es el motivo de la utilización de la planificación de procesos.

Distribuir el tiempo de procesador de manera justa y eficiente entre todas las tareas de los procesos que se encuentran en ejecución.

Ejercicio4. [0,65puntos]
Busca al menos 4 algoritmos de planificación de procesos y explícalo brevemente, citando sus ventajas y desventajas.

First-come, First-serve (FCFS): ejecuta los procesos o los pone en lista de espera en orden de llegada. Cada proceso es ejecutado solo cuando el proceso anterior termina su ejecución.

- Ventaja: asegura que todos los procesos son ejecutados
- Desventaja: los tiempos de espera pueden ser muy largos, sobre todo si un proceso tiene que esperar por otro de tiempo de ejecución largo.

Shortest-Job-First (SJF): ejecuta los procesos de tiempo de ejecución más cortos primero.

- Ventaja: agiliza la cola de procesos, al salir de los más rápidos de primero
- Desventaja: puede darse el caso de que los procesos más largos siempre sean asignados al final de la cola de procesos y que nunca se lleguen a ejecutar

Shortest Remaining Time Next (SRTF): da prioridad a aquellos procesos que se encuentran más cerca de su finalización, incluso interrumpiendo el proceso en ejecución si es necesario.

- Ventaja: mejora el tiempo de ejecución con respecto a SJF
- Desventaja: un proceso con tiempo de ejecución largo puede ser interrumpido continuamente con la llegada de procesos más cortos, lo que disminuye su rendimiento considerablemente, e incluso puede impedir su finalización

Round Robin: trata a todos los procesos por igual, dando un mismo tiempo de procesador a cada uno. Si el proceso no termina su ejecución dentro de este tiempo, es movido al final de la cola de procesos, en la que esperará nuevamente su turno.

- Ventaja: todos los procesos reciben tiempo de procesador equitativo, lo que reduce el tiempo de espera. Es circular, lo que garantiza la ejecución de todos los procesos.
- Desventaja: las constantes interrupciones a los procesos al ser movidos al final de la cola puede tener un impacto negativo en el rendimiento

Ejercicio5. [0,65puntos]
Busca cual es la función de cada uno de los siguientes métodos de la clase *Thread*:

- getName() regresa el nombre de la instancia de Thread
- getPriority() regresa la prioridad asignada a la instancia de Thread
- isAlive() regresa true si el proceso está vivo, false si está muerto
- join() suspende la ejecución de la Thread desde donde se invoca este método join, hasta que muera la Thread a la que se le invoca el join.
- sleep() suspende la ejecución de la Thread durante un tiempo específico
- start() comienza la ejecución del código *Runnable* de este Thread

Ejercicio6. [0,65puntos]
Cita las ventajas de la programación multihilo, y explica cuando ésta es útil.

- Mejora el rendimiento de los procesos al permitir múltiples operaciones de computación y E/S de manera simultánea
- Permite la utilización de múltiples procesadores, que se traduce en un paralelismo real
- Mejora los tiempos de respuesta de las aplicaciones al permitir la ejecución de computaciones complejas y operaciones de E/S en hilos independientes al hilo encargado a atender las peticiones del usuario
- Mejora los tiempos de respuesta de servidores: al permitir la ejecución de diferentes solicitudes en diferentes hilos, evita que las solicitudes se bloqueen entre ellas.
- Minimiza el uso de recursos del sistema: los hilos utilizan muchos menos recursos que los procesos tradicionales. La creación de nuevos hilos no supone ninguna reserva adicional de memoria.
- Mejora la comunicación y la compartición de recursos: los hilos comparten la misma memoria y todos los recursos del proceso al que pertenecen, lo que facilita y su compartición y asegura una comunicación eficaz entre hilos.

Ejercicio7.[4puntos] +info 1 videoconferencia
Partiendo del ejemplo visto en la videoconferencia (Carreras de coches) amplía el ejercicio para realizar un programa que simule aleatoriamente una carrera entre 3 caballos.
El programa solicitará los nombres de los caballos.
El programa generará de manera aleatoria el tiempo de cada uno de los tres caballos (*Random*).
El Programa dará salida a los 3 caballos a la vez. Y mostrará el tiempo que ha tardado cada caballo en terminar.
A continuación, se muestra un ejemplo:

```
Dame el nombre del caballo 1:
DAM
Dame el nombre del caballo 2:
DAW
Dame el nombre del caballo 3:
ASIR
Sale DAM
Sale DAW
Sale ASIR
Soy el caballo  DAW he tardado 2 segundos
Soy el caballo  ASIR he tardado 3 segundos
Soy el caballo  DAM he tardado 5 segundos
```

```
public class Carrera {

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Nombre del caballo 1: ");
      String nombre1 = scanner.nextLine();

      System.out.print("Nombre del caballo 2: ");
      String nombre2 = scanner.nextLine();

      System.out.print("Nombre del caballo 3: ");
      String nombre3 = scanner.nextLine();

      Caballo caballo1 = new Caballo(nombre1);
      Caballo caballo2 = new Caballo(nombre2);
      Caballo caballo3 = new Caballo(nombre3);

      new Thread(caballo1).start();
      new Thread(caballo2).start();
      new Thread(caballo3).start();
   }
}
```

```java
public class Caballo implements Runnable {

    private final String nombre;
    private final int segundos;

    public Caballo(String nombre) {
        this.nombre = nombre;
        this.segundos = ThreadLocalRandom.current().nextInt(11); // aleatorio entre 0 y 10
    }

    @Override
    public void run() {
        try {
            Thread.sleep(segundos * 1000L);  // suspende este hilo
            System.out.println("Soy el caballo " + nombre +". He tardado " + segundos + " segundos");
        }
        catch (InterruptedException e) {
            System.err.println("El programa ha sido interrumpido");
        }
    }
}
```

```
Nombre del caballo 1: Rocinante
Nombre del caballo 2: Pegaso
Nombre del caballo 3: Seabiscuit
Soy el caballo Seabiscuit. He tardado 0 segundos
Soy el caballo Pegaso. He tardado 1 segundos
Soy el caballo Rocinante. He tardado 7 segundos

Process finished with exit code 0
```

Ejercicio8. [1punto] **+info 2 videoconferencia**

Amplía el ejercicio anterior para mostrar el caballo ganador, dentro del hilo principal.

```
run:
Dame el nombre del caballo 1:
DAM
Dame el nombre del caballo 2:
DAW
Dame el nombre del caballo 3:
ASIR
Sale DAM
Sale DAW
Sale ASIR
Soy el caballo  DAW he tardado 2 segundos
Soy el caballo  ASIR he tardado 3 segundos
Soy el caballo  DAM he tardado 5 segundos
El ganador ha sido DAW
BUILD SUCCESSFUL (total time: 19 seconds)
```

```
public class Carrera {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Nombre del caballo 1: ");
        String nombre1 = scanner.nextLine();

        System.out.print("Nombre del caballo 2: ");
        String nombre2 = scanner.nextLine();

        System.out.print("Nombre del caballo 3: ");
        String nombre3 = scanner.nextLine();

        Caballo caballo1 = new Caballo(nombre1);
        Caballo caballo2 = new Caballo(nombre2);
        Caballo caballo3 = new Caballo(nombre3);

        Thread thread1 = new Thread(caballo1);
        Thread thread2 = new Thread(caballo2);
        Thread thread3 = new Thread(caballo3);

        // ejecuta los hilos
        thread1.start();
        thread2.start();
        thread3.start();

        try {
            // suspende este hilo hasta que los hilos correspondientes a cada caballo
            // terminan su ejecución y mueren
            thread1.join();
            thread2.join();
            thread3.join();

            String ganador =
                caballo1.masRapidoQue(caballo2) && caballo1.masRapidoQue(caballo3)
                    ? caballo1.getNombre()
                    : caballo2.masRapidoQue(caballo3)
                    ? caballo2.getNombre()
                    : caballo3.getNombre();

            System.out.println("El ganador es: " + ganador);
        }

        catch (InterruptedException e) {
            System.err.println("La carrera ha sido suspendida." +
                    "No se ha podido determinar ningún ganador");
        }
    }
}
```

```java
public class Caballo implements Runnable {

    private final String nombre;
    private final int segundos;

    public Caballo(String nombre) {
        this.nombre = nombre;
        this.segundos = ThreadLocalRandom.current().nextInt(1, 11); // aleatorio entre 1 y 10
    }

    public String getNombre() {
        return nombre;
    }

    public boolean masRapidoQue(Caballo caballo) {
        return this.segundos <= caballo.segundos;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(segundos * 1000L);  // suspende este hilo
            System.out.println("Soy el caballo " + nombre +". He tardado " + segundos + " segundos");
        }
        catch (InterruptedException e) {
            System.err.println("El programa ha sido interrumpido");
        }
    }
}
```

```
Nombre del caballo 1: Rocinante
Nombre del caballo 2: Pegaso
Nombre del caballo 3: Seabiscuit
Soy el caballo Rocinante. He tardado 2 segundos
Soy el caballo Seabiscuit. He tardado 5 segundos
Soy el caballo Pegaso. He tardado 7 segundos
El ganador es: Rocinante

Process finished with exit code 0
```

Ejercicio9. [1punto]
Amplía y añádele alguna mejora al simulador de carreras de caballos.

Clases:

Main: https://pastebin.com/HW8CiY27

Caballo: https://pastebin.com/1v3U216v

Caballos: https://pastebin.com/vGNhAztp

AvatarCaballo: https://pastebin.com/MSAtVDQc

Carrera: https://pastebin.com/i4BQxrC8

Trofeo: https://pastebin.com/uAJ2Hh3G

Captura:

https://drive.google.com/file/d/1Xrg4iS_iRyOSdB64tsG0SEgY2zjczhAi/view?usp=drive_link