Programación y sincronización de hilos

Sistemas Operativos

Descripción

El objetivo de la práctica es que el estudiante se familiarice con la sincronización de procesos y *threads*. Para ello, se va a plantear un problema en el que se deberá implementar el sistema de acceso a un parking usando *threads*.

El programa a implementar debe gestionar las entradas y salidas de vehículos al parking. Los posibles vehículos que pueden entrar a dicho parking son automóviles y camiones. Un automóvil ocupará una plaza de aparcamiento dentro del parking, mientras que un camión ocupará **dos plazas contiguas** de aparcamiento. Las plazas del parking se sitúan de manera contigua de la siguiente forma:

0	1	2	3	…	N

El parking dispondrá de una única entrada y una única salida, por las que entrarán y saldrán (respectivamente) los distintos tipos de vehículos permitidos. En la entrada de vehículos habrá un dispositivo de control que permita o impida el acceso de los mismos al parking, dependiendo del estado actual del mismo (plazas de aparcamiento disponibles). En caso de que el dispositivo de control permita el acceso a un vehículo, se le asignará el número de plaza de aparcamiento donde debe aparcar. Queda a decisión del control de acceso qué plaza asigna de entre todas las que haya libres.

Los tiempos de espera de los vehículos dentro del parking son aleatorios (para ello, se puede utilizar la función `rand()`). En el momento en el que un vehículo sale del parking, notifica al dispositivo de control el número de plaza que tenía asignada y se libera la plaza o plazas de aparcamiento que estuviera ocupando, quedando así éstas nuevamente disponibles.

Un vehículo que ha salido del parking esperará un tiempo aleatorio para volver a entrar nuevamente en el mismo. Por tanto, los vehículos estarán entrando y saliendo indefinidamente del parking. **Es importante que se diseñe el programa de tal forma que se asegure que, antes o después, un vehículo que permanece esperando a la entrada del parking entrará en el mismo.**

La salida del programa debería ser algo del tipo:

```
ENTRADA: Coche 1 aparca en 0. Plazas libre: 5
ENTRADA: Coche 2 aparca en 1. Plazas libre: 4
ENTRADA: Coche 3 aparca en 2. Plazas libre: 3
ENTRADA: Coche 4 aparca en 3. Plazas libre: 2
ENTRADA: Coche 5 aparca en 4. Plazas libre: 1
SALIDA: Coche 1 saliendo. Plazas libre: 2
ENTRADA: Coche 6 aparca en 0. Plazas libre: 1
SALIDA: Coche 5 saliendo. Plazas libre: 2
SALIDA: Coche 2 saliendo. Plazas libre: 3
SALIDA: Coche 4 saliendo. Plazas libre: 4
ENTRADA: Coche 8 aparca en 1. Plazas libre: 3
SALIDA: Coche 3 saliendo. Plazas libre: 4
ENTRADA: Coche 7 aparca en 2. Plazas libre: 3
ENTRADA: Coche 10 aparca en 3. Plazas libre: 2
ENTRADA: Coche 9 aparca en 4. Plazas libre: 1
SALIDA: Coche 8 saliendo. Plazas libre: 2
SALIDA: Coche 6 saliendo. Plazas libre: 2
SALIDA: Coche 9 saliendo. Plazas libre: 3
SALIDA: Coche 10 saliendo. Plazas libre: 5
ENTRADA: Camión 101 aparcado en 2. Plazas libre: 3
ENTRADA: Coche 2 aparca en 1. Plazas libre: 1
ENTRADA: Camión 102 aparcado en 4. Plazas libre: 1
SALIDA: Coche 7 saliendo. Plazas libre: 4
SALIDA: Camión 101 saliendo. Plazas libre: 3
SALIDA: Coche 2 saliendo. Plazas libre: 2
ENTRADA: Camión 103 aparcado en 2. Plazas libre: 1
SALIDA: Camión 102 saliendo. Plazas libre: 4
ENTRADA: Coche 1 aparca en 0. Plazas libre: 3
ENTRADA: Coche 9 aparca en 1. Plazas libre: 2
ENTRADA: Coche 4 aparca en 4. Plazas libre: 1
SALIDA: Coche 1 saliendo. Plazas libre: 2
ENTRADA: Coche 5 aparca en 0. Plazas libre: 1
SALIDA: Camión 103 saliendo. Plazas libre: 4
ENTRADA: Coche 3 aparca en 1. Plazas libre: 3
ENTRADA: Coche 10 aparca en 2. Plazas libre: 2
SALIDA: Coche 9 saliendo. Plazas libre: 4
^C
```

Objetivos parciales

- Gestionar correctamente las entradas y salidas del parking únicamente para automóviles. Un automóvil podrá acceder al parking si hay, al menos, una plaza de aparcamiento libre. En el caso en el que el parking esté lleno, el automóvil deberá esperar en la barrera de entrada a que se libere una plaza de aparcamiento. Para dar por correcto el funcionamiento de este apartado, el programa tiene que estar libre de interbloqueos y no producir inanición (**2 puntos**).

- Gestionar correctamente las entradas y salidas del parking para todo tipo de vehículos permitidos (tanto automóviles como camiones). Un camión sólo podrá acceder al parking si hay, al menos, dos plazas contiguas de aparcamiento libre. En el caso de que no haya plazas disponibles, el vehículo deberá esperar en la barrera de entrada hasta que el control de acceso le permita entrar. Para dar por correcto el funcionamiento de este apartado, el programa tiene que estar libre de interbloqueos y no producir inanición (**5 puntos**).

- Mostrar por pantalla el estado del parking después de que entre un vehículo (**1 punto**):

```
ENTRADA: Coche 1 aparca en 0. Plazas libre: 5
Parking: [1] [0] [0] [0] [0] [0]
ENTRADA: Coche 2 aparca en 1. Plazas libre: 4
Parking: [1] [2] [0] [0] [0] [0]
ENTRADA: Coche 3 aparca en 2. Plazas libre: 3
Parking: [1] [2] [3] [0] [0] [0]
ENTRADA: Coche 4 aparca en 3. Plazas libre: 2
Parking: [1] [2] [3] [4] [0] [0]
ENTRADA: Coche 5 aparca en 4. Plazas libre: 1
```

```
Parking: [1] [2] [3] [4] [5] [0]
SALIDA: Coche 2 saliendo. Plazas libre: 2
SALIDA: Coche 3 saliendo. Plazas libre: 3
ENTRADA: Coche 7 aparca en 2. Plazas libre: 2
Parking: [1] [0] [7] [4] [5] [0]
ENTRADA: Coche 6 aparca en 1. Plazas libre: 1
Parking: [1] [6] [7] [4] [5] [0]
SALIDA: Coche 1 saliendo. Plazas libre: 2
SALIDA: Coche 4 saliendo. Plazas libre: 3
ENTRADA: Coche 9 aparca en 0. Plazas libre: 2
Parking: [9] [6] [7] [0] [5] [0]
ENTRADA: Coche 10 aparca en 3. Plazas libre: 1
Parking: [9] [6] [7] [10] [5] [0]
SALIDA: Coche 5 saliendo. Plazas libre: 2
ENTRADA: Camion 103 aparcado en 4 y 5. Plazas libre: 0
Parking: [9] [6] [7] [10] [103] [103]
SALIDA: Coche 7 saliendo. Plazas libre: 1
ENTRADA: Coche 2 aparca en 2. Plazas libre: 1
…
```

- Implementar un algoritmo para gestionar un parking con varias plantas, todas ellas con el mismo número de plazas sabiendo que un camión no puede aparcar entre dos plantas (**1 punto**). Este apartado implicar reescribir el método de escritura del parking para mostrar el estado del parking para el número de plantas especificado.

- Ser capaz de pasar como argumentos el número de plazas por cada planta (PLAZAS), el número de plantas del parking (PLANTAS), el número de automóviles (COCHES) y de camiones (CAMIONES) que van a acceder al mismo. Es decir, el programa tendría que entender (**1 puntos**):

  ```
  parking PLAZAS PLANTAS
  parking PLAZAS PLANTAS COCHES
  parking PLAZAS PLANTAS COCHES CAMIONES
  ```

  Si no se ha realizado el apartado anterior deberá escribirse 1 como argumento de número de PLANTAS. Si no se especifica el número de coches (en el primer caso), éste será el doble del número de plazas por cada planta del parking por el número de plantas, es decir: COCHES = 2*PLAZAS*PLANTAS. Si no se especifica el número de camiones, éste será 0.

Nota: las puntuaciones para cada objetivo parcial son las puntuaciones máximas que se pueden obtener si se cumplen esos objetivos.

Nota: no se debe hacer un programa separado para cada objetivo, sino un único programa genérico que cumpla con todos los objetivos simultáneamente.

Nota: Para evitar que se produzcan interbloqueos o inanición, no se considerarán válidas soluciones a medida implementadas por el alumno, que solucionen casos concretos.

Si se produce un interbloqueo o inanición en algún de los hilos, la práctica se considera suspensa con un 0.

Entrega de prácticas

La entrega de prácticas se hará a través del Aula Virtual en las fechas anunciadas en el mismo. Se debe entregar todo el proyecto, incluyendo las librerías auxiliares que se dan al estudiante. En el proyecto debe venir un ejecutable del mismo, que funcione en un Ubuntu 20, o superior. El fichero que contenga a la función *main* debe llamarse *myparking.*c y el ejecutable *myparking*

Evaluación de la práctica

Se realizará mediante una defensa presencial de la práctica en las fechas indicas en el Aula Virtual. La práctica se evaluará comprobando el correcto funcionamiento de los distintos objetivos, y valorando la simplicidad del código, los comentarios, la óptima gestión de recursos, la gestión de errores y la defensa de la misma.

Autoría de la práctica

La práctica se debe realizar en grupos de 2 personas como máximo.

En caso de detectar copia, los estudiantes afectados serán suspendidos con un 0 en la práctica en la convocatoria pertinente. Una práctica será considerada copia en caso de que contenga la totalidad o una parte de la práctica de otr@ estudiante. Se considerará copia en caso de:

- Archivos que contengan la totalidad o fragmentos de código de otr@ estudiante
- Archivos que contengan la totalidad o fragmentos de código de internet