UNIVERSIDAD DE MÁLAGA | uma.es

Software para sistemas empotrados y dispositivos móviles

Septiembre 2019

Grado Ing. Software

2. Simular con señales POSIX el mecanismo de entrada/salida y control de aforo de un aparcamiento. Existen dos tipos de automóviles, los turismos y las furgonetas. Supongamos que tenemos una **hebra** que simula los turismos y una **hebra** que simula las furgonetas que mediante un temporizador indica la llegada al control de entrada del parking. Los turismos llegan con un periodo de 2 seg. y las furgonetas con un periodo de 3 seg. El control de entrada del parking **(hebra)** recibe el aviso de llegada, mantiene el número de vehículos que están dentro y cada 2 seg. envía una señal al control de salida **(hebra)** para simular de forma aleatoria la salida del automóvil (turismo o furgoneta). Suponemos que hay aforo suficiente para albergar todos los automóviles.

Diseñar la solución **OBLIGATORIAMENTE** con los mecanismos de temporizadores y hebras de POSIX. La gestión de las tareas periódicas de llegada de coches se realizará mediante temporizadores (create_timer, etc).

2. Se desea implementar el control de un sistema de alarma de una casa. El sistema está formado por detectores de vibración y detectores de presencia que se simulará mediante una hebra por tipo de detector. Supondremos que es un sistema muy simple donde cada una de las dos hebras controla un único sensor. Cada tarea realiza de forma periódica (según el periodo y plazo indicado en la tabla) el siguiente procedimiento:

- Calcula el valor de la vibración de las ventanas de la habitación. Para ello, los valores se pueden generar aleatoriamente, inicializando el generador de números aleatorios al principio del programa mediante *srand(time(NULL)),* y generando los números aleatorios dentro del rango [MIN..MAX] como *MIN+(rand()%(MAX-MIN+1))*. Nota: esta generación de números aleatorios debe protegerse mediante exclusión mutua.

- Calcula la detección de la presencia en la habitación. Para ello, los valores se pueden generar aleatoriamente, inicializando el generador de números aleatorios al principio del programa mediante *srand(time(NULL)),* y generando los números aleatorios dentro del rango [MIN..MAX] como *MIN+(rand()%(MAX-MIN+1))*. Nota: esta generación de números aleatorios debe protegerse mediante exclusión mutua.

- Si la vibración es mayor que la vibración máxima MAX, mostrará un mensaje de “Posible entrada por ventana”
- Si se detecta presencia se mostrará un mensaje “Posible presencia”
- En cualquier otro caso se mostrará un mensaje “Todo OK”.

La tarea que controla la presencia se considera más prioritaria que la de vibración. Además, existe una tarea de monitorización, con mínima prioridad *MonitorAlarma,* que informa periódicamente del estado de la casa con un periodo de 4 s.

OBLIGATORIO PARA IMPLEMENTAR EL PROBLEMA

- **Asignar prioridades en función de los requisitos temporales y recordar que se debe limitar la inversión de prioridades.**
- **El comportamiento periódico de las hebras deberá diseñarse con *clock_gettime* y *clock_nanosleep*.**

Los requisitos temporales de las tareas son:

Tareas	**Periodo**	**Plazo**
Presencia	2000 ms	1500 ms
Vibración	3000 ms	2500 ms
Monitor	4000 ms	4000 ms

Planificabilidad (1 punto)

Consideremos el siguiente conjunto de tareas:

Tarea	**Periodo**	**Plazo**	**Tiempo de ejecución en el caso peor**	**Recursos utilizados**
A	90	70	2	R1,R3
B	100	100	9	R1,R2,R5
C	30	30	4	R4,R5
D	1000	600	11	R1,R3
E	150	95	12	R1,R2,R3,R4

Los tiempos de ejecución en el caso peor de los recursos indicados en la tabla son los siguientes:

R1	0.3
R2	0.2
R3	0.2
R4	0.1
R5	0.1

1. Calcular los factores de bloqueo suponiendo que el planificador usa **techo de prioridad** y calcular el tiempo de respuesta de las tareas del sistema**. La asignación de las prioridades debe ser óptima. (0,75)**
2. Supongamos un planificador basado en tick con un Tick de 1 y un C_{tick} de 0,2 y un tiempo de tratamiento de las colas C_{cola}=0,1. Incluiremos el tiempo de cambio de contexto de 0,1. Calcular el tiempo de respuesta de la tarea **C teniendo en cuenta todas estas sobrecargas. (0,25)**