Universidad
Rey Juan Carlos

Práctica 1: Multithreading en C

Sistema empotrados y de tiempo real

2022/2023

En esta práctica se trabajarán conceptos de multithreading en C utilizando pthreads.

Objetivos:

- Repaso de C y pthreads
- Ver la diferencia de ejecutar: Un proceso multihilo en un sistema monocore o multicore
- Comprobar prácticamente que Linux no es un sistema de tiempo real
- No podemos decidir cuando las tareas empiezan o terminan.
- Demostrar que no se pueden asegurar los plazos de ejecución.

Desarrolla un programa (practica1.c) que realice la siguiente funcionalidad.

- El programa debe crear 4 threads que se ejecuten a la vez y no secuencialmente.
- Cada thread es una tarea periódica con C=**0.5** segundos y P=**0.9** segundos (asumimos que el deadline y periodo son iguales).
- Los propios threads son los encargados de estimar el tiempo utilizado en su ejecución y controlar la frecuencia a la que se ejecutan.
- Cada thread debe ejecutar 5 iteraciones, teniendo en cuenta que son iteraciones periódicas (P=0.9).
- En cada iteración del thread debes asegurarte que está ejecutando un código random durante 0.5 segundos. Puedes utilizar el siguiente código, pero asegurate que su ejecución es aproximadamente de 0.5 segundos.

```
volatile unsigned long long j;
for (j=0; j < 400000000ULL; j++);
```

- El programa debe mostrar una línea por cada iteración y thread ejecutado con la siguiente información por pantalla (thread e iteraciones empiezan en 1 siempre).

```
[1663147910.736590668] Thread 1 - Iteracion 1: Coste=0.50 s.
```

- Si la restricción temporal no se cumplido debe mostrar lo siguiente:

```
[1663147910.736590668] Thread 1 - Iteracion 1: Coste=1.5 s. (fallo temporal)
```

- La ejecución del programa debe realizarse en 4 escenarios distintos
    - Ejecución multicore
    - Ejecución monocore
    - Ejecución multicore + stress (todas las cpus)
    - Ejecución monocore + stress (todas las cpus)

- Para ejecutar procesos utilizando una configuración multicore o monocore puedes utilizar el comando taskset
    - Multicore:
        - **./practica1**
    - Monocore:
        - **taskset -c 0 ./practica1**

- Comandos útiles para la práctica
    - htop: visualizador de procesos en linux
    - stress: genera stress computacional en las cpus
    - time: calcula el tiempo de ejecución de un proceso

PREGUNTAS:

1. Ejecutar los siguientes casos y justificar su comportamiento:
    a. ./practica1 (multicore)
    b. ./practica1 (monocore)
    c. ./practica1 (monocore) + *stress*
    d. ./practica1 (multicore) + *stress*

2. ¿En qué casos de ejecución (nombrados anteriormente) el sistema es capaz de cumplir las restricciones temporales (tanto tiempo de cómputo como periodicidad)?
3. ¿Qué número mínimo de cpus se necesitan para que tu programa ejecute correctamente sin fallos de restricciones temporales? Usa el comando taskset para comprobarlo.
4. ¿Qué solución se podría proponer para cumplir plazos estrictos temporales de periodicidad en la ejecución de los *threads* utilizando la configuración actual que tienen los ordenadores del laboratorio?

Entregables:

- Entrega definida en Aula Virtual.