SISTEMAS EMPOTRADOS Y DE TIEMPO REAL

Práctica 1 – Gonzalo Vega Pérez

1. Ejecutar los siguientes casos y justificar su comportamiento:

a. ./practica1 (multicore)
Todo funciona correctamente. El planificador asigna cada hilo a una cpu diferente. Los procesos no compiten por usar la cpu y todos pueden ejecutar sin problemas.

b. ./practica1 (monocore)
Todos los hilos quieren utilizar la misma cpu al mismo tiempo. El planificador debe decidir cual de todos puede ejecutar e ir expulsndolos y asignándolos. Todos terminan su iteración casi al mismo tiempo con un coste muy similar. Pero no cumplen los requisitos temportales / de coste.

c. ./practica1 (monocore) + stress
Igual que el caso sin stress pero aun más lento debido a la carga extra en la cpu.

d. ./practica1 (multicore) + stress
Aunque los hilos no compitan entre sí, debido a la saturación en las cpus algunas iteraciones no cumplen los requisitos.

2. ¿En qué casos de ejecución (nombrados anteriormente) el sistema es capaz de cumplir las restricciones temporales (tanto tiempo de cómputo como periodicidad)?

Multicore (no stress)

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.

1 y 2 CPUs – Fallo temporal y de tiempo de cómputo.
3 CPUs – Fallo de tiempo de cómputo pero no fallo temporal.
4 CPUs – Ejecuta correctamente.

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?

Añadiendo cerrojos. Así la tarea que tiene el cerrojo tendría prioridad para terminar su ejecución y no sería expulsada para que otros hilos pudiesen ejecutar. De esta manera un hilo no ejecutaría hasta que el anterior hubiese terminado y soltase el cerrojo.