Facultad de Informática | UNIVERSIDAD NACIONAL DE LA PLATA

Ingeniería en Computación

Sistemas de Tiempo Real

Práctica Nº 5

Programación en Sistemas Operativos de Tiempo Real Duro.

La siguiente práctica debe realizarse utilizando el Sistema Operativo GNU\Linux con el parche RT-Preempt, el lenguaje de programación C (compilador gcc) y las librerías de pthreads. Recomendamos utilizar la versión “Live CD” otorgada por la cátedra.

Ejercicio 1:

Escriba un programa que contenga dos threads que se ejecutan periódicamente en 1000 oportunidades. Ambos threads se ejecutan 100 veces por segundo y cada vez que se ejecutan acumulan el tiempo de latencia que pasó desde que se cumplió una centésima de segundo, hasta que verdaderamente retomó la ejecución la tarea. En otras palabras, el tiempo que tardó en retomar el thread la tarea menos la centésima de segundo que debía transcurrir. Una vez que ambos threads hayan finalizado, imprima el promedio de la latencia acumulada para cada uno de los dos threads.

Ejercicio 2:

Escribir dos programas. Por un lado, un simulador de un sensor de temperatura, y por el otro, un programa que va a monitorizar las temperaturas recibidas.

Para realizar el sensor de temperatura, realice un programa que lea de un archivo el tiempo (nanosegundos) que debe esperar para entregar un dato y la temperatura (**°C**) que debe entregar. Dicho archivo debe tener por cada fila dos números (el tiempo y la temperatura) separados por un <*tab*>. Una vez cumplido el plazo leído del archivo el programa debe escribir en un *pipe* la temperatura y esperar el cumplimiento del siguiente plazo para volver a escribir. Debe continuar con esta mecánica hasta la finalización del archivo.

Para realizar el monitor de temperaturas, escriba un programa que reciba los datos del simulador. Dicho programa debe contener dos threads que están a la espera de recibir datos de temperatura. Dichos threads están bloqueados hasta la llegada de un dato que será extraído del *pipe* generado por el sensor de temperatura. Cada uno de los threads tiene una prioridad diferente. El thread de mayor prioridad es el encargado de leer la temperatura del *pipe* y debe verificar que la temperatura no exceda los 90 °C. Si la temperatura excede ese límite debe imprimir un aviso de la situación. Por el contrario, si la temperatura no excede el límite debe pasar el dato al thread de menor prioridad (sin que este paso implique un bloqueo del

thread de prioridad alta). El thread de menor prioridad debe informar un promedio de los últimos 3 valores de temperatura que obtuvo.