28/04/2025

Práctico de laboratorio

Tema: Interrupciones

Consigna:

Desarrollar un sistema de monitoreo de temperatura en tiempo real en POSIX en C, utilizando interrupciones, hilos y mecanismos de sincronización, asegurando tiempos de respuesta adecuados. El sistema deberá simular la lectura de temperatura y responder de forma eficiente y confiable a cambios críticos.

Descripción del sistema:

El sistema deberá simular un sensor de temperatura utilizando un generador de números aleatorios. Cada lectura generará un valor de temperatura que será evaluado periódicamente.
Al detectar que la temperatura excede un umbral definido, se deberá generar una interrupción que active una alarma.

Requisitos técnicos:

- Simulación de lectura de temperatura: Implementar una función que simule la lectura de temperatura generando valores aleatorios en un rango realista (por ejemplo, 15°C a 40°C). Esta función deberá ser ejecutada periódicamente por un hilo separado.
- Detección de interrupciones: Utilizar señales POSIX para manejar "interrupciones" que se activan cuando la temperatura excede un umbral crítico. Configurar un manejador de señales que procese las interrupciones y active una alarma.
- Manejo de hilos y sincronización: Implementar el sistema utilizando múltiples hilos: uno para simular la lectura de temperatura y otro para monitorizar y detectar condiciones críticas. Usar mecanismos de sincronización (por ejemplo, mutexes o semáforos) para proteger el acceso concurrente a la variable que contiene la última lectura de temperatura.
- Registro de eventos: El sistema debe registrar en un archivo de log cada vez que la temperatura exceda el umbral y cuando se active la alarma.
- Configurabilidad: Permitir que el umbral de temperatura crítica y la frecuencia de lectura de temperatura sean configurables mediante argumentos de línea de comando o un archivo de configuración.

Entregables:

- Código fuente en C del sistema implementado.
- Documentación que incluya:
- - Instrucciones de compilación y ejecución.
- - Descripción de la arquitectura del sistema.
- - Explicación del manejo de interrupciones y cómo se gestionan los tiempos de respuesta.

Notas adicionales:

- Se sugiere utilizar rand() o random() para la generación de temperaturas.
- Se puede usar usleep() o nanosleep() para simular los intervalos de lectura.
- Para simular la alarma, basta con imprimir en pantalla un mensaje tipo "ALARMA: Temperatura crítica detectada: 41°C" o registrar el evento en el log.

Ing Storaccio Luis