PROYECTO (MONITOREO DE SENSORES)

Santiago Chitiva Contreras

Martín Medina Novoa

Andrés David Rueda Sandoval

Pontificia Universidad Javeriana

Sistemas Operativos

2024

Resumen

Este documento presenta una versión preliminar de un proyecto de monitoreo de sensores, compuesto por dos programas en C: `sensores.c` y `monitor.c`. `sensores.c` simula el comportamiento de un sensor que lee datos de archivos y los envía a través de un pipe nominal a `monitor.c`, que se encarga de recibir y procesar estos datos. El propósito de este proyecto es implementar un sistema de comunicación interprocesos que permita la recolección y monitoreo de datos de sensores de temperatura y pH.

Introducción

El monitoreo de sensores es crucial en diversas aplicaciones industriales y científicas. Este proyecto se enfoca en la implementación de un sistema que simula la recolección de datos de sensores y su transmisión a través de un pipe nominal para su posterior procesamiento. La finalidad es asegurar la correcta comunicación entre procesos y el manejo eficiente de datos utilizando técnicas de programación concurrente en C.

Paradigma de Programación

El proyecto emplea la programación concurrente utilizando hilos y mecanismos de sincronización como mutexes y semáforos. La comunicación entre procesos se realiza mediante pipes nominales, que permiten la transferencia de datos entre el proceso productor (`sensores.c`) y el proceso consumidor (`monitor.c`). Este enfoque asegura que los datos sean transferidos de manera segura y eficiente, evitando condiciones de carrera y asegurando la integridad de la información transmitida.

Metodología

Las librerías usadas en los programas son las estándar de C, como stdlib.h, stdio.h, string.h, fcntl.h (manejo de archivos), sys/stat.h (define estructuras de datos), sys/types.h (tipos de datos que se usan), unistd.h.

- **Sensores.c**

1. Definición de Constantes y Estructuras:

- Se define la constante `MAX` para el número máximo de datos que se pueden leer de un archivo.

- La estructura `sensoresData` contiene el número de datos y un vector de punteros a los datos leídos del archivo.

- La estructura `Argumentos` maneja los argumentos de entrada al programa.

2. Funciones Principales:

- `leerArchivo`: Lee los datos de un archivo línea por línea y los almacena en una estructura `sensoresData`.

- `verificar_argumentos`: Verifica y almacena los argumentos de línea de comandos en la estructura `Argumentos`.

- `main`: Verifica los argumentos, lee los datos del archivo correspondiente según el tipo de sensor, y envía estos datos a través de un pipe nominal.

3. Instrucciones de Uso:

- Ejecutar el programa con los argumentos `-s [tipo_sensor] -t [tiempo] -f [archivo] -p [pipe_nominal]`.

- **Monitor.c**

1. Definición de Constantes y Estructuras:

- Se define la constante `MAX_BUF` para el tamaño máximo del buffer.

- La estructura `Argumentos` maneja los argumentos de entrada.

2. Funciones Principales:

- `verificar_argumentos`: Verifica y almacena los argumentos de línea de comandos en la estructura `Argumentos`.

- `main`: Verifica los argumentos, abre el pipe nominal en modo lectura, lee los datos y los imprime hasta que no haya más datos.

3. Instrucciones de Uso:

- Asegurarse de que el pipe nominal exista y esté conectado correctamente, luego ejecutar el programa para recibir las mediciones.

Resultados

El sistema de monitoreo de sensores se probó exitosamente con sensores simulados de temperatura y pH. Los datos fueron leídos correctamente de los archivos y transmitidos a través del pipe nominal. El programa `monitor.c` recibió y procesó estos datos, mostrando que la comunicación interprocesos se realizó de manera eficiente y sin errores. Se implementaron y

verificaron mecanismos de sincronización para asegurar la integridad de los datos durante la transferencia.

Conclusiones

El proyecto demuestra la viabilidad de utilizar técnicas de programación concurrente y comunicación interprocesos para el monitoreo de sensores. La implementación con hilos y semáforos asegura una correcta sincronización y manejo de datos. Los resultados indican que el sistema puede ser escalable para incluir más tipos de sensores y métodos de procesamiento de datos. Futuras mejoras podrían enfocarse en la optimización del rendimiento y la inclusión de más funcionalidades de monitoreo y análisis de datos.