Sensor_data_sim

(Github Repository Link: https://github.com/PriyamMascharak/Sensor_data_sim.git)

A simple implementation of a buffer using a dynamic array to simulate sensor reading, processing, and display.

Overview

This program was written in response to an interview assignment given by Nosh Robotics. It implements a dynamic buffer system that simulates sensor data collection and processing.

Assignment Question

The following question was asked:

> Create a C program in an online compiler:
>
> 1. Use a timer to simulate data generated by external sensor. Setup the timer to trigger every second and generate a random number (0 to 5) of random bytes and add this to a globally accessible data structure.
>
> 2. Separately wake up periodically (every 10s), check if 50 bytes are stored in the globally accessible data structure and print only the latest 50 bytes (in hex value) and delete the printed bytes from the data structure.
>
> 3. For example, 1st second 4 bytes are added, 2nd second 3 bytes are added, and by 10th second there are 39 bytes in the buffer. Thus at 10th second data is not printed. At 20th second, if there are more than 50 bytes in the buffer, the main thread only prints the latest 50 bytes and deletes them.

Considerations:

- Make an application that can be run on online C compiler platforms like Programiz.

Implementation Details

Based on the requirements, the program maintains a buffer that can grow as needed and handles the "latest 50 bytes" requirement correctly.

The best data structure to implement this is an ArrayList (dynamic array). Since the requirement was for a globally accessible buffer, the ArrayList functions implemented have no reference parameters to make the code cleaner.

We use the `srand()` and `rand()` functions to initialize and generate random values every one second.

As Requested the program runs perfectly on the online c compiler

Example Output

When 50 bytes are collected:

A3, 17, F2, 8D, 3E, 9C, 45, 22, BB, 01,
6F, C4, 7A, 33, 91, 58, D0, 0F, 67, AE,
29, 84, 12, 5B, C3, 76, EF, 47, 9A, 52,
B8, 2F, 81, 19, D4, 63, A5, 0C, 7B, E2,
3D, 95, 4E, 27, 8F, 11, 69, D2, 40, B7,
removing 50 bytes from buffer

Extra Features

- Debug mode for monitoring data generation
- The counter will reset after reaching UINT_MAX-2 to prevent overflow