Department of Computer Systems and Information Technology

OPEN ENDED LAB

COMPUTER ENGINEERING WORKSHOP CS-219

SE CS BATCH 2023

FALL SEMESTER 2024

SECTION: B

GROUP MEMBERS:

- **Abdul Moazzim (CS-23048)**
- **Nishwa (CS-23078)**
- **Ayesha Faiz (CS-23140)**

PROBLEM DESCRIPTION:

The Integrated Environmental Monitoring System (IEMS) is designed to address the need for efficient environmental data collection, processing, and reporting using contemporary computer engineering technologies. This project leverages the C programming language to implement a robust system that interacts with a free API to retrieve real-time environmental data, such as temperature and humidity, and integrates several fundamental programming concepts and practical applications.

PROBLEM OUTLINES:

- Utilize a free API to fetch real-time environmental data. e.g.: temperature, humidity.
- Store raw data in structured files for future use and analysis.
- Create shell scripts to automate data retrieval and processing tasks.
- Employ pointers and dynamic memory allocation to handle data manipulation.
- Implement real-time notifications using Linux system calls to alert relevant personnel of critical environmental conditions.
- Use header files to organize and modularize the code for better readability and maintainability.

METHODOLOGY

OBJECTIVE:

The objective of the Integrated Environmental Monitoring System (IEMS) is to develop a system for retrieving, processing, and reporting real-time environmental data, such as temperature and humidity, using C programming. The system will interact with a free API to collect environmental data and process it for insights. It will store raw and processed data efficiently using dynamic memory allocation and file handling. Shell scripts will automate tasks like data retrieval and processing. Real-time alerts will be implemented using Linux system calls to notify personnel of critical readings. The system will utilize a modular design with header files to ensure readability and maintainability.

CORE FUNCTIONALITIES:

- Real time data Retrieval: Interacts with a free API to fetch real-time environmental data (e.g., temperature, humidity).
- Data Processing: Processes the retrieved data to extract meaningful insights (e.g., averages, trends).

- Data Storage: Stores both raw and processed data in files for future reference and analysis.
- Task Automation: Utilizes shell scripts to automate the periodic retrieval, processing, and storage of data.
- Real time alerts: Implements real-time notifications using Linux system calls for critical environmental readings (e.g., temperature exceeding a threshold).
- Memory Management: Uses pointers and dynamic memory allocation in C for efficient data manipulation and storage.
- Modular Code Design: Organizes the code into separate modules using header files to enhance readability, maintainability, and scalability.

IMPLEMENTATION DETAILS:

- ➢ Data Retrieval (API Interaction): This program fetches real-time environmental data using the CURL library in C. The API provides temperature, humidity, and other environmental metrics in JSON format.

  → API used: OpenWeatherMap (or any free alternative)

  → Process:
  - Initialize CURL to make HTTP GET requests.
  - Parse the returned JSON data using the cJSON library
- ➢ Data Storage: Raw and processed data are stored in files using C’s standard file I/O operations like, fopen , fprintf and fclose.
- ➢ Data Processing: The retrieved JSON data is parsed to extract relevant metrics (e.g., temperature and humidity). These metrics are formatted and saved to the processed data file.

  → Dynamic Memory Allocation: Use malloc and free to allocate memory for parsed data dynamically.
- ➢ Automation (Shell Scripting): Shell scripts are used to schedule data retrieval and processing at regular intervals using cron.
- ➢ Real Time Alerts: Real-time alerts are implemented using Linux system calls. When critical thresholds are detected (e.g., high temperature), the system sends notifications via fork and exec.
- ➢ Modular Code Structure: Header files are used to organize the program into separate modules:

  → api.h: Handles API interactions.

→ Processing.h: Contains data processing logic.

→ Alerts.h: Manages real-time alerts.

KEY FEATURES:

DYNAMIC MEMORY USAGE	Allocates memory dynamically for efficient data handling.
LINUX SYSTEM CALL	Real-time alerts using system calls for notifications.
AUTOMATION:	Shell scripts for scheduled operations via cron.
CODE MODULARITY:	Header files for clear and maintainable code structure.
FILE MANAGEMENT:	Separate storage of raw and processed data for better tracking.

CHALLENGES AND SOLUTIONS:

- ➢ Handling API errors:
  - → Used robust error handling to manage failed requests
  - → Implemented retries with exponential backoff.
- ➢ Memory Management:
  - → Ensured proper deallocation of dynamically allocated memory to prevent leaks.

- Real time execution:
  → Integrated Linux system calls effectively for real-time functionality.

CONCLUSION:

This project effectively showcases fundamental concepts of computer engineering through C programming, including API interaction, file management, dynamic memory allocation, and modular programming. By integrating automated task execution and real-time alert systems, it offers a practical and efficient solution for environmental monitoring, ensuring timely responses to critical changes in monitored conditions. This system not only demonstrates technical skills but also addresses real-world applications, enhancing the overall effectiveness of environmental control.