Department of Computer Systems and Information Technology

OPEN ENDED LAB

COMPUTER ENGINEERING WORKSHOP CS-219

SE CS BATCH 2023

FALL SEMESTER 2024

SECTION: A

- **DAMIYA NAEEM (CS-027)**

Problem Description

Construct an integrated environmental monitoring system in C, covering a range of fundamental concepts and practical applications. The project involves interacting with a free API that provides real-time environmental data. The system's core functionalities include data retrieval, processing and reporting.

Problem Outline:

- Interact with a free API to retrieve real-time environmental data (e.g., temperature, humidity).
- Store raw and processed data in files.

- Create shell scripts to automate tasks such as data retrieval and processing.
- Utilize pointers and dynamic memory allocation in the C program to optimize data manipulation and enhance
- efficiency
- Implement real-time alerts using Linux system calls to notify relevant personnel of critical environmental readings.
- Use header files to modularize the C code and enhance code readability.

Methodology

Objective

The primary goal of this project is to construct a robust environmental monitoring system in C that leverages real-time data from a free API. The system will showcase key computer engineering concepts, such as API interaction, data processing, file management, dynamic memory allocation, and Linux system integration.

System Overview

Core Functionalities

1. Data Retrieval: Interact with an open environmental API to collect real-time environmental data (e.g., temperature, humidity).
2. Data Processing: Analyze and process raw data to extract meaningful insights and detect critical conditions.
3. Data Storage: Store raw and processed data in files for historical record-keeping.
4. Automation: Use shell scripts to automate data retrieval and processing tasks.
5. Real-Time Alerts: Implement real-time notifications for critical readings using Linux system calls.
6. Code Modularity: Use header files for modular code structure, improving readability and maintainability.

Implementation Details

1. Data Retrieval (API Interaction)

The program retrieves 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 cURLto make HTTP GET requests.

    - Parse the returned JSON data using the cJSON library.

2. Data Storage

Raw and processed data are stored in files using C’s standard file I/O operations (fopen , fprintf , fclose).

- **File Structure**:
    - Raw Data File: Contains unprocessed API responses.
    - Processed Data File: Contains formatted and summarized data for easy interpretation.

3. 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.

4. Automation (Shell Scripting)

Shell scripts are used to schedule data retrieval and processing at regular intervals using cron.

5. 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.

6. 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 Calls	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

1. **Handling API Errors**:
    - Used robust error handling to manage failed requests.
    - Implemented retries with exponential backoff.
2. **Memory Management**:
    - Ensured proper deallocation of dynamically allocated memory to prevent leaks.
3. **Real-Time Execution**:
    - Integrated Linux system calls effectively for real-time functionality.

Conclusion

This project successfully demonstrates key concepts of computer engineering using C, such as API interaction, file management, dynamic memory allocation, and modular programming. By automating tasks and implementing real-time alerts, the system provides a practical solution for environmental monitoring.