COMPUTER ENGINEERING WORKSHOP

S.E. (CIS) OEL REPORT

Project Group ID:

NAME OF MEMBER #1 Mashaal Ali CS-# 91
NAME OF MEMBER #2 Anas Yazdani CS-# 102
NAME OF MEMBER #3 Osama Humayun CS-# 139

BATCH: 2023

Department of Computer and Information Systems Engineering

NED University of Engg. & Tech.,
Karachi-75270

CONTENTS

S.No. **Page No.**

CHAPTER 1

PROBLEM DESCRIPTION

Monitoring environmental parameters such as temperature and humidity is crucial for industries like agriculture, healthcare, disaster management, and urban planning. Rapid changes in these parameters can significantly impact crop yields, public health, and climate resilience. Traditional weather monitoring systems often rely on manual data collection and analysis, resulting in delayed responses and inefficiencies.

This project aimed to address these issues by designing and implementing an **automated weather monitoring system**. The system was developed using **C programming** and Linux utilities, incorporating real-time data fetching, data processing, critical alert generation, and workflow automation. The primary goals of this project were:

1. **Fetching Real-Time Data:** Integrate with the OpenWeather API to collect weather information and log raw outputs for validation.
2. **Data Processing:** Convert raw API data into a user-friendly format, highlighting key metrics like temperature, humidity, and weather descriptions.
3. **Critical Alerts:** Detect and respond to predefined thresholds for temperature and humidity by generating alerts with real-time timestamps.
4. **Automation:** Automate the entire workflow using **Linux crontab**, ensuring consistent operation without manual intervention.

This project offers a scalable and cost-effective solution for weather monitoring, adaptable for various use cases, from personal research to industrial-scale implementations.

CHAPTER 2

METHODOLOGY

The weather monitoring system was designed using a modular approach to ensure reliability, scalability, and efficiency. The following steps summarize the methodology:

1. Data Fetching

The OpenWeather API was used to fetch real-time weather data. The fetch_data() function employed the **libcurl** library to send HTTP GET requests, dynamically storing the JSON response using a custom memory structure. Raw API responses were logged in **raw_data.txt** for future reference and debugging.

2. Data Processing

The fetched JSON data was parsed using the **Jansson** library to extract critical environmental parameters such as:

- **Temperature:** Converted from Kelvin to Celsius.

- **Humidity:** Expressed as a percentage.
- **Weather Descriptions:** Summarized for qualitative insights (e.g., "haze").

This processed data was saved in **processed_data.txt**, formatted for easy readability and further analysis. Each entry included the timestamp of the data retrieval, enabling trend identification over time.

3. Critical Alerts

Predefined thresholds for temperature (>20°C) and humidity (>10%) were set to identify critical conditions. If these thresholds were exceeded, the system triggered the following actions:

- Logging alerts in **alerts.txt**, including detailed messages with timestamps.
- Printing alert messages to the console for real-time monitoring.
- Hypothetically sending email notifications using the Linux mail command.

The alert system ensures timely detection and response to adverse weather conditions.

4. Statistical Analysis

To provide insights into trends, the system calculated averages of temperature and humidity based on the most recent five readings. The save_avg_data() function extracted relevant entries from **processed_data.txt**, computed averages, and stored the results in **avg_data.txt**. This feature aids in understanding broader patterns in environmental conditions.

5. Automation with Crontab

The system's workflow was automated using **Linux crontab**, ensuring hourly execution without manual intervention. A shell script was designed to compile and run the program. Key commands included:

- **Compilation:** gcc main.c functions.c -o env_monitor -lcurl -ljansson
- **Automation:** 0 * * * * cd /path_to_project && ./run_monitor.sh

This setup guaranteed consistent data collection and processing, enhancing the system's efficiency and reliability.

CHAPTER 3

RESULTS

1. Raw Data Collection

Raw weather data fetched from the OpenWeather API was saved in **raw_data.txt**. This data served as the foundation for all subsequent processing and analysis.

2. Processed Data

Processed weather metrics such as temperature, humidity, and weather descriptions were saved in **processed_data.txt**. Example entry:

TEMPERATURE: 28.90°C
HUMIDITY: 48.00%
WEATHER DESCRIPTION: haze

3. Critical Alerts

The system detected and logged critical environmental conditions. Example entries from **alerts.txt**:

ALERT TIME: 2024-11-21 14:32:00
Critical Temperature Alert! Temperature: 28.90°C

These alerts provide actionable insights, enabling users to respond promptly.

4. Average Data Calculation

The system calculated and saved averages of the most recent five readings in **avg_data.txt**. Example entry:

AVERAGE TEMPERATURE: 28.90°C
AVERAGE HUMIDITY: 48.00%

5. Automation Performance

Hourly execution using Linux crontab ensured consistent operation. Outputs in **processed_data.txt** and **alerts.txt** confirmed seamless automation.

Conclusion and Future Work

This project successfully implemented an automated weather monitoring system using C programming and Linux tools. By integrating real-time API data with data processing, alert generation, and automation, the system provides a robust solution for environmental monitoring.

Key Outcomes:

- Automated real-time data collection and analysis.
- Reliable detection of critical weather conditions.
- User-friendly data outputs for trend analysis.

Future Enhancements:

1. **Dynamic Configuration:** Allow users to modify alert thresholds without recompilation.
2. **Expanded Metrics:** Include additional parameters like wind speed and air pressure.
3. **Visualization Tools:** Add graphical representations of weather trends.
4. **Multi-Location Support:** Enable monitoring of multiple regions simultaneously.

This project showcases how programming and automation can address real-world challenges in weather management, offering a foundation for future advancements in environmental monitoring systems.

----------END----------

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-219
Course Title: Computer Engineering Workshop
Open Ended Lab
SE Batch 2023, Fall Semester 2024
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1		
S2		
S3		

CRITERIA AND SCALES				Marks Obtained		
				S1	S2	S3
Criterion1: Has the student implemented an efficient and scalable solution for data retrieval, processing, and reporting?						
0	1	2	3			
The student has not even implemented a basic solution that meets the project's requirements.	The student has implemented a basic solution that meets the project's requirements but may lack optimization in certain aspects.	The student has implemented a proficient and well-optimized solution.	The student has implemented an exceptionally efficient and scalable solution.			
Criterion 2: Has student demonstrated a strong understanding of C programming fundamentals?						
0	1	2	3			
The student doesn't have basic understanding of C programming fundamentals.	The student exhibits a basic understanding of C programming fundamentals.	The student demonstrates a strong understanding of C programming fundamentals.	The student demonstrates an exceptional understanding of C programming fundamentals.			
Criterion 3: How well written is the report?						
0	1	2	3			
The submitted report is unfit to be graded.	The report is partially acceptable.	The report is complete and concise.	The report is exceptionally written.			
			Total Marks:			