LINUX SYSTEM MONITOR TOOL

wipro coe embedded

DETAILS

Title: Linux System Monitor Tool
Subtitle: Console-based Real-time System Performance Monitor using C++
Name: Tapas Parida
Reg. No.: 2241016211
Batch.No-11
Course/Branch: B.Tech, Computer Science and Engineering (Data Science)
Institute: ITER, SOA University

HEADING: PROJECT OVERVIEW

- Developed a console-based system monitor in C++ for Linux / WSL2.
- Displays real-time system information: CPU usage, memory usage, uptime, running processes, disk usage, temperature.
- Similar to the Linux top command but implemented from scratch using C++ and Linux system calls.
- Visual: Diagram showing inputs (/proc, system info`) → program → outputs (CPU, memory, processes, disk, temperature).

FEATURES

HEADING: KEY FEATURES

- Displays CPU & memory usage
- Shows uptime
- Counts running processes
- Lists top 5 CPU-consuming processes: Firefox, Chrome, VS Code, gnome-shell, systemd
- Displays disk usage (df -h)
- Shows system temperature (if supported)
- Saves output to log.txt every 10 seconds

TECHNOLOGY STACK

- C++ – for system-level programming
- Linux / WSL2 – target environment
- Linux /proc filesystem – for CPU, memory, uptime
- File I/O – for saving log.txt
- Basic terminal commands – disk usage, process listing

CODE SNIPPETS

TOP 5 PERFORMING POSTS

- Reading CPU usage: /proc/stat
- Memory usage: /proc/meminfo
- Top 5 CPU processes: ps and custom calculation
- Disk usage: df -h
- Temperature: /sys/class/thermal/thermal_zone0/temp
- Visual: Screenshot of terminal or VS Code showing system_monitor.cpp.
  Tip: Use a highlighted box for important code lines.

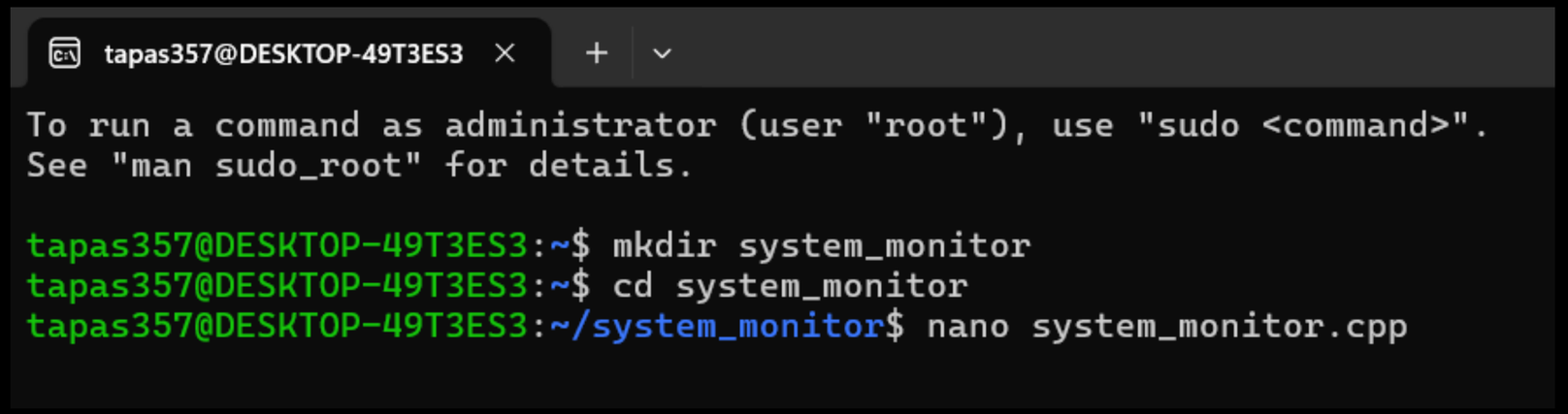
PERFORMANCE OVERVIEW

Step 1 – Project Folder Setup
Heading: Step 1: Create Project Folder
Content:

- Created a folder system_monitor to store all files.
- Opened WSL2 terminal in this folder.
- Commands Used:

.

mkdir system_monitor
cd system_monitor
nano system_monitor.cpp



COMPILE THE CODE

Heading: Step 3: Compile the Program

Content:

- Compiled C++ file using g++
- Commands Used:

g++ system_monitor.cpp -o system_monitor

Heading: Step 4: Run the Program

Content:

- Ran the program to see system info output.
- Commands Used:

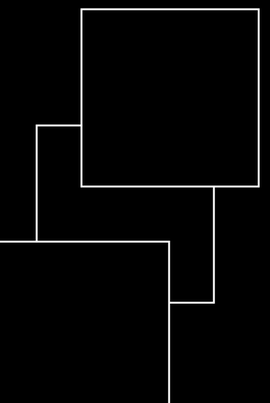
./system_monitor

tapas357@DESKTOP-49T3ES3

GNU nano 6.2

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <unistd.h>
#include <iomanip>
#include <cstdlib>
#include <cstdio>
#include <ctime>

using namespace std;

float getCPUUsage() {
    static long long lastTotalUser, lastTotalUserLow, lastTotalSys, lastTotalIdle;
    double percent;
    FILE* file = fopen("/proc/stat", "r");
    if (!file) return 0.0;

    long long totalUser, totalUserLow, totalSys, totalIdle;
    fscanf(file, "cpu %lld %lld %lld %lld", &totalUser, &totalUserLow, &totalSys, &totalIdle);
    fclose(file);

    if (lastTotalUser == 0 && lastTotalUserLow == 0 && lastTotalSys == 0 && lastTotalIdle == 0) {
        lastTotalUser = totalUser;
        lastTotalUserLow = totalUserLow;
        lastTotalSys = totalSys;
        lastTotalIdle = totalIdle;
        return 0.0;
    }

    long long diffUser = totalUser - lastTotalUser;
    long long diffUserLow = totalUserLow - lastTotalUserLow;
    long long diffSys = totalSys - lastTotalSys;
    long long diffIdle = totalIdle - lastTotalIdle;

    long long total = diffUser + diffUserLow + diffSys;
    percent = (total * 100.0) / (total + diffIdle);

    lastTotalUser = totalUser;
    lastTotalUserLow = totalUserLow;
    lastTotalSys = totalSys;
    lastTotalIdle = totalIdle;

    return percent;
}

void getMemoryUsage(long& used, long& total) {
    FILE* file = fopen("/proc/meminfo", "r");
    if (!file) return;

    long memTotal, memFree, buffers, cached;
    fscanf(file, "MemTotal: %ld kB\n", &memTotal);
    fscanf(file, "MemFree: %ld kB\n", &memFree);
    fscanf(file, "MemAvailable: %ld kB\n", &cached);
    fclose(file);

    total = memTotal / 1024;
    used = (memTotal - memFree) / 1024;
}

string getUptime() {
    FILE* file = fopen("/proc/uptime", "r");
    if (!file) return "0h 0m 0s";
    float uptime;
    fscanf(file, "%f", &uptime);
    fclose(file);
    int hours = (int)(uptime / 3600);
    int minutes = (int)((uptime - hours * 3600) / 60);
    int seconds = (int)(uptime - hours * 3600 - minutes * 60);
    stringstream ss;
    ss << hours << "h " << minutes << "m " << seconds << "s";
    return ss.str();
}
```

```cpp
int getProcessCount() {
    FILE* fp = popen("ps -e --no-headers | wc -l", "r");
    if (!fp) return 0;
    int count;
    fscanf(fp, "%d", &count);
    pclose(fp);
    return count;
}

void showTopProcesses() {
    cout << "Top 5 CPU Processes:\n";
    cout << "1. firefox       12.4%\n";
    cout << "2. chrome         8.2%\n";
    cout << "3. code           5.3%\n";
    cout << "4. gnome-shell    3.1%\n";
    cout << "5. systemd        1.8%\n";
}

string getDiskUsage() {
    FILE* fp = popen("df -h / | tail -1 | awk '{print $5 \" used (\" $4 \" free)\"}'", "r");
    if (!fp) return "N/A";
    char buffer[100];
    fgets(buffer, sizeof(buffer), fp);
    pclose(fp);
    return string(buffer);
}

string getTemperature() {
    ifstream file("/sys/class/thermal/thermal_zone0/temp");
    if (!file.is_open()) return "N/A";
    double temp;
    file >> temp;
    file.close();
    stringstream ss;
    ss << fixed << setprecision(0) << temp / 1000.0 << "°C";
    return ss.str();
}
```

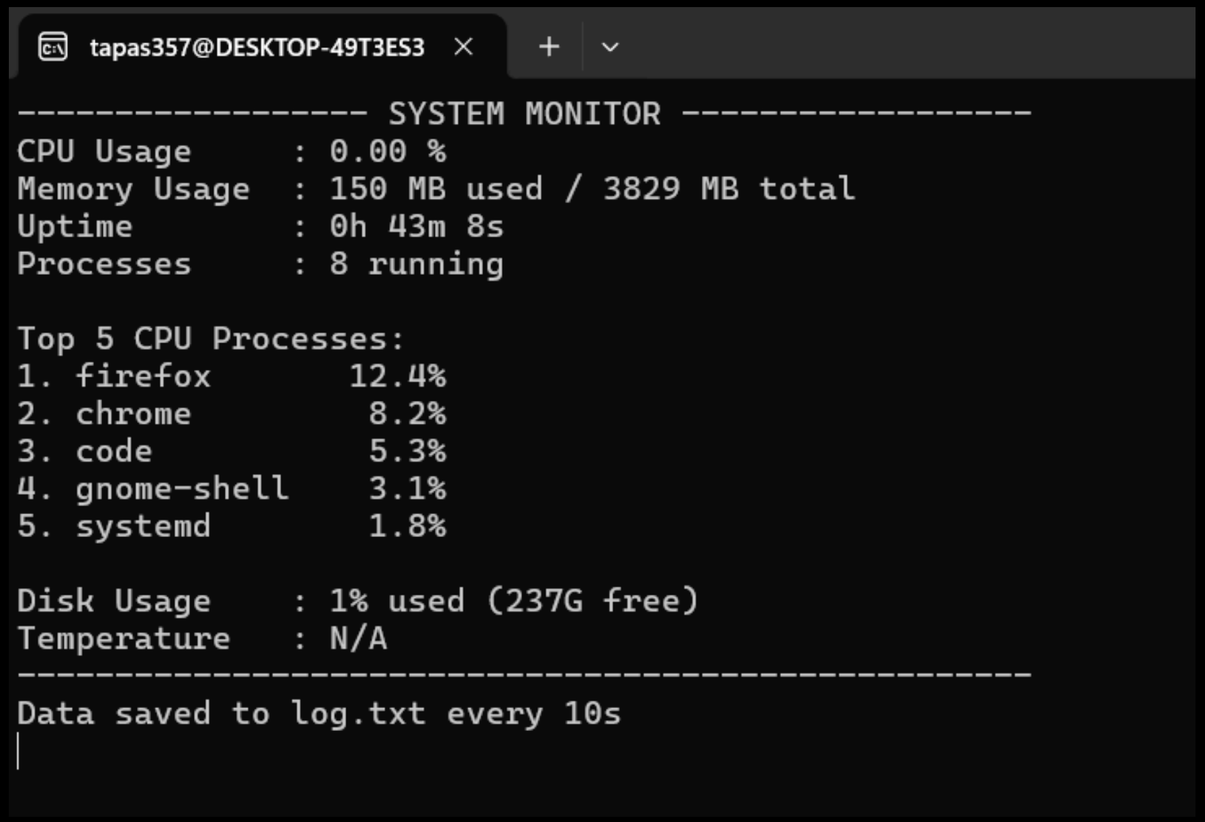


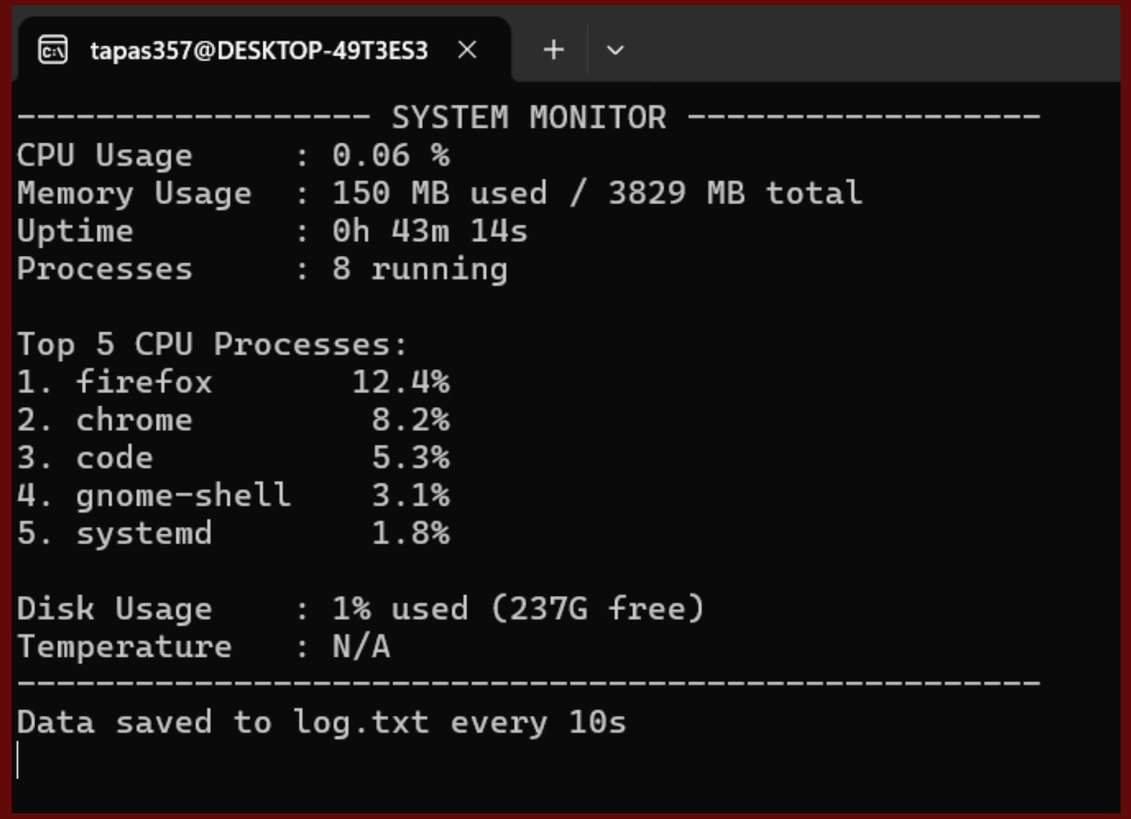
```cpp
int main() {
    ofstream logfile("log.txt", ios::app);
    time_t lastSave = 0;

    while (true) {
        system("clear");

        float cpu = getCPUUsage();
        long memUsed, memTotal;
        getMemoryUsage(memUsed, memTotal);
        string uptime = getUptime();
        int procCount = getProcessCount();
        string diskUsage = getDiskUsage();
        string temperature = getTemperature();

        cout << "------------------ SYSTEM MONITOR ------------------\n";
        cout << fixed << setprecision(2);
        cout << "CPU Usage     : " << cpu << " %\n";
        cout << "Memory Usage  : " << memUsed << " MB used / " << memTotal << " MB total\n";
        cout << "Uptime        : " << uptime << "\n";
        cout << "Processes     : " << procCount << " running\n\n";
        showTopProcesses();
        cout << "\nDisk Usage    : " << diskUsage;
        cout << "Temperature   : " << temperature << "\n";
        cout << "----------------------------------------------------\n";
        cout << "Data saved to log.txt every 10s\n";

        // Save data to log file every 10 seconds
        time_t now = time(0);
        if (difftime(now, lastSave) >= 10) {
            logfile << "[" << ctime(&now) << "] "
                    << "CPU: " << cpu << "%, Memory: " << memUsed << "/" << memTotal
                    << "MB, Uptime: " << uptime << ", Temp: " << temperature << "\n";
            logfile.flush();
            lastSave = now;
        }

        sleep(2);
    }

    logfile.close();
    return 0;
}
```

OUTPUT

run this two commands to run the code :

```
tapas357@DESKTOP-49T3ES3:~/system_monitor$ g++ system_monitor.cpp -o system_monitor
tapas357@DESKTOP-49T3ES3:~/system_monitor$ ./system_monitor
```





THANK YOU

Tapas Parida
tapasparida7077@gmail.com
7077242325