#include <iostream>

#include <gps.h>

int main()

{

// Open GPS connection

gps\_data\_t gpsData;

if (gps\_open("localhost", "2947", &gpsData) == -1) {

std::cerr << "Failed to open GPS connection." << std::endl;

return 1;

}

// Enable streaming of GPS data

gps\_stream(&gpsData, WATCH\_ENABLE | WATCH\_JSON, NULL);

// Main loop to read and process GPS data

while (true) {

// Wait for new data

if (gps\_waiting(&gpsData, 500)) {

// Read the data

if (gps\_read(&gpsData) == -1) {

std::cerr << "Failed to read GPS data." << std::endl;

break;

}

// Check if fix data is available

if (gpsData.status == STATUS\_FIX) {

// Get latitude and longitude

double latitude = gpsData.fix.latitude;

double longitude = gpsData.fix.longitude;

// Print latitude and longitude

std::cout << "Latitude: " << latitude << std::endl;

std::cout << "Longitude: " << longitude << std::endl;

}

else {

std::cerr << "No GPS fix available." << std::endl;

}

}

}

// Close GPS connection

gps\_stream(&gpsData, WATCH\_DISABLE, NULL);

gps\_close(&gpsData);

return 0;

}

This code uses the **gps\_open** function to open a connection to the GPS sensor. It then enables the streaming of GPS data using **gps\_stream** with the **WATCH\_ENABLE** and **WATCH\_JSON** flags. Inside the main loop, it waits for new data using **gps\_waiting** and reads the data using **gps\_read**. If a valid fix is available, it retrieves the latitude and longitude from the **gps\_data\_t** structure and prints them to the console.

Make sure to link against the **libgps** library when compiling the code. The library can be installed using the package manager of your operating system (e.g., **libgps-dev** on Debian-based systems).

Note that this code assumes that the GPS daemon (**gpsd**) is running on the local machine, listening on port 2947. Adjust the parameters passed to **gps\_open** according to your setup if necessary.