Streams HLS based on two different files as a live service within a 30-second window in C++

In this example, the code reads two `.m3u8` playlist files into memory, and then generates an output playlist in real-time, alternating between the segments from the two input playlists, and including a `#EXT-X-PROGRAM-DATE-TIME` header with the current time. The output playlist is updated every 30 seconds, and the code waits for the next window by sleeping the main thread for 30 seconds.

Note that this code is a basic example and does not handle edge cases such as when one of the input playlists is exhausted, or when the number of segments in the window is not an exact multiple of the number of playlists.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <chrono>
#include <thread>

const int window_size = 30;
const std::string playlist_1 = "playlist_1.m3u8";
const std::string playlist_2 = "playlist_2.m3u8";
const std::string output = "output.m3u8";

std::vector<std::string> read_playlist(const std::string& filename)
{
    std::vector<std::string> playlist;
    std::ifstream file(filename);
    std::string line;
    while (std::getline(file, line))
    {
        playlist.push_back(line);
    }
    return playlist;
}

int main()
{
    // Read playlists
    auto playlist1 = read_playlist(playlist_1);
    auto playlist2 = read_playlist(playlist_2);

    // Create output playlist
    std::ofstream stream(output);
    stream << "#EXTM3U\n";

    int playlist1_index = 0;
    int playlist2_index = 0;
    while (true)
    {
        // Get current time
        auto now = std::chrono::system_clock::now();

        // Write header
        stream << "#EXT-X-PROGRAM-DATE-TIME:"
               << std::chrono::time_point_cast<std::chrono::seconds>(now).time_since_epoch().count()
               << "\n";

        // Write segments
        for (int i = 0; i < window_size; ++i)
        {
            if (playlist1_index < playlist1.size())
            {
                stream << playlist1[playlist1_index++] << "\n";
            }
            if (playlist2_index < playlist2.size())
            {
                stream << playlist2[playlist2_index++] << "\n";
            }
        }

        // Wait for next window
        std::this_thread::sleep_for(std::chrono::seconds(window_size));
    }

    return 0;
}
```