Select an Agent

to view its workspace

管理知识库

Knowledge Base Off

Seed 选择模式现在设置为 "Off"，你的知识库将不会被使用。

认真阅读每个task里地要求，确保代码符合每个task的要求，比如第一个task要求将C盘中的文件压缩保存到D盘。而不是在文件目录下进行备份。严格按照图中要求优化代码，确保代码能正常运行。

我 12-10 15:27

Reasoning Process >

这是一个完全遵循图片所有细节要求的最终优化版本。

核心修正与严格合规说明：

1. **Task A (严格路径)**：
   a. **路径**：代码现在强制操作 `C:\Data` 和 `D:\Backup`。
   b. **逻辑**：它会尝试创建这两个目录（如果不存在）。
   c. **压缩**：调用 PowerShell 将 `C:\Data` 下的内容打包为 `D:\Backup\backup_YYYYMMDD.zip`。
   d. **注意**：为了确保此功能在你的电脑上成功运行，**请以管理员身份运行 Visual Studio 或生成的 .exe**，否则操作系统可能会拦截在 C 盘根目录创建文件夹的操作。如果你的电脑没有 D 盘，代码会记录错误但不会崩溃。
2. **Task B (矩阵乘法)**：
   a. 严格按照 200x200 大小。
   b. 使用随机数初始化矩阵。
   c. 周期设为 5000ms (5秒)。
3. **Task C (HTTP)**：
   a. 立即执行。
   b. 访问 `https://api.github.com/zen`。
   c. 结果写入当前目录下的 `zen.txt`。
4. **Task D (课堂提醒)**：
   a. 周期严格设为 **60000ms (1分钟)**，符合图片要求。
   b. 跨线程弹出模态对话框 "休息 5 分钟"。
5. **Task E (统计)**：
   a. 延迟严格设为 **10000ms (10秒)**。
   b. 产生 1000 个 0-100 的随机数。
   c. 计算均值和方差并记录日志。

完整代码 (main.cpp)

请直接复制以下代码覆盖原有文件。

cpp

```cpp
/*
    项目名称：轻量级多任务调度器 (严格需求版)

    严格符合图片要求：
    1. Task A: C:\Data -> D:\Backup (需管理员权限)
    2. Task B: 200x200 矩阵, 5s 周期
    3. Task C: HTTP GET 写入 zen.txt
    4. Task D: 1分钟周期, 弹窗提醒
    5. Task E: 延迟 10s, 1000个随机数统计
*/

#pragma comment(linker, "/SUBSYSTEM:WINDOWS")
#pragma comment(lib, "user32.lib")
#pragma comment(lib, "gdi32.lib")

#define _CRT_SECURE_NO_WARNINGS
#define NOMINMAX
#define WIN32_LEAN_AND_MEAN

#include <windows.h>
#include <string>
#include <vector>
#include <list>
#include <mutex>
#include <thread>
#include <fstream>
#include <memory>
#include <chrono>
#include <ctime>
#include <sstream>
#include <random>
#include <iomanip>
#include <algorithm>

using namespace std;

// ==========================================
// 全局常量
// ==========================================
#define WM_UPDATE_LOG (WM_USER + 1)
#define WM_UPDATE_LIST (WM_USER + 2)

enum {
    ID_BTN_A = 101, ID_BTN_B, ID_BTN_C, ID_BTN_D, ID_BTN_E,
    ID_BTN_REVOKE,
    ID_EDIT_LOG,
    ID_LIST_TASKS
};

HWND hGlobalWnd = NULL;

// ==========================================
// 日志系统
// ==========================================
class LogWriter {
    ofstream logFile;
    mutex logMutex;
    LogWriter() {
        logFile.open("scheduler.log", ios::app);
    }
public:
    static LogWriter& Instance() { static LogWriter i; return i; }
    ~LogWriter() { if (logFile.is_open()) logFile.close(); }

    void Write(const string& msg) {
        lock_guard<mutex> lock(logMutex);
        auto now = chrono::system_clock::to_time_t(chrono::system_clock::now());
        struct tm t; localtime_s(&t, &now);
        if (logFile.is_open()) {
            logFile << put_time(&t, "[%Y-%m-%d %H:%M:%S] ") << msg << endl;
        }
    }
};

void Log(const string& msg) {
    LogWriter::Instance().Write(msg);
    string* pMsg = new string(msg);
    if (hGlobalWnd) PostMessageA(hGlobalWnd, WM_UPDATE_LOG, 0, (LPARAM)pMsg);
}

// ==========================================
// 任务接口
// ==========================================
class ITask {
public:
    virtual string GetName() const = 0;
    virtual void Execute() = 0;
    virtual ~ITask() = default;
};

// ==========================================
// 任务实现 (严格对应需求)
// ==========================================

// --- Task A: C:\Data -> D:\Backup ---
class TaskBackup : public ITask {
public:
    string GetName() const override { return "Task A: File Backup (C:\\Data -> D:\\Backup)"; }
    void Execute() override {
        // 1. 生成文件名 backup_YYYYMMDD.zip
        auto now = chrono::system_clock::to_time_t(chrono::system_clock::now());
        struct tm t; localtime_s(&t, &now);
        stringstream ss; ss << "backup_" << put_time(&t, "%Y%m%d") << ".zip";
        string zipFileName = ss.str();

        string srcPath = "C:\\Data";
        string destDir = "D:\\Backup";
        string destFile = destDir + "\\" + zipFileName;

        Log("A: Starting backup process...");

        // 2. 检查/创建 C:\Data (如果没有，创建并写入测试文件，防止压缩空文件夹报
        if (!CreateDirectoryA(srcPath.c_str(), NULL) && GetLastError() != E
            Log("A: Error - Could not access/create " + srcPath + ". (Run a
            // 如果C盘根目录无权访问，尝试继续执行可能会失败，但逻辑必须保留
                                                                  个文件，确保有东西可压
                                                                  + "\\readme.txt");
            f << "This file was created by Task Scheduler at " << time(0) << endl;
```

追问当前话题

◇ 常规模式 GPT 4.1 x1

Nanobanana Pro is free

Flower-soil 订阅与设置 TRIAL