



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。

```

}
// ****
// 7. UI 界面 (Console UI & Observer 实现)
// ****
class ConsoleUI : public IOObserver {
private:
    mutex uiMutex;
public:
    ConsoleUI() {
        TaskScheduler::Instance().AddObserver(this);
    }

    // 收到调度器的通知，刷新状态(这里简单的打印出来)
    void OnTaskEvent(const string& message) override {
        lock_guard<mutex> lock(uiMutex);
        // 为了不打断输入行，这里简单处理
        cout << "[System Notification] " << message << "\n" << flush;
    }
}

void ShowMenu() {
    while (true) {
        {
            lock_guard<mutex> lock(uiMutex);
            cout << "\n" + "-----\n";
            cout << " Lightweight Multi-Task Scheduler \n";
            cout << "-----\n";
            cout << "[A] File Backup (One-time, Delayed)\n";
            cout << "[B] Matrix Calc (Periodic: 5s)\n";
            cout << "[C] HTTP GET (Immediate)\n";
            cout << "[D] Reminder (Periodic: 1min, Popup)\n";
            cout << "[E] Random Stats (Delayed: 10s)\n";
            cout << "[Q] Quit\n";
            cout << "-----\n";
            cout << "Select Task to Schedule > " << flush;
        } // 释放锁等待输入

        char choice;
        cin >> choice;
        choice = toupper(choice);

        if (choice == 'Q') break;

        shared_ptr<ITask> task = TaskFactory::CreateTask(choice);
        if (task) {
            int delay = 0;
            int interval = 0;

            // 根据题目要求配置预设参数
            switch (choice) {
                case 'A': delay = 2000; break; // 延迟2秒方便观察
                case 'B': interval = 5000; break; // 周期5秒
                case 'C': delay = 0; break; // 立即
                case 'D': interval = 60000; break; // 周期1分钟 (调试可改为5000)
                case 'E': delay = 10000; break; // 延迟10秒
            }

            TaskScheduler::Instance().AddTask(task, delay, interval);
        } else {
            cout << "Invalid selection!" << endl;
        }

        // 简单防抖
        this_thread::sleep_for(chrono::milliseconds(200));
    }
}
};

// ****
// Main Entry
// ****
int main() {
    // 设置控制台标题
    system("title Lightweight Task Scheduler - Project 3");

    cout << "Initializing Scheduler..." << endl;

    // 初始化UI并运行主线程
    ConsoleUI ui;
    ui.ShowMenu();

    cout << "Stopping Scheduler..." << endl;
    TaskScheduler::Instance().Stop();

    cout << "Goodbye!" << endl;
    return 0;
}

```

#### 修复说明：

1. 移除 C++17 依赖：删除了 `#include <filesystem>` 和 `namespace fs = ...;`，因为你的 Visual Studio 设置为 C++14 或更早版本。现在代码使用通用的文件流，兼容性更好。
2. 安全性增强：添加了 `#define NOMINMAX` 来防止 Windows 头文件干扰 C++ 标准库。使用了 `localtime\_s` 替代 `localtime` 以符合 Visual Studio 的安全要求。
3. 运行方式：直接将上述代码复制到 Visual Studio 中运行即可，无需更改项目配置。

Gemini 3 Pro Preview

12:10 14:39



#### Reasoning Process >

这两个错误分别对应了 **代码逻辑初始化** 问题和 **项目配置** 问题。

1. **LNK2019 / WinMain 错误**：这是最关键的错误。你的 Visual Studio 项目被创建为了“Windows 桌面应用程序”（它寻找 `WinMain` 入口），但我的代码写的是“控制台应用程序”（使用 `main` 入口）。
2. **C26495 警告**：这是因为结构体内部的变量没有默认初始化。
3. **C26115 警告**：这是静态分析器对锁范围的误报，虽然逻辑是对的，但我会优化一下写法让编译器“开心”。

请 **完全复制** 下面的代码。我添加了一行特殊的 `#pragma` 指令，强制 Visual Studio 将其作为控制台程序链接，这样就不用你去修改复杂的项目属性了。

```

cpp
// ****
// 修复核心配置问题
// ****
#pragma comment(linker, "/SUBSYSTEM:CONSOLE") // <--- 关键修复：强制告诉链接器这是一个控制台程序，解决 LNK2019

#define _CRT_SECURE_NO_WARNINGS
#define NOMINMAX

#include <iostream>
#include <string>
#include <vector>
#include <queue>
#include <mutex>
#include <condition_variable>
#include <thread>
#include <functional>
#include <fstream>
#include <memory>
#include <chrono>
#include <ctime>
#include <iomanip>
#include <sstream>
#include <random>
#include <windows.h>

using namespace std;

// 1. 日志模块
class LogWriter {
private:
    ofstream logfile;
    mutex logMutex;
    LogWriter() {
        logfile.open("scheduler.log", ios::app);
        if (!logfile.is_open())
            logfile << "---- Log System Started ---\n";
    }
};

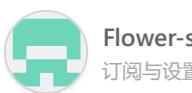
public:
    ~LogWriter() {
        if (logfile.is_open())
            logfile << "---- Log System Closed ---\n";
    }
};

```

追问当前话题

Nanobanana Pro is free

TRIAL



订阅与设置

◇ 常规模式 ◈ GPT 4.1 ^

x1



