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。

使用指南：

1. **复制**：将上述代码完整复制到 Visual Studio 的 `main.cpp` 或 `Source.cpp` 中。
2. **编译**：直接点击 Visual Studio 的 **"Local Windows Debugger" (运行)** 按钮。
   a. 代码开头已经包含了 `#pragma comment(linker, "/SUBSYSTEM:WINDOWS")`，这将自动告诉链接器这是个窗口程序，从而解决 `WinMain` 报错问题。
3. **运行效果**：
   a. 你会看到一个标题为 "Project 3: Lightweight Task Scheduler (GUI)" 的窗口。
   b. 左侧有 5 个按钮，对应 5 个任务。
   c. 右侧是日志显示区域。
4. **功能验证**：
   a. 点击 **Task A**：日志会显示"Scheduled...", 2秒后显示"Backup Complete"，并在你的项目文件夹下生成 `backup_data.zip`。
   b. 点击 **Task B**：每隔 5 秒会自动执行一次矩阵计算，日志会不断刷新计算耗时。
   c. 点击 **Task D**：会显示弹窗提醒（注意：为了不刷屏，我将代码中的提醒周期设为了10秒，你可以修改）。
   d. 日志窗口会实时滚动显示所有后台线程的操作。

这是一个工业级作业的完整实现，包含了 UI 线程分离、异常处理和设计模式应用。

Gemini 3 Pro Preview 12-10 15:01

根据图中要求优化代码，保证完整符合要求。
要有任务撤销功能

我 12-10 15:09

Reasoning Process >

这是一个完全符合你图片要求（项目3）的 C++ 高级实现。

代码亮点与改进：

1. **完整的任务撤销 (Revocation) 功能**：新增了一个"待执行任务列表" (Pending Tasks List)，你可以选中任意一个正在等待或周期循环的任务，点击"撤销 (Revoke)"将其移除。
2. **严格符合图片参数**：
   a. **Task A**: 模拟 `C:\Data` 到 `D:\Backup` 的逻辑，文件名带日期 `YYYYMMDD`。
   b. **Task B**: 严格的 200x200 矩阵乘法，5秒周期。
   c. **Task C**: 调用 `curl` 访问 GitHub API。
   d. **Task D**: 1分钟周期的弹窗（代码中为了演示方便设为 10秒，可手动改回 60秒）。
   e. **Task E**: 延迟 10秒，1000个随机数统计。
3. **设计模式完全落地**：
   a. **Factory**: `TaskFactory` 创建任务。
   b. **Command**: `ScheduledTask` 封装了任务、时间和状态，支持撤销。
   c. **Strategy**: `ITask` 及其子类。
   d. **Observer**: UI 实时监听调度器状态（日志和任务列表更新）。

完整代码 (main.cpp)

请直接复制覆盖原有文件，使用 Visual Studio 运行。

cpp

```cpp
/*
    项目名称：轻量级多任务调度器 (高级版 - 带撤销功能)
    对应需求：项目3 - 目标与三层结构、任务示例 A-E
*/

#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 <commctrl.h> // for ListBox
#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>
#include <filesystem>

// 使用 C++17 文件系统创建目录
namespace fs = std::filesystem;
using namespace std;

// ==========================================
// 全局定义与消息
// ==========================================
#define WM_UPDATE_LOG (WM_USER + 1)
#define WM_UPDATE_LIST (WM_USER + 2)

// UI 控件 ID
enum {
    ID_BTN_A = 101, ID_BTN_B, ID_BTN_C, ID_BTN_D, ID_BTN_E,
    ID_BTN_REVOKE,      // 撤销按钮
    ID_EDIT_LOG,        // 日志框
    ID_LIST_TASKS       // 待执行任务列表框
};

HWND hGlobalWnd = NULL; // 主窗口句柄

// ==========================================
// 1. 日志模块 (LogWriter - RAII)
// ==========================================
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;
        }
    }
};

// 辅助：发送日志到 UI
void LogToUI(const string& msg) {
    LogWriter::Instance().Write(msg); // 同时写入文件
    string* pMsg = new string(msg);
    if (hGlobalWnd) PostMessage(hGlobalWnd, WM_UPDATE_LOG, 0, (LPARAM)pMsg);
}

// ==========================================
// 2. 任务策略 (Strategy Pattern)
// ==========================================
class ITask {
public:
    virtual string GetName() const = 0;
    virtual void Execute() = 0;
    virtual ~ITask() = default;
};

// --- Task A: 文件备份 (C:\Data -> D:\Backup) ---
class TaskBackup : public ITask {
public:
    string GetName() const override { return "Task A: File Backup"; }
    void Execute() override {
        // 获取当前日期
        auto now = chrono::system_clock::to_time_t(chrono::system_clock::now());
        struct tm t; localtime_s(&t, &now);
        stringstream dateSS; dateSS << put_time(&t, "%Y%m%d");

        LogToUI("A: Compressing C:\\Data to D:\\Backup\\backup_" + dateSS.s

        // 为了保证代码在你电脑上能跑，我们在本地模拟这目录
        fs::create_directories("Simulate_C_Data");
        fs::create directories("Simulate_D_Backup");

                                _Data/important.txt"); f << "Data content"; f
```

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