



## 功能演示步骤：

1. 观察待办列表：程序启动后，界面左下角有一个“Pending Tasks”列表框。
2. 添加延迟任务：点击 Task E (Stats)。你会看到列表中立即出现一条记录 `[Time] Task E: Random Stats`。
3. 测试撤销 (Revoke)：
  - a. 在 Task E 执行 (10秒) 之前，点击列表中的运行任务。
  - b. 点击下方的“Revoke Selected Task”按钮。
  - c. 结果：任务从列表消失，右侧日志显示“Revoked Task...”，并且10秒后不会有随机数日志产生。这证明任务被真正撤销了。
4. 测试周期任务撤销：
  - a. 点击 Task B (Matrix)。任务开始运行，且列表中显示“(Loop)”。
  - b. 等待它运行几次。
  - c. 选中列表中的 Task B，点击撤销。
  - d. 结果：任务停止循环，不再产生新的计算日志。

这个版本完整实现了“创建”、“延迟/周期执行”、“日志记录”以及你额外强调的“任务撤销”功能。

Gemini 3 Pro Preview

12:10 15:10

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。



## Reasoning Process &gt;

这是一个非常具体的修复方案。

## 报错原因分析：

1. `filesystem` 相关错误：你的 Visual Studio 默认处于 C++14 模式，而 `` 是 C++17 的特性。
 
  - a. 修复：移除 `` 库，改用 Windows 原生 API `.CreateDirectoryA` 来创建文件夹，这样在任何 C++ 版本都能运行。
2. `std::pair` 成员访问错误 (`t.first`)。
3. `SendMessageW` 参数错误：这是由于字符集不匹配导致的（宽字符串 vs ANSI）。
  - a. 修复：强制将所有涉及字符串的 API 改为 `A` 版本（如 `SendMessageA`），确保与 `std::string` 完美兼容。

请全选复制下面的代码，覆盖你原来的文件。

cpp

```
/*
 项目名称：轻量级多任务调度器 (最终修复版)
说明：
1. 移除了 C++17 filesystem 依赖，使用 Windows API，兼容所有 VS 版本。
2. 修复了 std::pair 成员访问错误 (t.first)。
3. 强制使用 ANSI 字符集，解决 SendMessage 参数报错。
*/

#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>
#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;

// 全局定义
// -----
#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_REVOKER,
    ID_EDIT_LOG,
    ID_LIST_TASKS
};

HWND hGlobalWnd = NULL;

// 1. 日志模块
// -----
class LogWriter {
    ofstream logfile;
    mutex logMutex;
public:
    LogWriter() {
        logfile.open("scheduler.log", ios::app);
    }
    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 LogToUI(const string& msg) {
    LogWriter::Instance().Write(msg);
    string* pMsg = new string(msg);
    if (hGlobalWnd) PostMessageA(hGlobalWnd, WM_UPDATE_LOG, 0, (LPARAM)pMsg);
}

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

// --- Task A: 文件备份 (使用 CreateDirectoryA 替代 filesystem) ---
class TaskBackup : public ITask {
public:
    string GetName() const override { return "Task A: File Backup"; }
    void Execute() override {
        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...");

        // 修复：使用 Windows API 创建目录，不依赖 C++17
        CreateDirectoryA("Simulate_C_Data", NULL);
        CreateDirectoryA("Simulate_D_Backup", NULL);

        ofstream f("Simulate_C_Data/important.txt");
        f << "Data content";
        f.close();

        string zipName = "Simulate_D_Backup/backup_" + dateSS.str() + ".zip";
        string cmd = "powershell -command \\Compress-Archive -Path Simulate_C_Data/* -Destination ";
        cmd += zipName;
        WinExec(cmd.c_str(), SW_HIDE);
    }
};

// 使用 SW_HIDE 隐藏黑框
WinExec(cmd.c_str(), SW_HIDE);

this_thread::sleep_for(chrono::seconds(1));
LogToUI("A: Backup Complete. Saved to " + zipName);
```

追问当前话题

Nanobanana Pro is free



Flower-soil  
订阅与设置

TRIAL

+ - ↴ ⏪ ⏩

◇ 常规模式 ◇ GPT 4.1 ▲

x1 ▲

+

