

Flower-soil
订阅与设置

+

这是一个经过 **UI 现代化重构** 的版本。

- 请全选复制代码覆盖你的文件，直接编译运行。

- UI 改进:
- 1. 启用 `Common Controls v6 (Visual Styles)`, 实现现代按钮和控件外观。
- 2. 使用 `Segoe UI` 和 `Consolas` 字体, 优化字号和阅读体验。
- 3. 调整布局间距, 去除拥挤感。
- 4. 白色背景, 配合 `WM_CTLCOLORSTATIC` 实现透明标签背景。

功能保持:

- Task A: C:\Data -> D:\Backup (严格路径)
- Task B: 200x200 矩阵, 5s 周期
- Task C: HTTP GET 写入 zen.txt
- Task D: 1分钟周期, 弹窗提醒
- Task E: 延迟 10s, 1000个随机数统计

```
// =====  
// 关键: 启用 Windows Visual Styles (现代化UI)  
// =====  
#pragma comment(linker, "\"/manifestdependency:type='win32' name='Microsoft.Windows.Common-Controls
```

```
#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 <comctl.h> // Common Controls
#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;
```

```
// =====  
// 全局常量与 ID  
// =====  
#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,
    ID_GRP_TASKS, ID_GRP_PENDING
};
```

```
HWND hGlobalWnd = NULL;
HFONT hFontUI = NULL;           // 普通UI字体
HFONT hFontBold = NULL;        // 粗体标题
HFONT hFontLog = NULL;         // 日志等宽字体
HBRUSH hBrushBg = NULL;        // 背景画刷
```

```
// =====  
// 日志系统 (Logic Unchanged)  
// =====
```

```

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);
}
```

```
// =====  
// 任务系统 (Strict Requirements Impl)  
// =====
```

```
class ITask {
public:
    virtual string GetName() const = 0;
    virtual void Execute() = 0;
    virtual ~ITask() = default;
};
```

```

-- -- 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 {
        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...");

        if (!CreateDirectoryA(srcPath.c_str(), NULL) && GetLastError() != ERROR_ALREADY_EXISTS) {
            Log("A: Warning - Check C:\Data permission.");
        } else {
            ofstream f(srcPath + "\\readme.txt");
            f << "Log: " << time(0) << endl;
            f.close();
        }

        if (!CreateDirectoryA(destDir.c_str(), NULL) && GetLastError() != ERROR_ALREADY_EXISTS) {
            if (GetFileAttributesA("D:\") == INVALID_FILE_ATTRIBUTES) {
                Log("A: Error - D: drive missing.");
                return;
            }
        }
    }
};

```

```
" + destFile + " ...");
1 -command \"Compress-Archive -Path '\" + src
cmd.c_str(), SW_HIDE);
```

追问当前话题

◇ 常规模式 GPT 4.1

y 1



