逆向上机作业1：Windows系统编程实验

程序1

实验任务

该程序能够创建一个本机的 OllyDbg 进程（使用 CreateProcess，显式指定 OllyDbg 可执行文件的路径）

实验过程

查阅 CreateProcess的文档

官方文档给出的示例使用 CreateProcess 创建进程，路径由 `argv[1]` 指定。

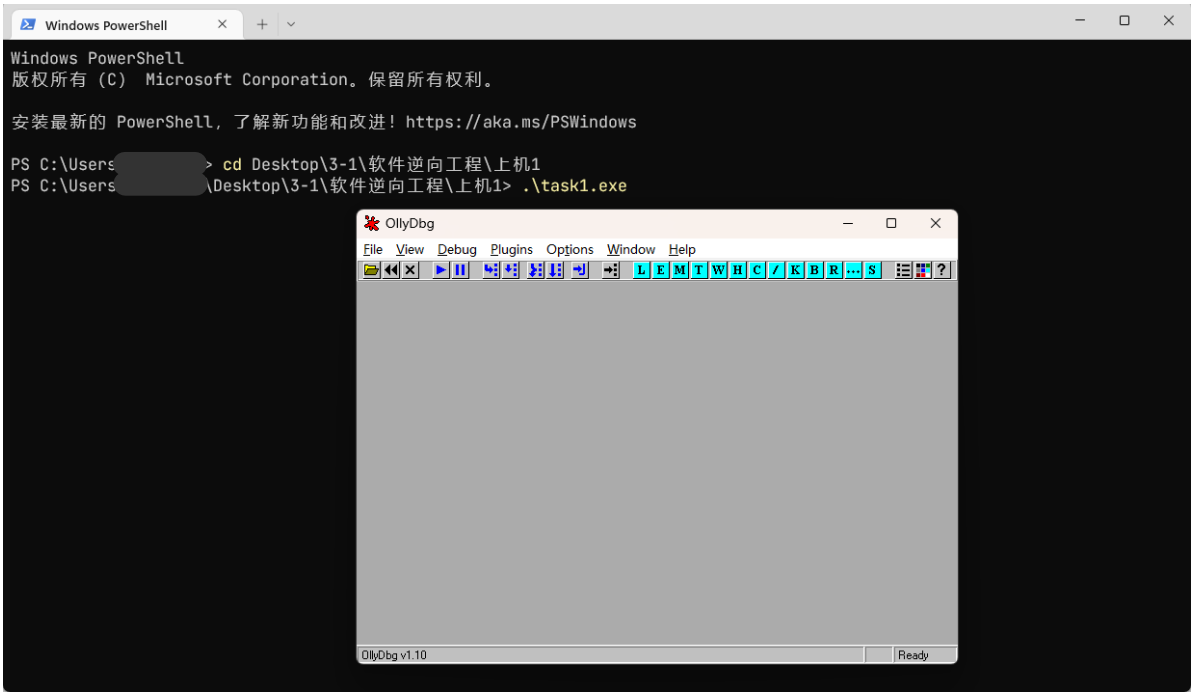
本题中需要显式指定 OllyDbg 可执行文件的路径，所以将 `argv[1]` 替代为本机的 OllyDbg 路径，再删去对 argv 参数个数的检测即可。

得到如下代码：

```
#include <stdio.h>
#include <windows.h>

void main()
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    if (!CreateProcess(NULL, // No module name (use command line)
        "C:\\Users\\Qian Liu\\Desktop\\3-1\\软件逆向工程
\\odbg110\\OLLYDBG.EXE",
        // Command line
            NULL,  // Process handle not inheritable
            NULL,  // Thread handle not inheritable
            FALSE, // Set handle inheritance to FALSE
            0,     // No creation flags
            NULL,  // Use parent's environment block
            NULL,  // Use parent's starting directory
            &si,   // Pointer to STARTUPINFO structure
            &pi))  // Pointer to PROCESS_INFORMATION structure
    {
        printf("CreateProcess failed (%d).\n", GetLastError());
        return;
    }
    // Wait until child process exits.
    WaitForSingleObject(pi.hProcess, INFINITE);

    // Close process and thread handles.
```

```
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
```

编译运行，测试成功



程序2

实验任务

- 该程序能够创建一个线程，显示 MessageBox
- 在以上子线程中，编程获得 kernel32.dll 在当前系统中的路径信息，作为内容显示在以上的 MessageBox 中
- 在以上子线程中，编程获得子线程所加载 kernel32.dll 中的 GetCurrentThreadId() 函数的地址，调用该函数，获得子线程的线程编号，将线程编号连接到上一问的 kernel32.dll 路径后面，再将连接结果字符串显示在 MessageBox 中

实验过程

创建线程

使用 CreateThread，代码：

```
handle = CreateThread(NULL, 0, ThreadFun, &t, 0, NULL);
```

显示 MessageBox

使用 MessageBox，代码：

```
MessageBox(NULL, ker_path, TEXT("Win_prog"),
MB_ICONINFORMATION);
```

获得 kernel32.dll 路径

使用 `HMODULE h_kernel32 = GetModuleHandle(szKernel32);` 获得 kernel32.dll 的模块句柄，然后使用 `GetModuleFileName(h_kernel32, ker_path, 50);` 获取 kernel32.dll 路径，写入 ker_path 变量。

获得子线程的 GetCurrentThreadId() 函数地址

使用 GetProcAddress，代码：`GetCurrentThreadId_addrs = GetProcAddress(h_kernel32, "GetCurrentThreadId");` 调用该函数：`sub_id = GetCurrentThreadId_addrs();`

完整代码

```
#include <process.h>
#include <stdio.h>
#include <string.h>
#include <strsafe.h>
#include <tchar.h>
#include <windows.h>

HANDLE handle;

DWORD WINAPI ThreadFun(LPVOID pM)
{
    int sub_id;
    char ker_path[100];
    char sub_id_char[20];
    LPTSTR szKernel32 = TEXT("kernel32.dll");

    // sub_id = GetCurrentThreadId(); //子线程的线程ID
    typedef long long int(WINAPI * PGETID)();
    PGETID GetCurrentThreadId_addrs;

    HMODULE h_kernel32 = GetModuleHandle(szKernel32);
    GetModuleFileName(h_kernel32, ker_path, 50);

    GetCurrentThreadId_addrs = GetProcAddress(h_kernel32, "GetCurrentThreadId");
    sub_id = GetCurrentThreadId_addrs();
    itoa(sub_id, sub_id_char, 10);

    strncat(ker_path, ": id = ", 10);
    strncat(ker_path, sub_id_char, 10);
    MessageBox(NULL, ker_path, TEXT("Win_prog"), MB_ICONINFORMATION);

    return 0;
}

int main()
{
    int t = 0;
    handle = CreateThread(NULL, 0, ThreadFun, &t, 0, NULL);

    WaitForSingleObject(handle, INFINITE);
}
```

编译运行，测试成功