插件

- 让自己的应用程序支持更多的功能

```
ON_COMMAND_RANGE(id1, id2, func)
id1 最小值 id2 最大值 func 函数指针
id1 id2 之间的消息都会调用 func 函数
```

加载插件的方法：

1. 定义导出函数

```
void PInit();          // 初始化
void PUninit();        // 反初始化
const char* PName();   // 插件名称
void PClick();         // 插件点击事件
```

1. 遍历文件夹，加载DLL，获取导出函数，判断是否是插件

```
    // 添加菜单
    CMenu* pmnPlugin = new CMenu;
    pmnPlugin->CreatePopupMenu();
    CMenu* pmnMain = GetMenu();
    pmnMain->AppendMenu(MF_POPUP, (UINT_PTR)pmnPlugin->m_hMenu, "插件");

    int nIDIdx = 0;
    // 遍历文件夹
    CFileFind finder;
    BOOL bWorking = finder.FindFile("plugin\\*.dll");
    while (bWorking)
    {
        bWorking = finder.FindNextFile();

        // 加载DLL，获取导出函数地址，判断是否是插件
        HMODULE hmPlugin = ::LoadLibrary(finder.GetFilePath());
        if (hmPlugin == nullptr)
        {
            continue;
        }
        PluginInterface pi;
        pi.m_pfnPInit = (PInit_type)::GetProcAddress(hmPlugin, "PInit");
        pi.m_pfnPUninit = (PUninit_type)::GetProcAddress(hmPlugin, "PUninit");
        pi.m_pfnPClick = (PClick_type)::GetProcAddress(hmPlugin, "PClick");
        pi.m_pfnPName = (PName_type)::GetProcAddress(hmPlugin, "PName");
        if (pi.m_pfnPInit == nullptr
            || pi.m_pfnPUninit == nullptr
```

```
            || pi.m_pfnPName == nullptr
            || pi.m_pfnPClick == nullptr)
        {
            ::FreeLibrary(hmPlugin);
            continue;
        }
        m_mpPlugin[PLUGIN_ID + nIDIdx] = pi;
        // 添加菜单
        pi.m_pfnPInit();
        pmnPlugin->AppendMenu(MF_STRING, PLUGIN_ID + nIDIdx++, pi.m_pfnPName());
    }
```

2. 调用导出函数

```
void CMainFrame::OnFileMenuItems(UINT nID)
{
    m_mpPlugin[nID].m_pfnPClick;
}
```

实现一个插件的步骤：

1. 实现应用程序导出函数

```
void PInit();          // 初始化
void PUninit();        // 反初始化
const char* PName();   // 插件名称
void PClick();         // 插件点击事件
```