基类 c_cmd_target 只是提供了一个框架，框架中提供了建立消息地图的函数 load_cmd_msg() 以及搜索消息的函数 find_msg_entry()。完整的机制建立，是在其派生类 c_wnd 中实现的。

GuiLite 的窗口实例（即 c_wnd 类实例）在运行的时候，通过链表的方式联系在一起。所有的消息（包括了鼠标点击消息、键盘消息等）都可在此链表中溯源，找到消息相关窗口，以此找到相应的处理函数。

c_wnd 类的成员变量中，包含了父窗口、第一个子窗口、前兄弟窗口和后兄弟窗口，具体如下所示。

```
c_wnd*      m_parent;         //父窗口
c_wnd*      m_top_child;      //第一个子窗口
c_wnd*      m_prev_sibling;   //前兄弟窗口
c_wnd*      m_next_sibling;   //后兄弟窗口
```

GuiLite 的所有控件都是派生自 c_wnd 类的，通过这些指针成员变量相连，构建一个巨大的链表。与此链表相关的另一关键数据结构为 WND_TREE，它包含了 c_wnd 类实例的各种属性，其原型如代码清单 6-18 所示。

**代码清单 6-18　WND_TREE 结构体原型**

```
typedef struct struct_wnd_tree
{
    c_wnd*          p_wnd;              //c_wnd指针实例
    unsigned int    resource_id;        //资源ID
    const char*     str;                //窗口字符串
    short           x;                  //窗口位置
    short           y;
    short           width;              //窗口宽度
    short           height;             //窗口高度
    struct struct_wnd_tree* p_child_tree;  //子窗口的链表指针
}WND_TREE;
```

建立链表的工作，是由 c_wnd 的成员函数 connect() 实现的。这是一个虚函数，c_wnd 的派生类可以根据自己的需求重载它。使用者可以构建 WND_TREE 的数组，将需要显示的控件填充在数组中，并规定各控件的位置、大小等属性，再使用 connet() 函数将这些控件连接起来，如示例 6-11 所示（代码来自示例工程 HelloGuiLite）。

【示例 6-11】构建控件。

```
static WND_TREE s_desktop_children[] =
{
    {(c_wnd*)&s_start_menu, ID_START_MENU, 0, 400, 50, 475, 633, NULL},
    {(c_wnd*)&s_start_button, ID_START_BUTTON, 0, 0, 682, 47, 38, NULL},
    {(c_wnd*)&s_start_button1, ID_START_BUTTON, 0, 0, 400, 47, 38, NULL },
    { NULL,0,0,0,0,0,0 }
```