基于 MFC 的插件应用程序设计与实现

丰晋军　　许铁山

（江南计算技术研究所，江苏　　无锡　　214083 ）

摘　要　讨论了插件应用系统的基础理论及应用优势，　　据此设计了基本的插件应用系统框架模块并通过　　MFC 基础平台予以具体实现。

关键字　插件；动态链接库；　　MFC

1　插件体系结构

软件开发手段的演化，就在于以最小的代价得到更健壮且易于扩展和维护的“好”的应用系统，开发工具的持续改进和开发思想的进化使得我们有可能实现上述目标。

从面向过程的开发至面向对象的编程，直至目前面向组件的开发，正是上述思维的展现。基于插件的应用系统从体系结构设计出发，着力构建低耦合的，灵活可扩展的且支持无编译热插拔的应用系统，通过分析应用需求，提炼功能相似的模块并设计相应的模块间接口，我们就可以将该部分功能分离出来，综合来看，基于插件的应用系统有以下优势：

(1) 实现真正意义上的软件组件的“即插即用”　。

(2) 在二进制级上集成软件　减少大量的软件重新编译与发布所带来的麻烦。

(3) 能够很好地实现软件模块的分工开发　　并且能够大量吸取他人的优点。

(4) 可以较好地实现代码隐藏　保护知识产权。

基于插件的体系结构如图　1 所示。

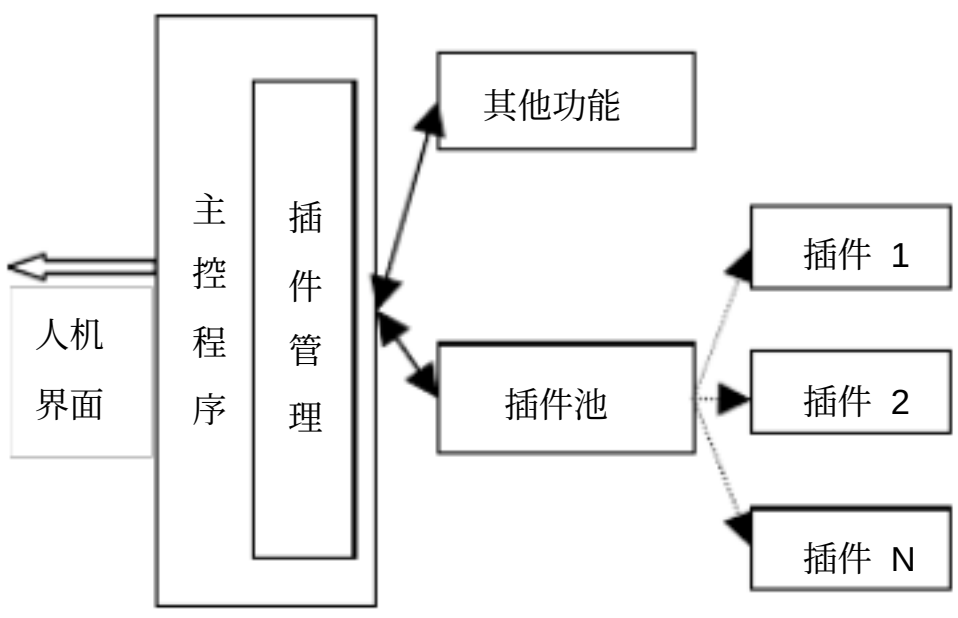

图 1　插件应用系统体系结构

从图　1 可知，一个完成的基于插件的应用系统共分两部分：插件主控程序（含插件调度核心模块）以及插件池（存放所有插件）。主控程序通过核心模块提供插件管理功能，　　主要包括：

（1）注册及卸载插件：　插件在使用之前必须注册或存放于特定的路径中，主控程序根据相关配置参数对插件完成初始化工作。

（2）启用及禁用插件：主控程序可以根据用户指令，对某些已装载的插件予以启用或禁用。

（3）显示插件信息：包括插件描述，开发者信息，版本和版权声明等。

（4）配置插件参数：　插件本身的运行需要对某些参数进行定制。

根据模块规划，　插件实现特定的功能并将接口暴露出来，根据需要，可能还需要调用主控程序提供的方法以操作资源或数据。

2　设计思路及　MFC 实现

据上述讨论，我们设计一个基本的插件应用系统框架，其中主控程序是基于　MFC 对话框的应用程序，　插件使用动态库实现，插件管理部分使用专门的　　CPluginManager 类实现，其实现的函数如图　2 所示。

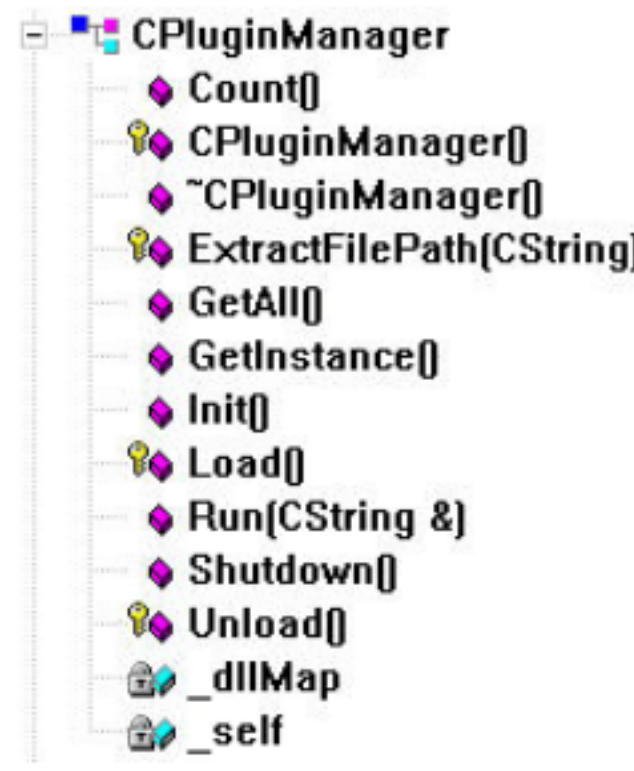

图 2　插件管理类视图

就主要函数说明如表　1 所示。

表 1 插件管理类提供的函数

函数名称	函数说明
Init	初始化，搜索所有有效插件
Shutdown	释放资源
GetAll	得到所有插件名，用 CString 对象返回，名之间用：隔开
Count	返回可用插件的个数
Run	调用插件提供的函数接口
Load	载入所有插件
UnLoad	卸载所有插件
ExtractFilePath	提取插件存放路径

以 Load 函数为例，我们使用 STL 的 MAP 数据结构存放插件句柄和插件的对应，代码如下：

```
void CPluginManager::Load()
{
    // ……相关变量定义省略
    GetModuleFileName(AfxGetApp()->m_hInstance,filepath,
MAX_PATH-1);
    SetCurrentDirectory(ExtractFilePath(filepath));
    CFileFind finder;
    CString strWildCard = _T("*.plx");
    BOOL bWorking = finder.FindFile(strWildCard);
    while (bWorking)
    {
        bWorking = finder.FindNextFile();
        if (finder.IsDots() || finder.IsDirectory())
            continue;
        HMODULE hm = LoadLibrary(finder.GetFilePath());
        if ( !hm )
        {// ……载入失败处理代码 }
        else if(NULL == GetProcAddress(hm , "PluginMain"))
        {// ……非可用插件处理代码 }
        else
    _dllMap.insert(make_pair(finder.GetFileName() , hm));
    }
}
```

插件提供的接口函数如下（仅作为示例，如需其它接口，可照此添加）：

```
#ifdef PLUG1_EXPORTS
#define PLUG_API __declspec(dllexport)
#else
#define PLUG_API __declspec(dllimport)
#endif

PLUG_API void PluginMain(void)
{
    ::MessageBox(NULL , "插件 1 测试成功!" ,
"Plugin1" , MB_OK);
}
```

主控程序使用树控件展示插件功能，程序初始化时首先调用初始化函数完成控件注册，然后得到所有控件的名称并以叶节点的形式显示出来，用户双击相应的叶节点时，主控程序调用插件提供的函数 PluginMain ，调用过程如下：

```
void CTreeCtrlDlg::OnDblClkTree(NMHDR* pNMHDR,
LRESULT* pResult)
{
    m_hTreeItem = m_wndTree.GetSelectedItem();
    CString plName=
    m_wndTree.GetItemText(m_hTreeItem);
    g_PluginManager->Run(plName);
    *pResult = 0;
}
```

主控程序运行时及双击树形图叶节点时界面，如图 3所示。

图 3 主控程序运行界面

主控程序所在路径下建立 plugin 目录，并存放三个插件文件。

3 结束语

开放的组件化体系结构模块组成清晰，同时也方便了系统扩展和后续维护，本文从插件应用系统的体系结构入手，设计了主控模块与插件池的功能要求，给出了插件管理类实现的功能，并基于 MFC 给出了相应的具体实现，需要说明的是：

（1）插件应用系统涉及到主控程序和插件之间的双向交互，插件也可以利用主控程序提供的接口访问公共资源和数据，本文对该部分功能未作实现。

（2）真正的应用系统由于插件众多，不可避免的涉及到插件之间的协同和冲突检测，需要在结构设计上解决。

（3）插件的实现方式较多，可以根据具体的应用需求和系统规模选用比较合适的一种。

总的来看，基于组件的插件应用系统，由于存在诸多优势，必将带来更多的应用前景和用户体验。

参考文献

[1] 彭永康，章义来，插件及其接口的研究与应用，计算机应用，2003，6：122~123

[2] 于珊珊，软件插件技术及其应用研究，电脑学习，2007，8：55~56

收稿日期：5 月 22 日 修改日期：6 月 2 日

作者简介：丰晋军（1980-），男，工程师，主要研究方向信息安全、数据库管理。