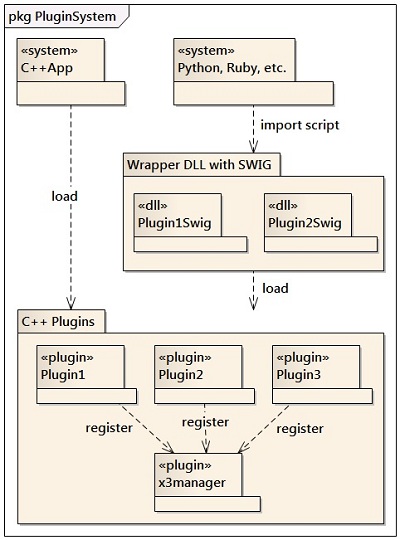
## 一、插件工作原理（x3py）



插件系统主要有两种使用形式，一是在C++的主程序或动态库中使用C++插件，另一种是在Python等其他编程语言中使用C++插件。

1、插件管理器（x3manager）

插件管理器（x3manager）也是一个插件，按照统一标准实现了插件的导出函数。x3manager插件的主要作用是在多个插件之间进行中介联系，从而允许插件之间能够互相访问而不直接发生联系，各个插件动态库加载后自动向x3manager登记。例如要创建一个对象，通过插件管理器来查询是在哪个插件中实现了该类，从而创建出对象、访问接口功能。

相对于之前的 x3c插件框架，x3py插件框架中的插件管理器不负责插件动态库的加载管理，只保留中介作用，由应用程序等外部模块来加载插件动态库文件。由于提供了多种加载方法，这样更灵活、职责更清晰。

2、C++程序使用C++插件

在C++的主程序或动态库中使用C++插件时，先加载各个插件动态库，然后就可使用插件接口。插件加载方式的详细说明见 如何加载插件

2.1、同时使用多个插件的简单加载方法

#include <portability/x3port.h> // 相当于#include<windows.h>

#include <nonplugin/useplugins.h> // 包含辅助加载类，一个工程内只能包含一次

void yourFunction()

{

// 多个插件文件名，以NULL结尾

const char\* plugins[] = {

"x3manager.pln", "plsimple.pln", "observerex.pln", NULL

};

// 自动加载和卸载插件，插件在程序文件的plugins子目录下

x3::AutoLoadPlugins autoload(plugins, "plugins");

// 可以使用接口了。。。

}

2.2、仅使用一个插件的简单加载方法

#include <portability/x3port.h> // 相当于#include<windows.h>

// 插件在程序文件的plugins子目录下，如果不定义则插件与程序同目录

// 注意末尾有目录分隔符，在包含useplugin.h前定义，如果不定义则为默认的空串

#define PLUGIN\_PATH "plugins/"

// 要使用的插件文件名，没有目录和后缀名(.pln)

#define PLUGIN\_NAME "plsimple"

// 包含辅助加载类，一个工程内只能包含一次

#include <nonplugin/useplugin.h>

void yourFunction()

{

// 可以使用接口了。。。

}

2.3、遍历目录下的所有插件

#include <nonplugin/scanplugins.h>

void yourFunction()

{

// 默认遍历 plugins 目录，加载其中的插件文件

x3::loadPlugins();

// 可以使用接口了

test();

// 卸载所有插件，后加载的先卸载

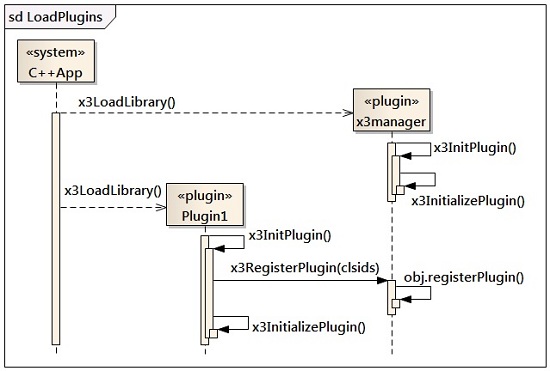
x3::unloadPlugins();

}

**二、插件调用过程**

在插件实现时，通过包含 [pluginimpl.h](https://github.com/rhcad/x3py/blob/master/interface/core/module/pluginimpl.h) 文件，自动实现了 x3InternalCreate、x3InitPlugin、x3FreePlugin、x3CreateObject 这四个导出函数，通过导出函数就可在插件之间建立联系。下面将说明这四个导出函数之间的联系。

**1、加载插件**



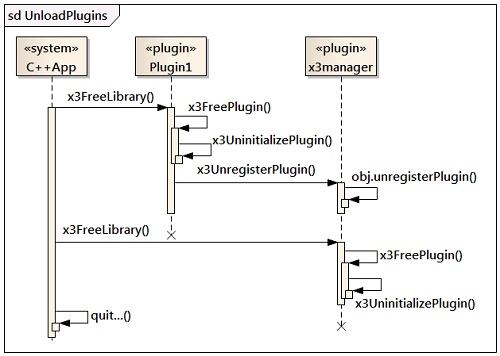
使用 x3LoadLibrary 函数加载动态库文件。使用此函数而不是WinAPI的 LoadLibraryA 函数，是为了确保在VC++、MinGW GCC、Linux、Mac 等编译环境下能正常加载动态库文件，且确保自动调用 x3InitPlugin 函数进行插件初始化。

x3InitPlugin 函数用于在插件加载时进行初始化、向管理器登记（x3RegisterPlugin）此插件的所有类ID（CLSID串）。通过在插件CPP文件中包含 [pluginimpl.h](https://github.com/rhcad/x3py/blob/master/interface/core/module/pluginimpl.h) 文件，自动实现了 x3InitPlugin。插件动态库被多次加载时不会多次重复初始化，这是在 x3InitPlugin 函数中通过计数来保证的。

x3Initialize 函数由各个插件自行实现，用来完成附加的初始化操作，由 x3InitPlugin 函数调用。

插件加载方式的详细说明见 [如何加载插件](https://github.com/rhcad/x3py/wiki/LoadPlugin)。

**2、卸载插件**

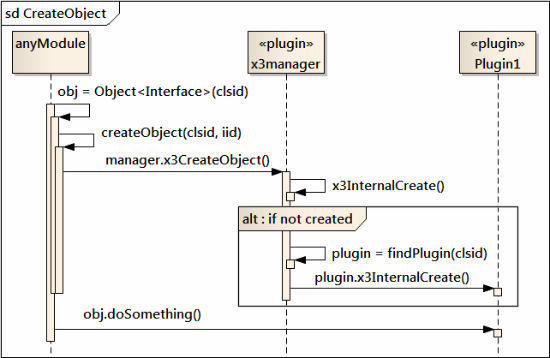


使用 x3FreeLibrary 函数卸载动态库文件。使用此函数而不是WinAPI的 FreeLibrary 函数，是为了确保在VC++、MinGW GCC、Linux、Mac 等编译环境下能正常卸载动态库文件，且确保自动调用 x3FreePlugin 函数进行插件注销。

x3FreePlugin 函数用于在插件卸载时进行对象释放、向管理器注销（x3UnregisterPlugin）此插件的信息。通过在插件CPP文件中包含 [pluginimpl.h](https://github.com/rhcad/x3py/blob/master/interface/core/module/pluginimpl.h) 文件，自动实现了 x3FreePlugin。

x3Uninitialize 函数由各个插件自行实现，用来完成附加的资源释放操作，由 x3FreePlugin 函数调用。

**3、使用插件接口**



通过使用智能指针模板类 [x3::Object](https://github.com/rhcad/x3py/blob/master/interface/core/objptr.h) 来使用插件接口，自动处理引用计数。

x3::Object模板类通过x3::createObject函数去调用 x3CreateObject 导出函数来创建对象。如果插件自身的 x3InternalCreate 不能创建对象，则再转给插件管理器，由后者根据类ID查找对应插件，然后调用对应插件的 x3InternalCreate 函数。