# DLL

## 概述

DLL（Dynamic Link Library），又称为“应用程序拓展”。是一个个独立的模块，包含资源文件、函数等，供应用程序需要时调用。现阶段，大部分应用程序都会设计成主程序+DLL的框架。当主程序需要哪个功能的时候，就会去链接并调用对应的DLL，因为DLL是彼此独立的，所以主程序的加载速度更快。这样也提升了工作效率。

同时，由于DLL的独立性，使得程序的维护和更新变得更加容易，成本大大降低。因为程序只需要更新相应的模块，而不是整体的更新。

## 分类

链接库分为动态链接库和静态链接库两种，后缀分别为.dll和.lib。

### 静态链接库

静态链接库，在运行的时候就直接把代码全部加载到程序中，通过如下方式调用：

#include <Psapi.h>

#pragma comment (lib,”Psapi.lib”);

### 动态链接库

动态链接库在需要的时候加载，在不需要的时候就卸载释放资源。

1. 使用LoadLibrary动态加载DLL
2. 使用GetProAddress获取DLL中导出函数的指针
3. 最后使用FreeLibrary卸载指定DLL

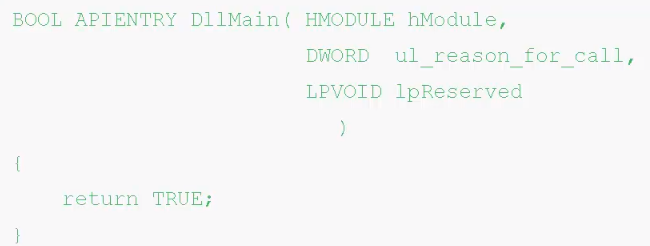
在我们的Visual Studio的编译环境下，DLL又分为三类：

1. 非MFC的DLL：即使用SDK API进行编程，能被其他所有语言调用；
2. MFC规则DLL：可以使用MFC进行编程，能被其他所有语言调用；
3. MFC扩展DLL：可以使用MFC进行编程，但只能被用MFC编写的程序调用

注：第二种使用MFC框架开发，可以使用封装类，第三种不常用。

## 入口点与参数

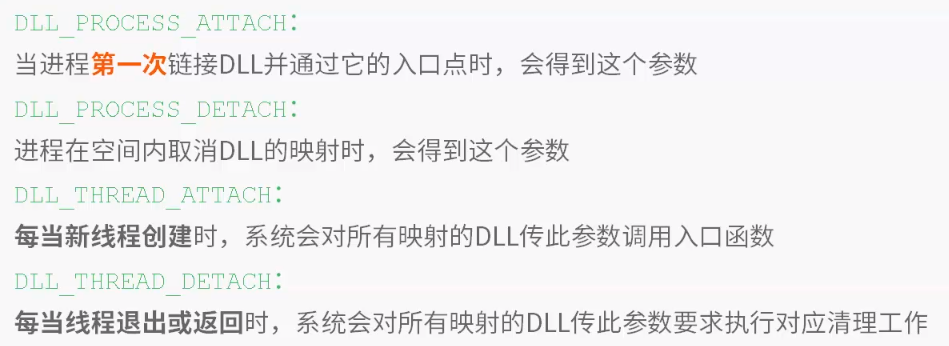
DLL文件的入口点函数（相当于main函数）：



参数：

HModule：指向DLL本身的实例句柄

ul\_reson\_for\_call：调用DLL的原因，取值包括：



lpReserved：保留，未使用

## 编写与导出

DLL的导出函数使用：

extern “c” \_declspec(dllexport)

导入函数使用：

extern “c” \_declspec(dllimport)

extern “c”作为一种编译约定，要求编译器将它所声明的函数按照C语言的方式被编译导出，如果不加这个符号，则导出的函数名可能会被改写。

## DLL动态加载



# 线程注入

## 概述

线程注入，是通过开启远程线程的方式，将DLL加载到目标宿主进程中的常用方式。目前，远程线程注入，作为WinNT平台下最常用的DLL注入方式之一，在游戏、安全等领域扮演者极为重要的角色。

由于WinNT系统下进程空间的独立性，获取其他进程的信息，就需要通过进入目标进程空间的方式，而使用线程注入就可以轻易地实现目的。但是，滥用线程注入，或者是注入一个具有编写BUG的DLL，很容易对宿主进程的稳定性造成破坏，进而导致进程崩溃。

## 注入可行性

Kernel32.dll和user32.dll是两个在大部分程序上都会调用的DLL。

同一个DLL，在不同的进程中，不一定被映射（加载）在同一个内存地址下。但是kernel32.dll和user332.dll例外，它们总是被映射到进程的内存首选地址。因此，在所有使用这两个DLL的进程中，这两个DLL的内存地址是相同的。

目标进程🡪传入DLL地址🡪开启远程线程🡪加载DLL🡪实现DLL的注入

## 实现

依次使用如下函数实现线程注入：

