PinTools的插桩粒度可以分为：image（我认为可以理解为输入文件），section（类似于基本块，但PinTools中定义的基本块与平常我们所说的基本块不太一样，不过目前这个不同没有影响到我写程序），routines（函数或方法，function），instruction（语句）

PinTools编程有一定模式，大致可以分为三个部分Analysis routines, Instrument routines和Main

以[ManualExamples/malloctrace.cpp](https://software.intel.com/sites/landingpage/pintool/docs/81205/Pin/html/index.html" \l "FunctionArguments" \t "http://www.mamicode.com/_blank)为例，Analysis routines是你想在插入点做什么，比如那个MallocAfter（ADDRINT ret）会被放在malloc函数返回后，作用是打印malloc函数的返回值；Instrument routines是对插桩函数的描述和固定，经常用到RTN\_InsertCall、INS\_InsertCall等等，参数是插入点、插入点前后、插入的函数、插入函数的参数列表，就像构造一个函数原型，你的Analysis routines是什么样子的，参数列表里填对应的数据类型和数据就可以；Main主程序，没仔细研究，感觉模式比较固定，就是一些初始化操作、调用插桩函数、启动工具的命令。

PinTools的插桩粒度可以分为：image（我认为可以理解为输入文件），section（类似于基本块，但PinTools中定义的基本块与平常我们所说的基本块不太一样，不过目前这个不同没有影响到我写程序），routines（函数或方法，function），instruction（语句）

**Pin的三种插入（代码）模式：**

**1．Instruction  level (Ins )**

即在每一条原始程序指令前，后插入自己的代码，即调用这个函数，这里我不详细讲，因为pin的文档很全面，自己去就行。但是有一点这个级别执行可想而知，会执行的很慢，很耗时间。（因为在每天指令上加入代码，即使自己的代码只包含一行(ps: nop)，代码就至少膨胀5倍以上）

**2. Function level(RTN)**

Pin通过符号表信息来找到这些需要插入的位置，要使用这种模式的代码插入，首先得调用pin内置的初始化符号表函数，即PIN\_InitSymbols()

**3. Basic  block level(BBL)**

即基本调用块级别,插入模式，只在trace时候可用。

下图为pin的执行过程：

0. 启动自己（pin.exe），以debugger的模式，附加要分析程序的进程

1. 注入pinvm.dll到宿主程序中

2. 注入自己的工具代码(即pintool.dll)

3. 激活主程序进程

该例子实现了一个统计执行的指令数目的功能。可以看出，一个Pin工具主要包括4个部分：

(1)初始化Pin环境：PIN\_Init

(2)设置用于Instrument的回 调函数：INS\_AddInstrumentFunction添加一个回调函数，该回调函数的执行时机是每执行一条指令

(3)应用结束前工作，与PIN\_Init对应：PIN\_AddFiniFunction

(4)启动Instrument过程