**《软件安全》实验报告**

姓名：胡博浩 学号：2212998 班级：信息安全

**一、实验名称：**

程序插桩及Hook实验

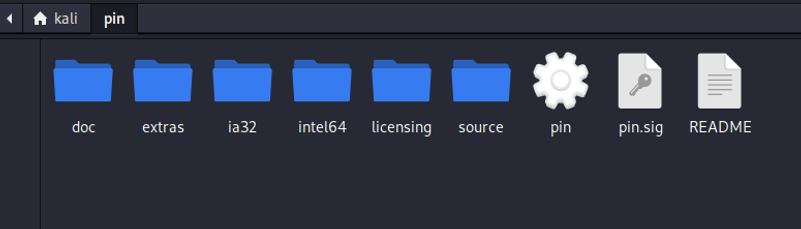
**二、实验要求：**

复现实验一，基于Windows MyPintool或在Kali中复现malloctrace这个PinTool，理解Pin插桩工具的核心步骤和相关API，关注malloc和free函数的输入输出信息。

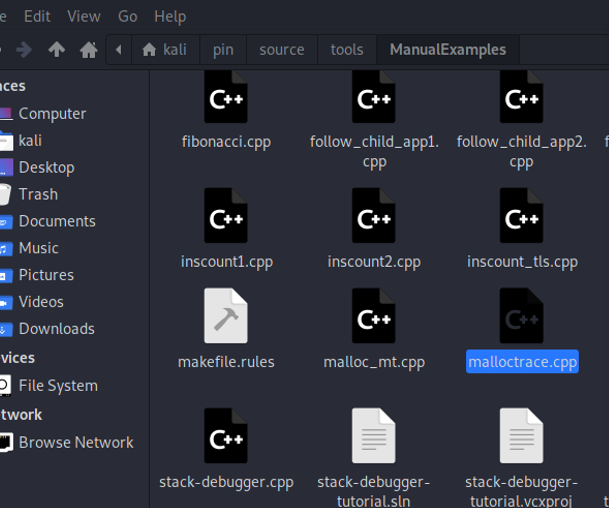
**三、实验过程：**

（一）在Kali中复现malloctrace

1.首先安装Pin3.18：

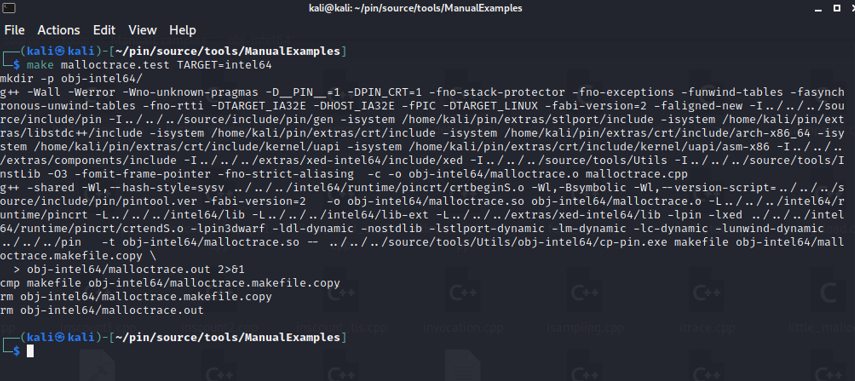


2.进入tools目录中的ManualExamples，找到malloctrace.cpp：

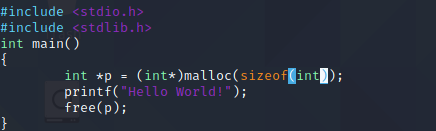


3.编译malloctrace.cpp，产生动态链接库文件：

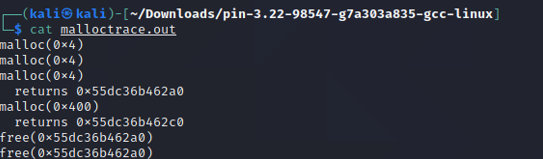
make malloctrace.test TARGET=intel64



4. 对下述代码编译得到可执行程序：



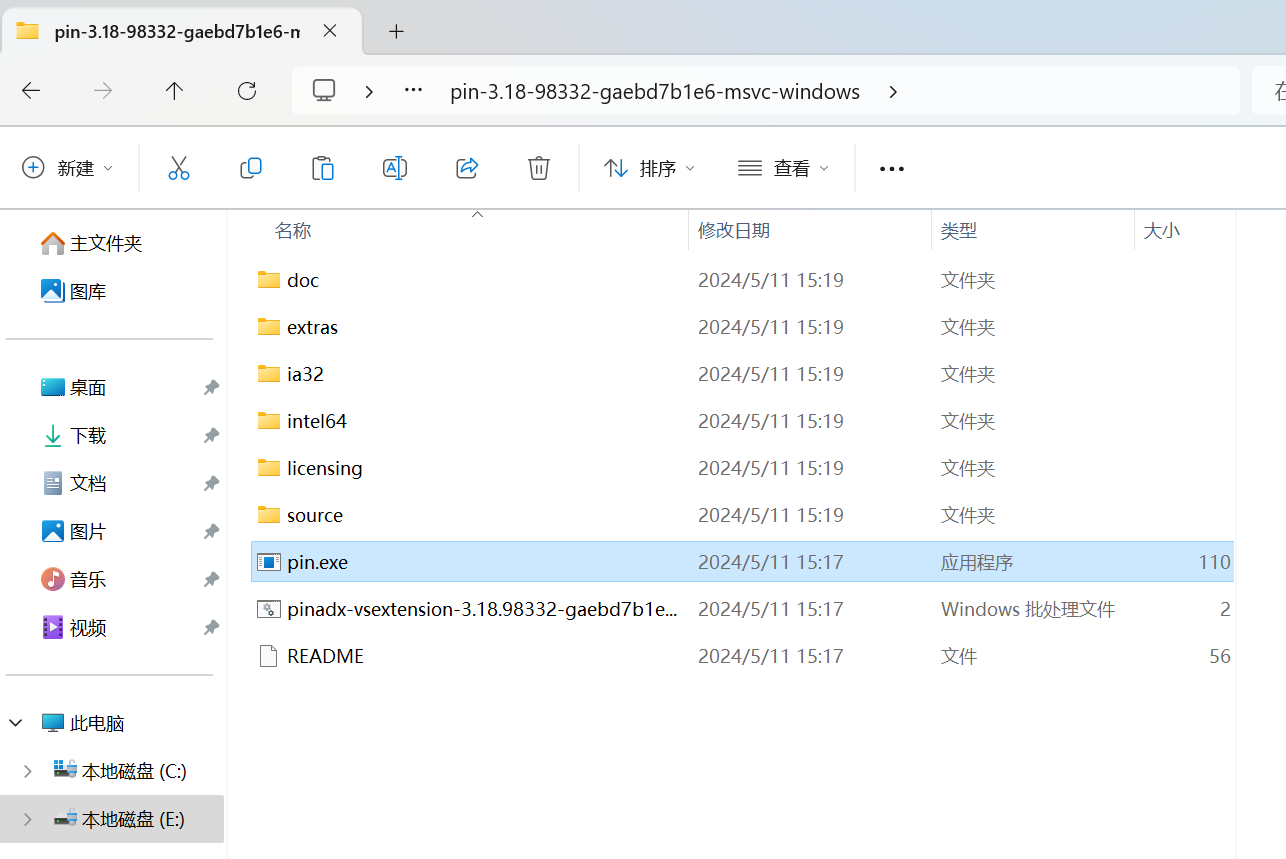
5.对可执行程序进行插桩：



可以发现，每次使用malloc申请堆内存空间的时候，都会输出申请的空间的大小，并且得到申请的空间的起始地址。当使用free释放内存空间的时候，会输出释放的空间的起始地址。

（二）在Windows系统里，基于Windows MyPintool进行malloctrace的复现

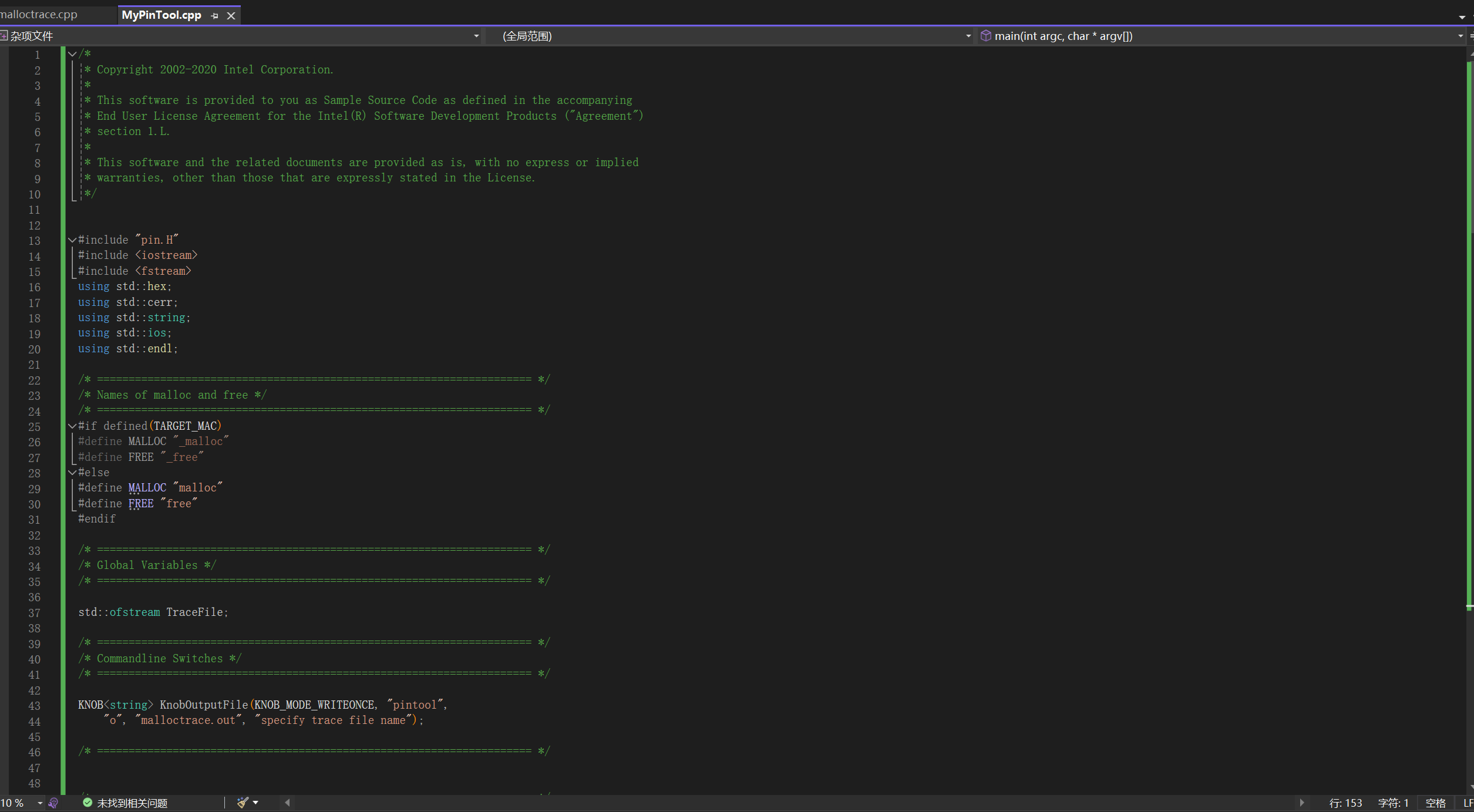
1. 安装Pin3.18，并解压放在E盘中：



2. 找到malloctrace.cpp，将其复制到MyPinTool.cpp：

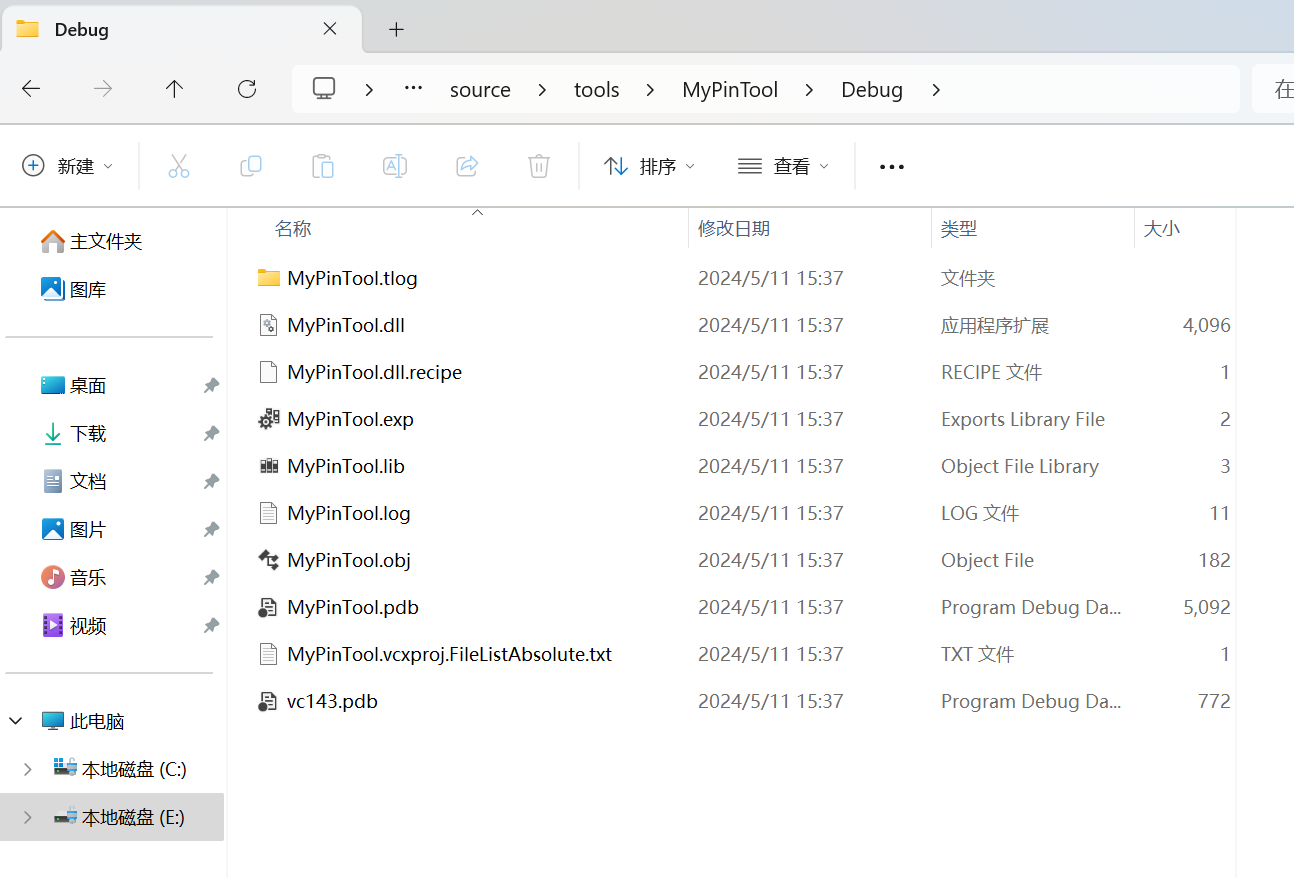
malloctrace.cpp在source/tools/ManualExamples路径下;

MyPinTool.cpp在source/tools/MyPinTool路径下。



3.生成其动态链接库文件：

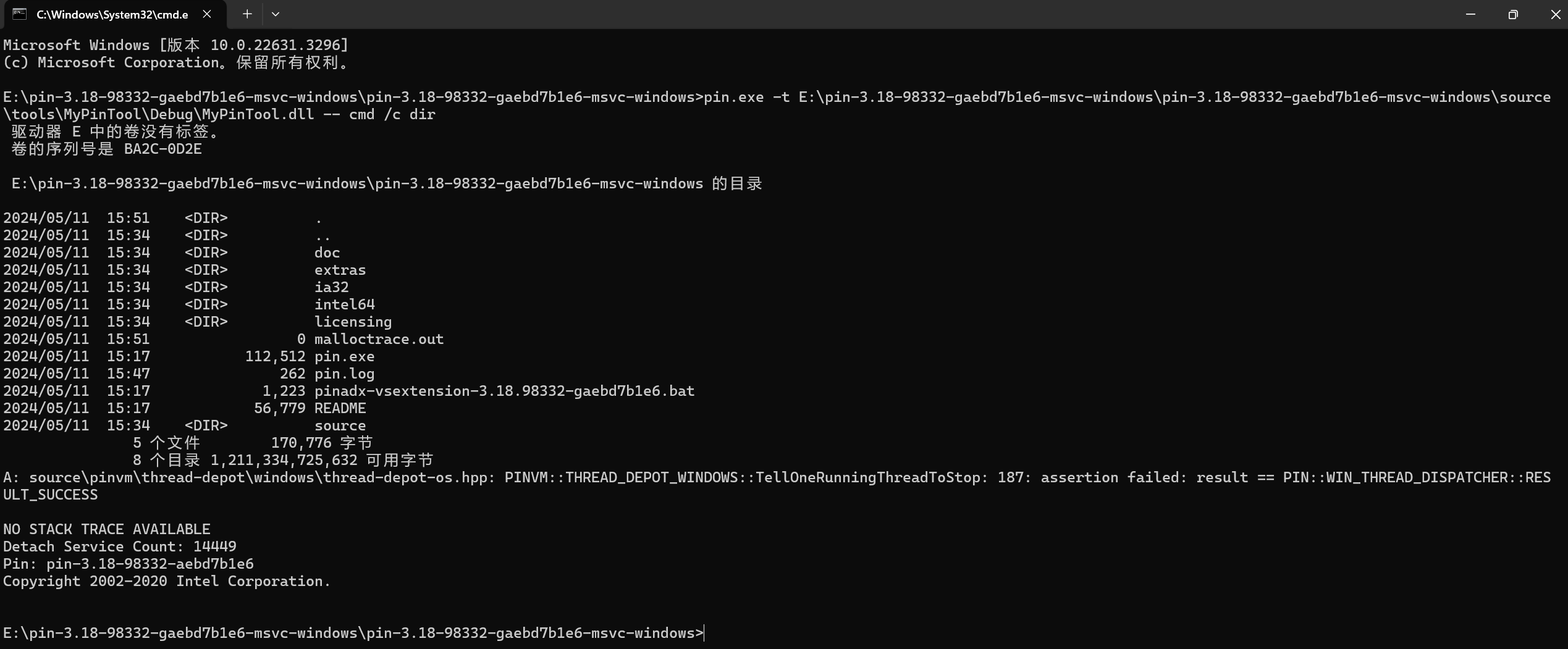
打开MyPinTool.vcxproj，由于我的visual studio是2022的版本，所以先升级再运行。整个项目生成后，在 debug 路径下生成了一个 MyPinTool.dll 文件。



4. 利用pin工具进行程序插桩：

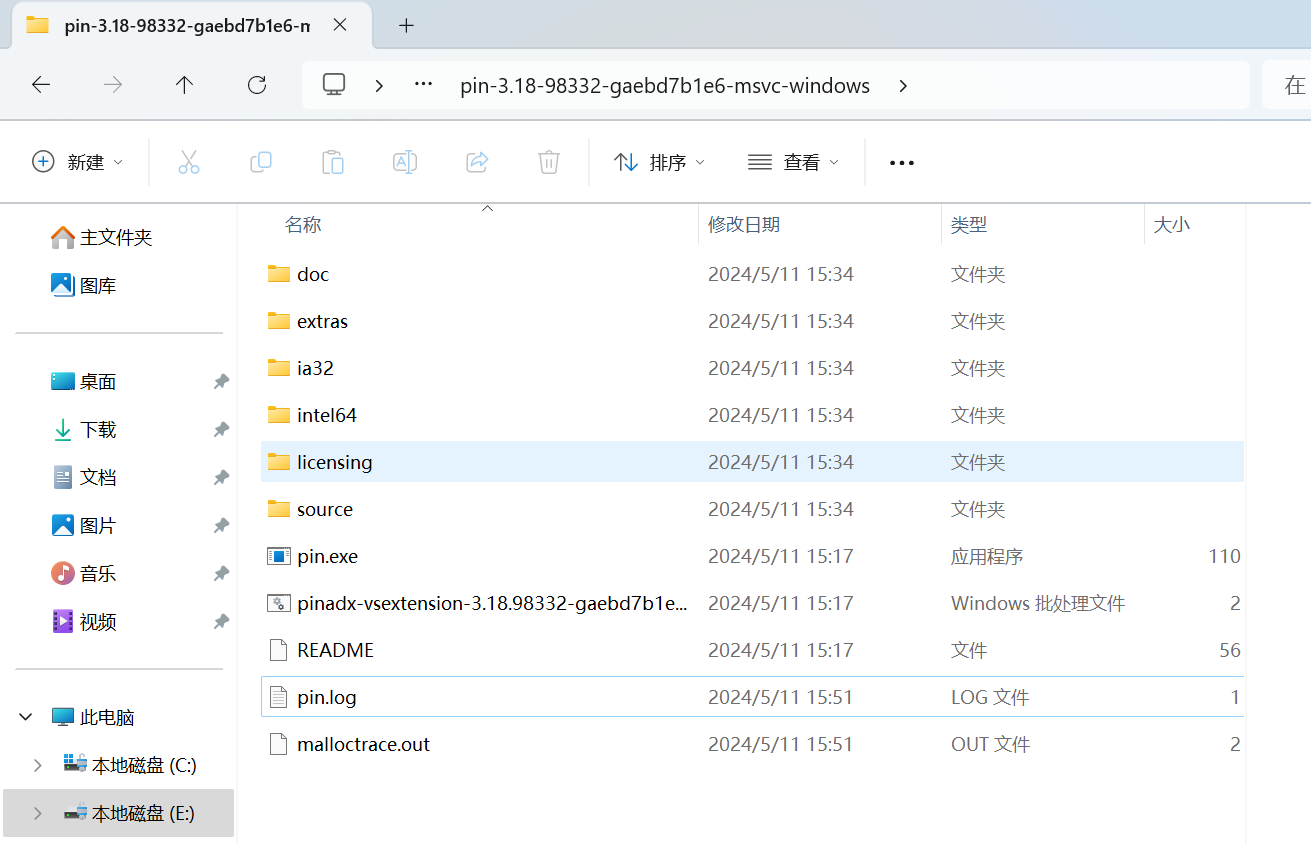
进入E:\pin-3.18-98332-gaebd7b1e6-msvc-windows\pin-3.18-98332-gaebd7b1e6-msvc-windows目录，打开cmd，运行如下命令

pin.exe -t E:\pin-3.18-98332-gaebd7b1e6-msvc-windows\pin-3.18-98332-gaebd7b1e6-msvc-windows\source\tools\MyPinTool\Debug\MyPinTool.dll -- cmd /c dir



5.查看malloctrace.out文件：

在pin路径下生成了一个malloctrace.out文件，打开，如下图：





所以，Malloctrace的作用是打印出程序中调用malloc时传递的参数和返回值，以及调用free时传递的参数。

（三）实验分析总结

1. Pin插桩工具的核心步骤：

首先要进行初始化：通过调用函数PIN\_Init实现

然后注册插桩函数：通过使用XXX\_AddInstrumentFunction注册一个插桩函数，由于使用了指令级插桩，因此在原始程序的每条指令执行前，都会进入到我们注册的插桩函数中，然后执行相应的操作。

之后注册退出回调函数：通过使用PIN\_AddFiniFunctino注册一个程序退出时的回调函数，当应用退出的时候会调用该函数。

最后启动程序：使用函数PIN\_StartProgram启动程序。

2.Pin插桩工具的相关API：

PIN\_InitSymbols():初始化符号处理。在调用其他Pin API之前，通常需要先调用这个函数。

PIN\_Init():Pin工具的初始化操作。在插桩开始前，需要先调用这个函数来初始化Pin工具。

IMG\_AddInstrumentFunction():镜像级插桩，注册一个插桩函数，以在程序加载时对镜像进行插桩。

PIN\_AddFiniFunction():注册一个程序退出时的回调函数，以进行一些结束处理。这个函数允许你在程序退出时执行一些收尾工作，比如输出最终的结果或者释放资源。

PIN\_StartProgram():启动需要进行插桩的程序。在初始化完毕并注册好插桩函数后，使用这个函数来启动目标程序的执行。

RTN\_FindByName():根据函数的名称查找函数的标识。通过这个函数可以获得指定函数的信息，比如地址等。

RTN\_InsertCall():在指定的函数执行前或执行后注册一个回调函数。可以利用这个函数在目标函数的执行前后插入自定义的代码，实现对函数执行过程的监控和控制。

3. malloc和free函数的输入输出信息：

在本次实验中，malloctrace工具记录了malloc和free函数的输入输出信息。通过分析输出文件的结果，我对两个函数的输入输出的理解如下：

a）malloc函数的输入输出信息：

输入：malloc函数接受一个参数，即需要分配的内存大小（以字节为单位）。

输出：malloc函数返回一个指向分配内存起始位置的指针，如果内存分配失败，则返回NULL。

malloctrace工具会记录成功分配内存时的相关信息，例如分配的内存大小、分配的内存地址以及分配时的函数调用栈信息。

b）free函数的输入输出信息：

输入：free函数接受一个参数，即需要释放的内存块的起始地址。

输出：free函数没有明确的输出，它会释放参数所指向的内存块，但不返回任何值。

malloctrace工具会记录成功释放内存时的相关信息，例如释放的内存地址以及释放时的函数调用栈信息。

这些记录的信息有助于我们深入了解程序的内存使用情况，包括内存的分配和释放情况，以及相关操作发生的上下文和位置。通过分析这些信息，我们可以及时发现内存泄漏、重复释放等问题，并进行相应的优化和改进。

**四、心得体会：**

本次实验主要分为两个部分：在Kali中利用malloctrace进行内存分析，在Windows10下利用MyPinTool框架进行malloctrace分析可执行文件。通过这个过程，我深入了解了Pin工具的基本工作原理，并学习了如何使用其提供的API来实现对程序的动态分析和修改。

首先，我了解到了程序插桩的概念，即在程序执行过程中，通过在特定位置插入代码来观察程序的行为或者对程序进行修改。而Pin工具提供了一个强大的框架，可以让我们在不需要源代码的情况下对程序进行插桩。

在利用malloctrace工具进行内存分析时，我深入理解了程序运行时内存分配和释放的行为，并了解了malloctrace输出信息的含义，包括内存分配和释放函数的调用参数和返回值、堆栈信息等。在分析可执行文件时，我进一步加深了对Pin插桩工具的理解，并学会了利用MyPinTool框架进行Pin Tool的编写和调试。

通过这个过程，我对程序的执行过程有了更深入的理解，也掌握了一种强大的工具来进行程序分析和优化。我相信这些知识和技能会在我未来的研究和工作中发挥重要作用，让我能够更好地理解和优化程序的性能，提升自己在计算机领域的技术能力。