《程序插桩及Hook实验》实验报告

姓名：汤清云 学号：2013536 班级： 1075

**实验名称：**

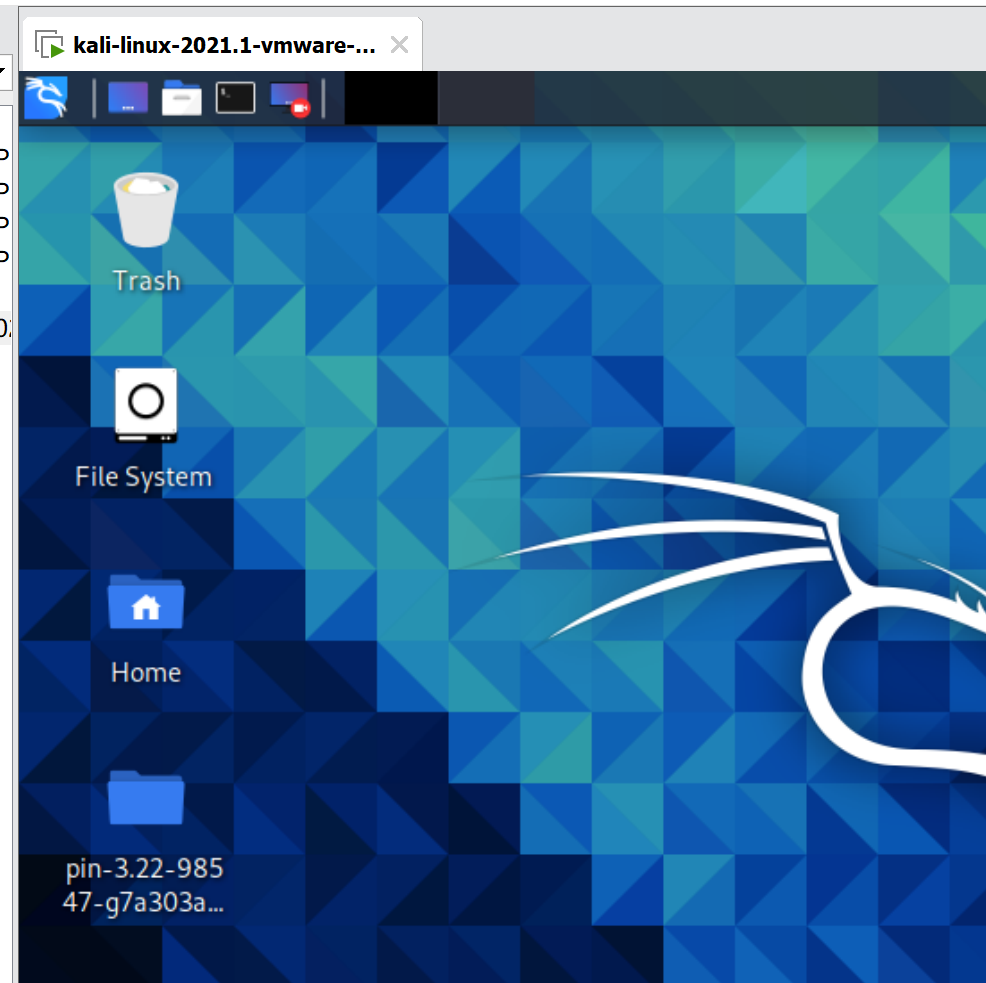
**程序插桩以及hook实验**

**实验要求：**

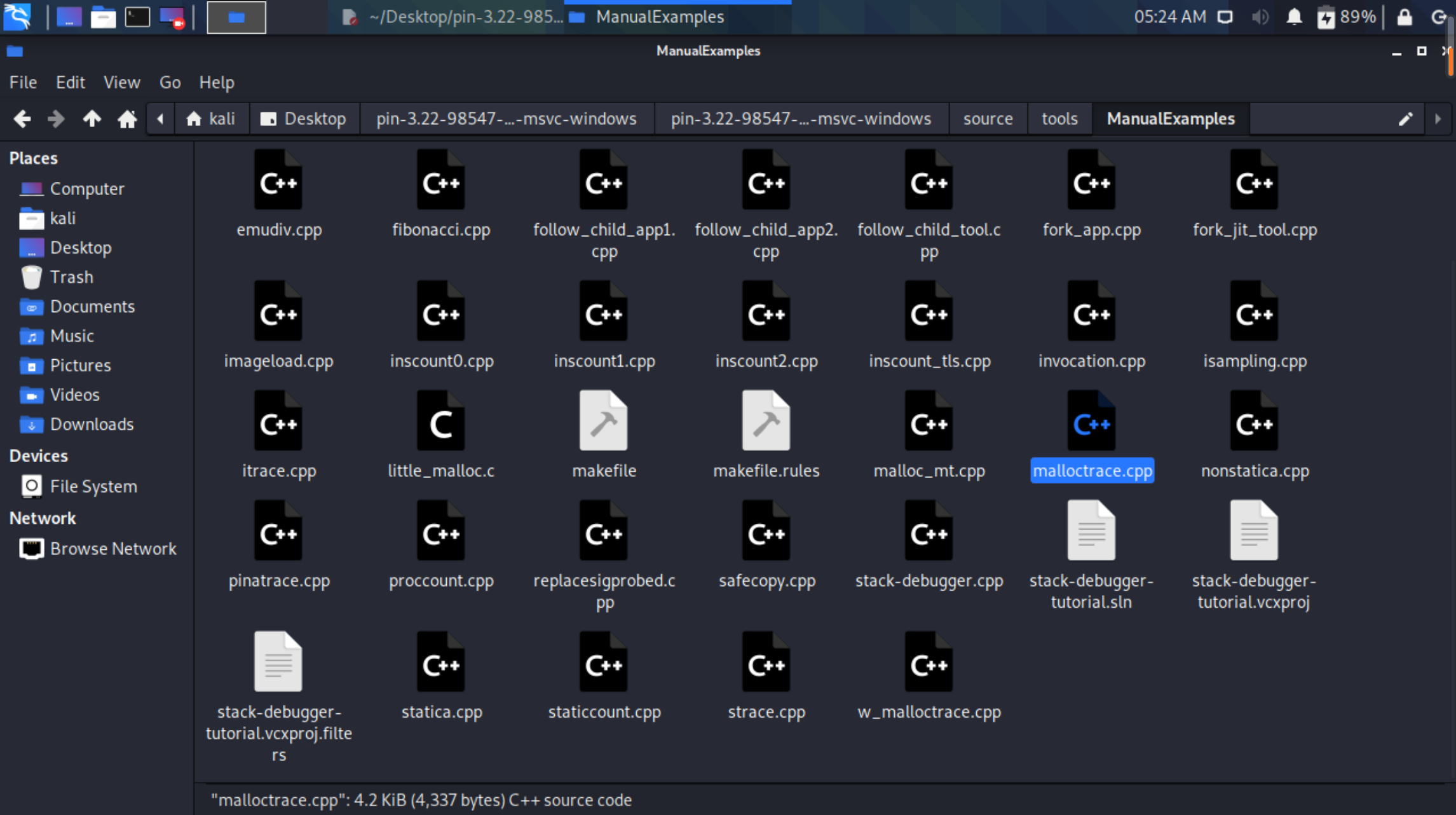
**实验过程：**

1. **安装pin：**

**在官网下载pin压缩包，解压缩后移入kali虚拟机：**



1. **打开source-tools-manualExamples-malloctrace.cpp，查看该pintool**

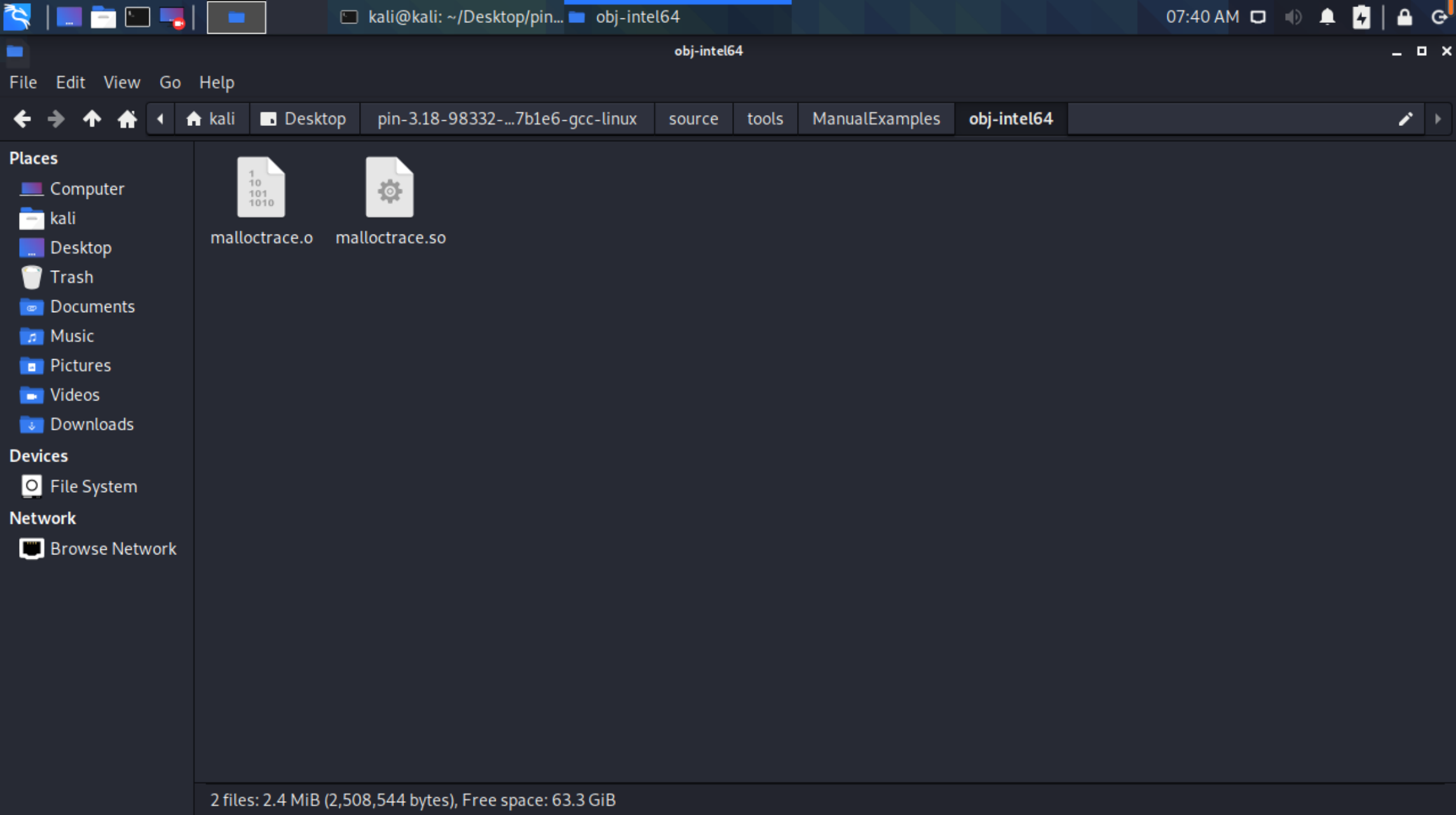


1. **Malloctrace中代码如下：**

|  |
| --- |
| **#include "pin.H"**  **#include <iostream>**  **#include <fstream>**  **using std::cerr;**  **using std::endl;**  **using std::hex;**  **using std::ios;**  **using std::string;**  **#if defined(TARGET\_MAC)**  **#define MALLOC "\_malloc"**  **#define FREE "\_free"**  **#else**  **#define MALLOC "malloc"**  **#define FREE "free"**  **#endif**  **std::ofstream TraceFile;**  **KNOB< string > KnobOutputFile(KNOB\_MODE\_WRITEONCE, "pintool", "o", "malloctrace.out", "specify trace file name");**  **//指定输出文件为malloctrace.out**  **VOID Arg1Before(CHAR\* name, ADDRINT size) { TraceFile << name << "(" << size << ")" << endl; }**  **VOID MallocAfter(ADDRINT ret) { TraceFile << " returns " << ret << endl; }**  **VOID Image(IMG img, VOID\* v)**  **{**  **//测试malloc()和free()函数。打印每个malloc()或free()的输入参数，以及malloc()的返回值。**  **// 找到malloc函数**  **RTN mallocRtn = RTN\_FindByName(img, MALLOC);**  **if (RTN\_Valid(mallocRtn))**  **{**  **RTN\_Open(mallocRtn);**  **//使用malloc()来打印输入参数值和返回值。**  **RTN\_InsertCall(mallocRtn, IPOINT\_BEFORE, (AFUNPTR)Arg1Before, IARG\_ADDRINT, MALLOC, IARG\_FUNCARG\_ENTRYPOINT\_VALUE, 0,IARG\_END);**  **RTN\_InsertCall(mallocRtn, IPOINT\_AFTER, (AFUNPTR)MallocAfter, IARG\_FUNCRET\_EXITPOINT\_VALUE, IARG\_END);**  **RTN\_Close(mallocRtn);**  **}**  **//找到free函数。**  **RTN freeRtn = RTN\_FindByName(img, FREE);**  **if (RTN\_Valid(freeRtn))**  **{**  **RTN\_Open(freeRtn);**  **//使用free()输出输入参数值。**  **RTN\_InsertCall(freeRtn, IPOINT\_BEFORE, (AFUNPTR)Arg1Before, IARG\_ADDRINT, FREE, IARG\_FUNCARG\_ENTRYPOINT\_VALUE, 0,**  **IARG\_END);**  **RTN\_Close(freeRtn);**  **}**  **}**  **//当应用退出的时候调用本函数**  **VOID Fini(INT32 code, VOID\* v) { TraceFile.close(); }**  **INT32 Usage()**  **{**  **cerr << "This tool produces a trace of calls to malloc." << endl;**  **cerr << endl << KNOB\_BASE::StringKnobSummary() << endl;**  **return -1;**  **}**  **int main(int argc, char\* argv[])**  **{**  **//初始化引脚和符号管理器**  **PIN\_InitSymbols();**  **if (PIN\_Init(argc, argv))//调用函数pin\_init完成初始化**  **{**  **return Usage();**  **}**  **//写入文件，因为cout和cerr可能被应用程序关闭**  **TraceFile.open(KnobOutputFile.Value().c\_str());**  **TraceFile << hex;**  **TraceFile.setf(ios::showbase);**  **//将要调用的图像配准到仪器函数中。**  **IMG\_AddInstrumentFunction(Image, 0);//注册一个插桩函数，在原始程序的每条指令执行前都会进入image函数中。**  **PIN\_AddFiniFunction(Fini, 0);//注册退出回调函数，在退出时调用该函数**  **// Never returns**  **PIN\_StartProgram();//使用该函数启动程序**  **return 0;**  **}** |

1. **编译运行，产生动态链接库：**

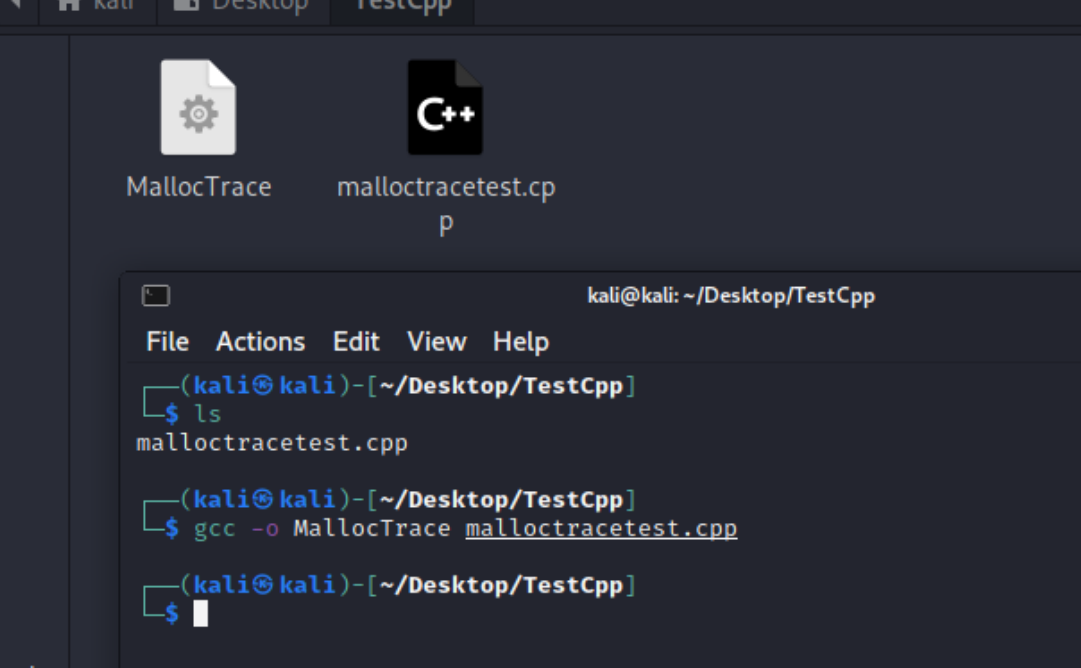
**使用语句make malloctrace.test TARGET intel-64生成链接库**



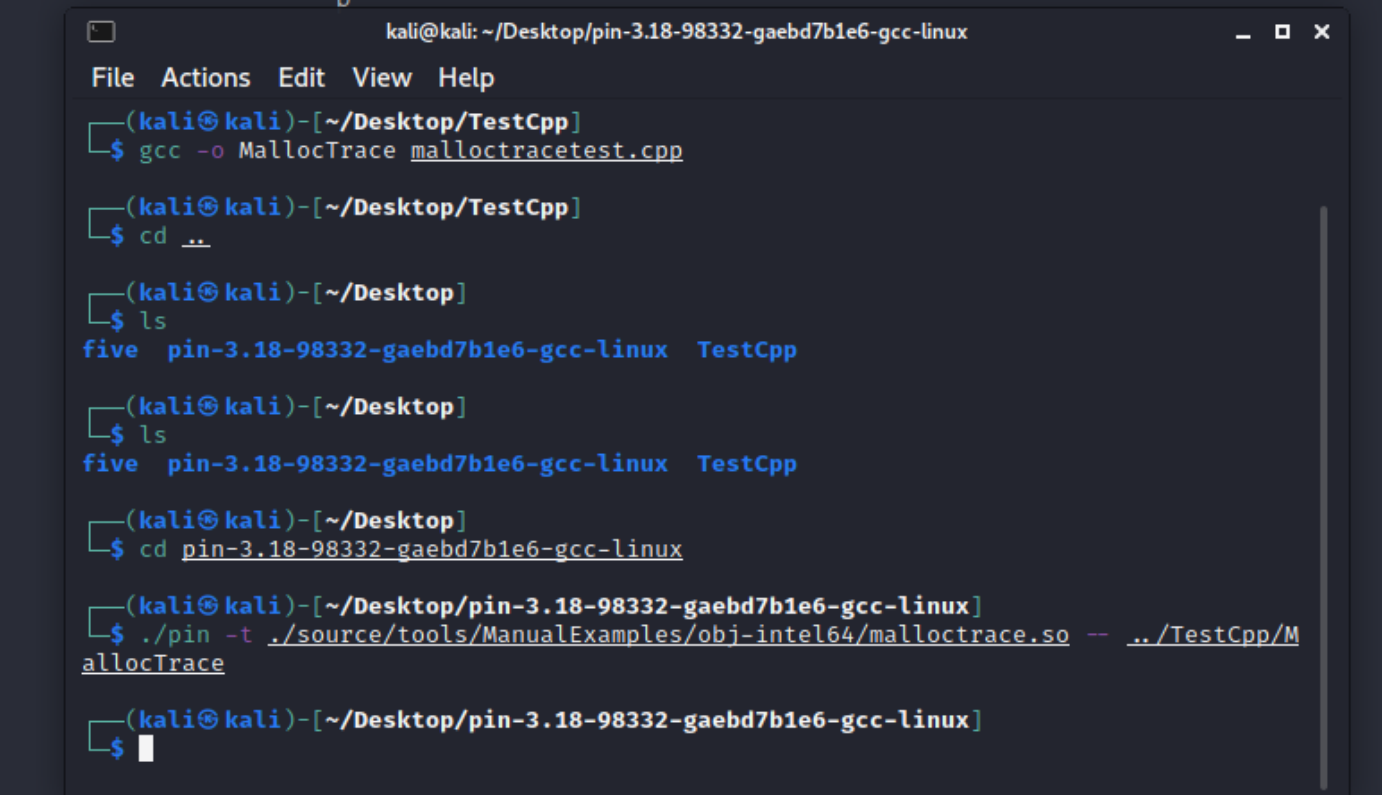
1. **在别处生成test文件，包含malloc和free函数。**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **#include <stdlib.h>**  **int main()**  **{**  **char\* str;**  **/\* 最初的内存分配 \*/**  **str = (char\*)malloc(15);**    **free(str);**    **return 0;**  **}** |

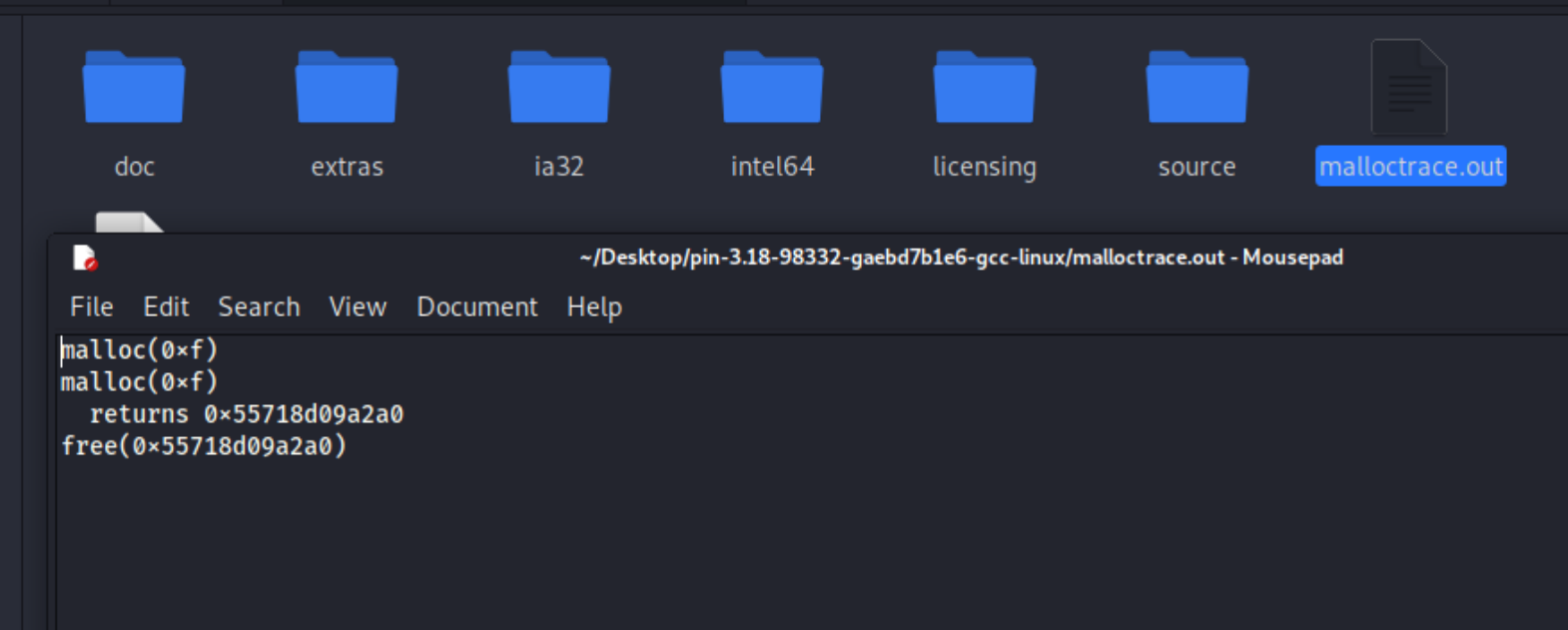
1. **使用语句gcc -o MallocTrace malloctracetest.cpp进行编译**



1. **使用语句./pin -t ./source/tools/ManualExamples/obj-intel64/malloctrace.so -- ../TestCpp/MallocTrace链接动态链接库并且运行：**



1. **在pin文件夹下可看见输出结果：**



1. **0xf即所预留空间大小，return的值就是设定空间的首地址。**

**心得体会：**

**学习了如何使用pin实现函数插桩。**