《漏洞利用及渗透测试基础》实验报告

姓名：赵悦蛟 学号：2313650 班级： 1071

**实验名称：**

程序插桩和HOOK实验

**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。

**实验过程：**

1. 安装pintool

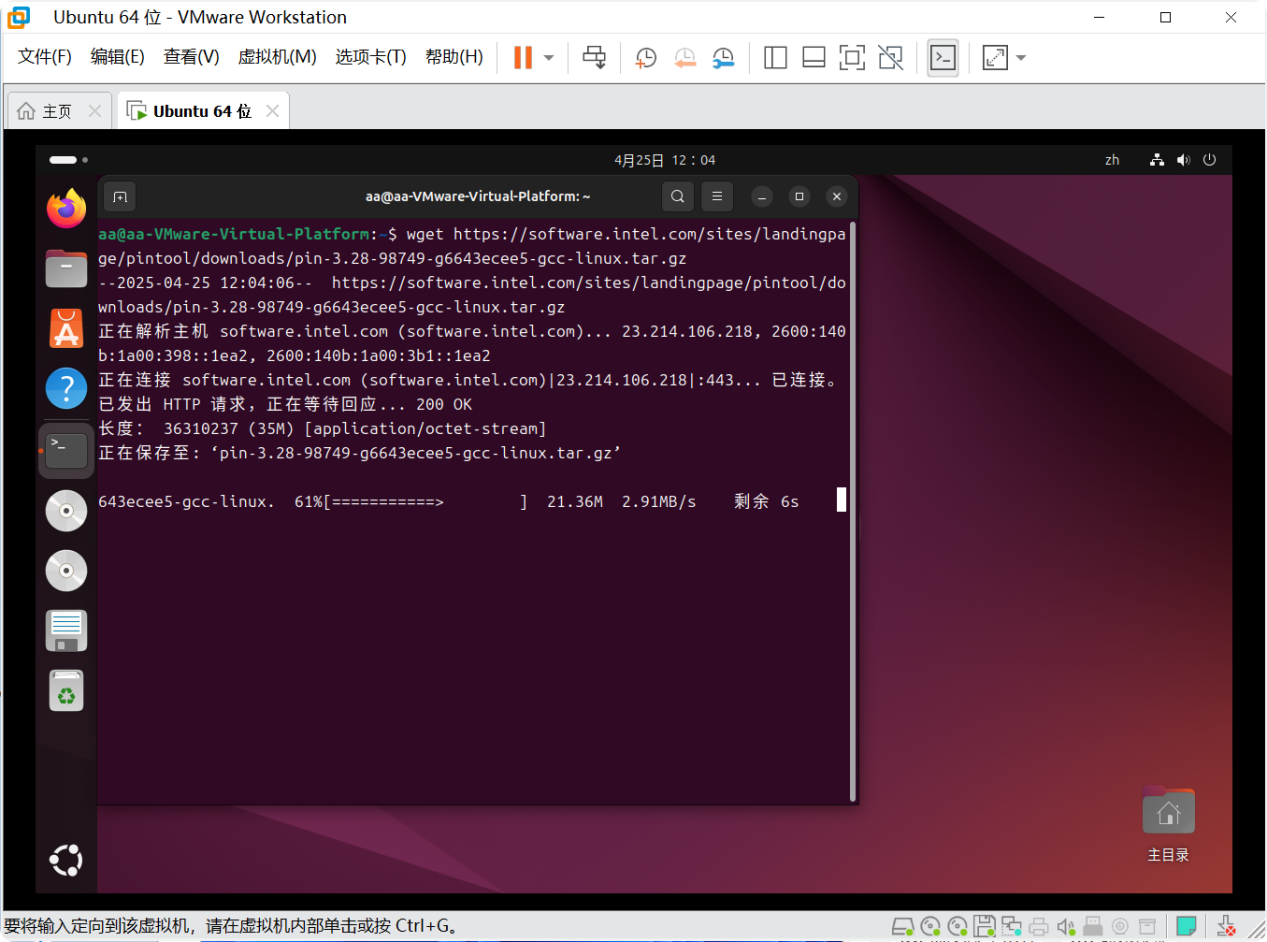


图 1下载pin

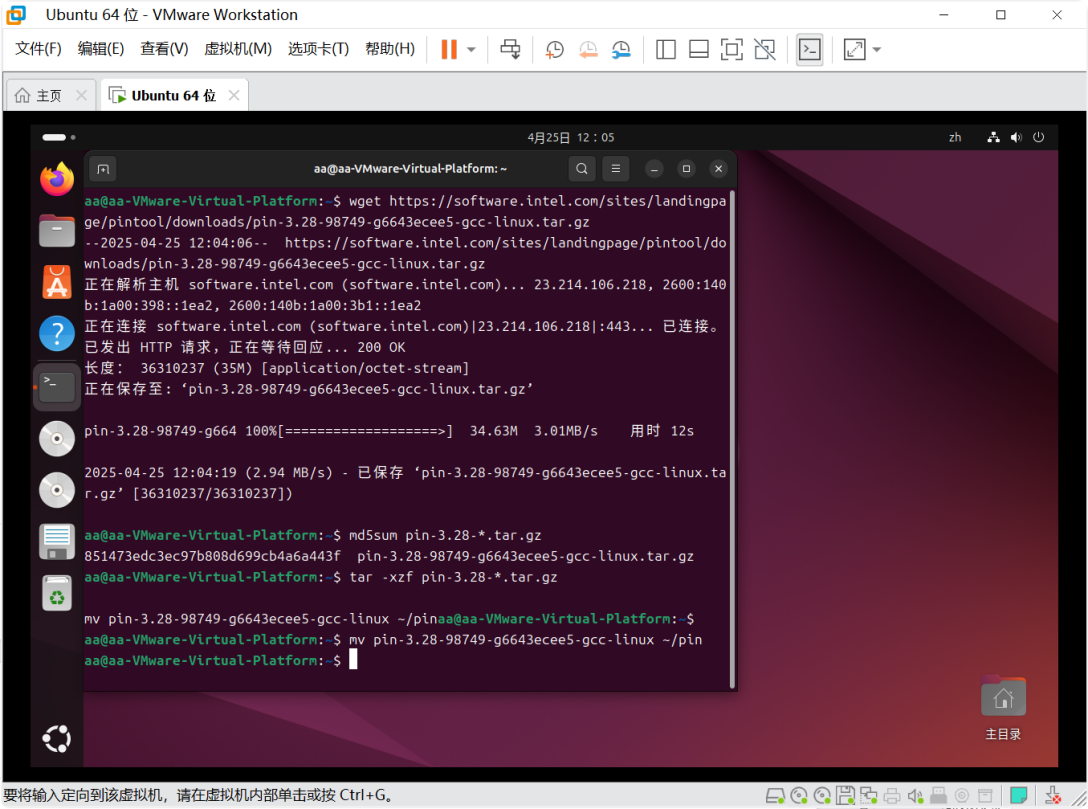


图 2解压并安装

然后使用echo 'export PIN\_ROOT=~/pin' >> ~/.bashrc， echo 'export PATH=$PATH:$PIN\_ROOT' >> ~/.bashrc ，source ~/.bashrc 命令为虚拟机设置环境变量

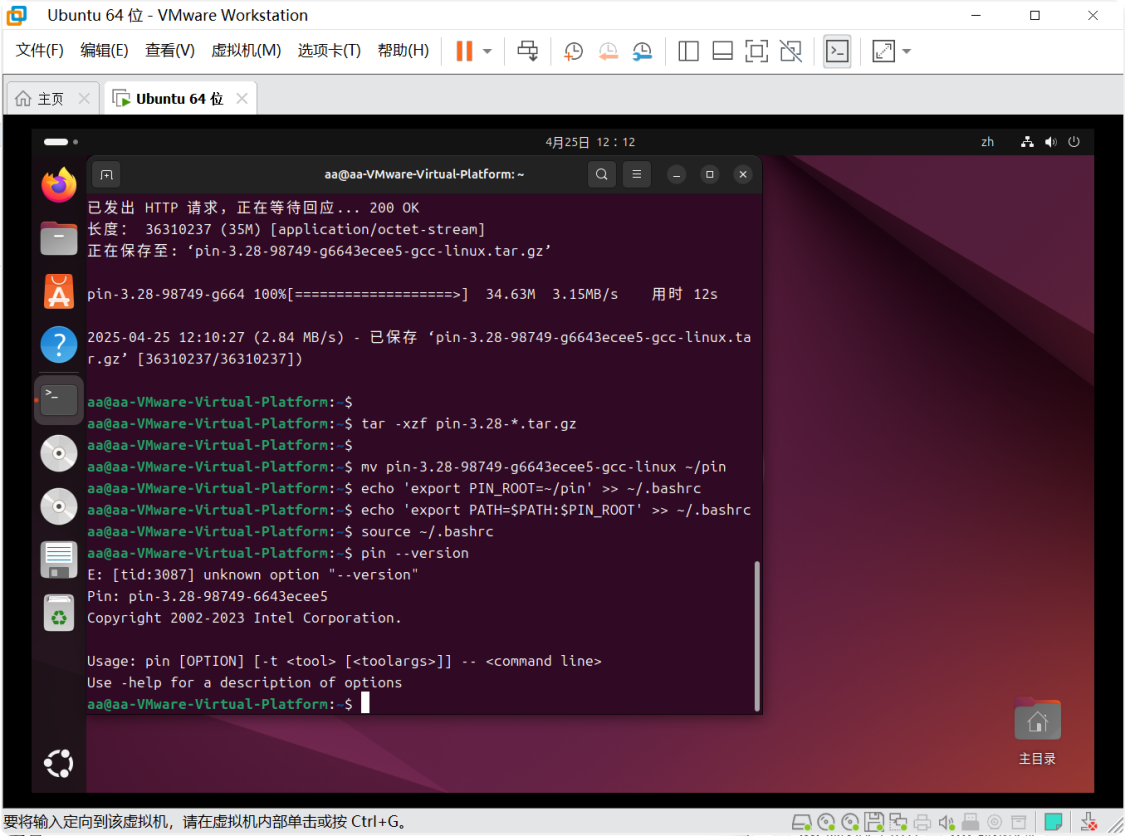


图 3安装配置完成

1. 编译pintool

在虚拟机的终端使用make obj-intel64/inscount0.so命令编译pintool

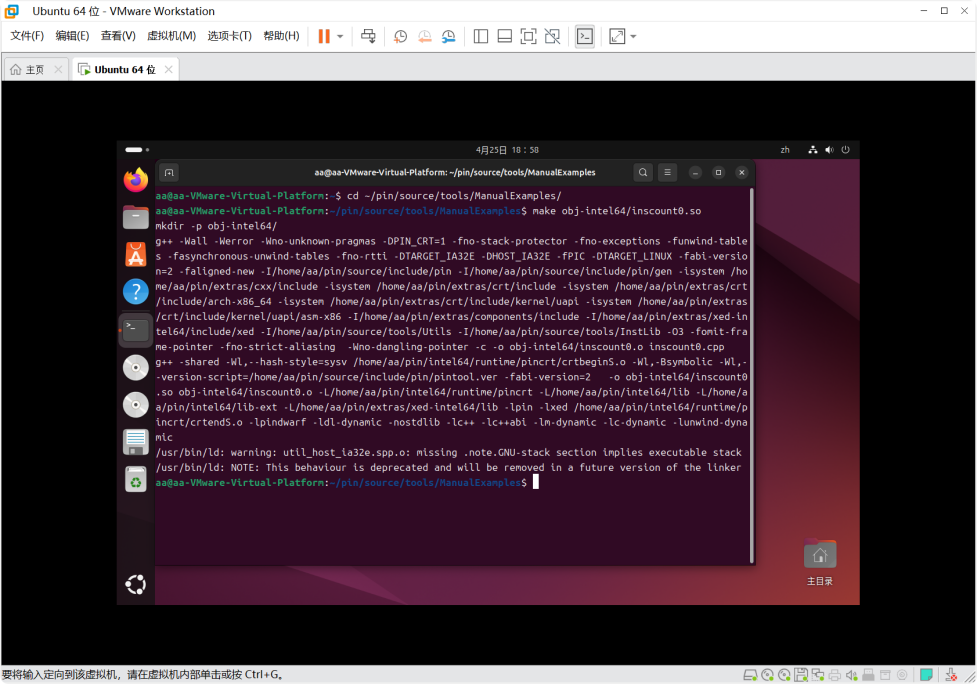


图 4编译pintool

1. 复现插桩实验

1）创建一个新文件 FirstCpp.c，作为测试文件

使用cd ~/pin/source/tools/ManualExamples ，nano FirstCpp.c命令创建测试程序

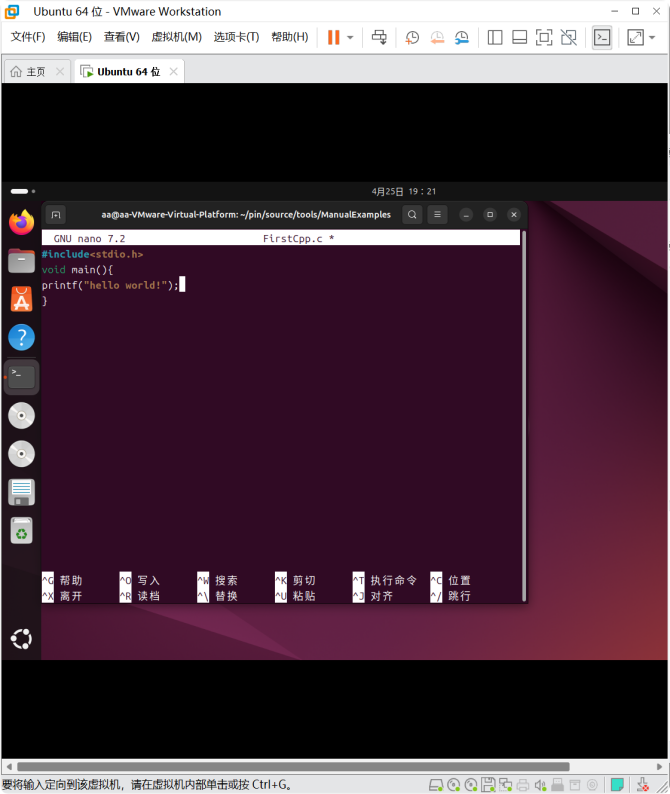


图 5编写测试程序

使用gcc -o FirstCpp FirstCpp.c 命令编译测试程序

2）进行插桩

使用pin -t obj-intel64/inscount0.so -- ./FirstCpp，~/pin/pin -t ~/pin/source/tools/ManualExamples/obj-intel64/malloctrace.so -- ./FirstCpp命令进行插桩

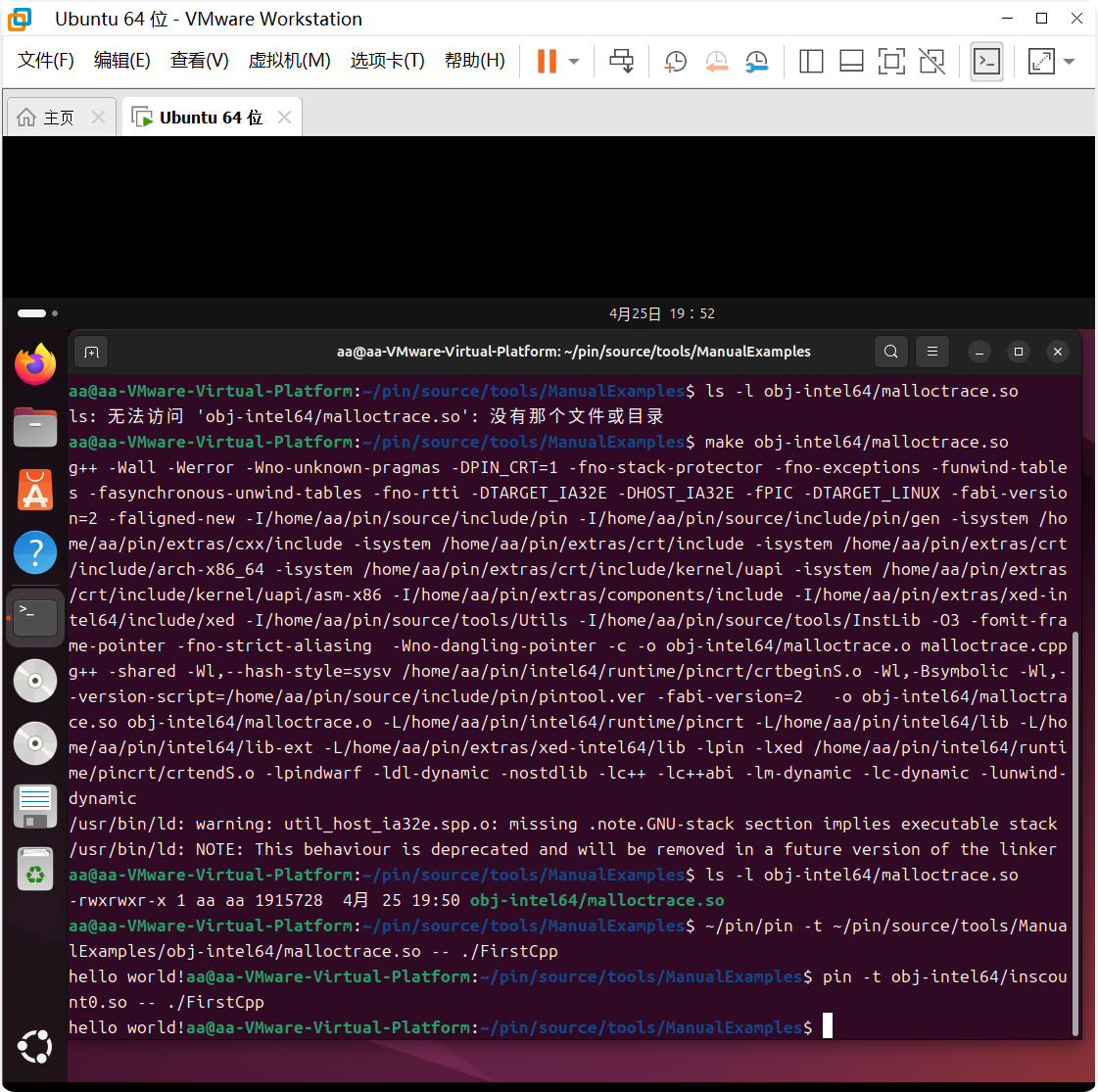
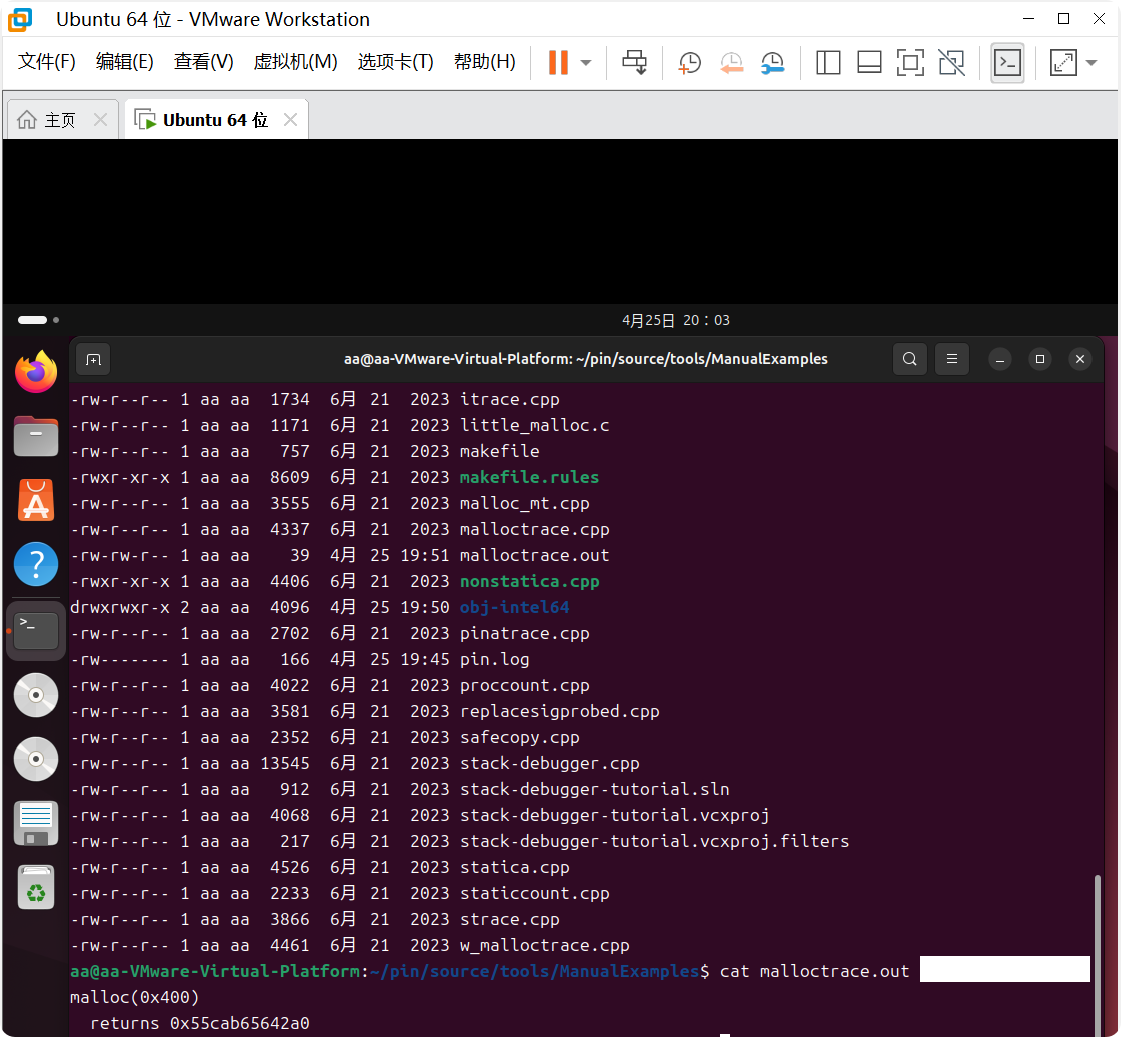


图 6运行插桩命令

成功执行，输出helloworld!

1. 查看输出结果

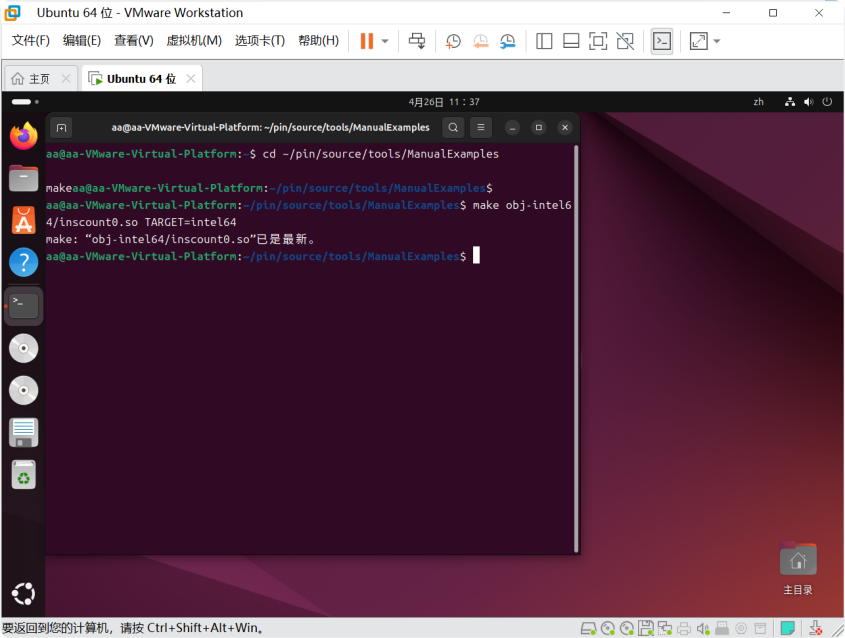


如图，进行插桩测试之后生成了一个输出文件 malloctrace.out，查看这个文件，文件内容为malloc(0x400) returns 0x55cab65642a0，表示程序通过 malloc 动态分配了 1024字节的内存空间分配的内存起始地址为 0x55cab65642a0。malloc 函数成功地分配了内存，并返回了指向该内存块的指针。

4.Inscount0 的复现

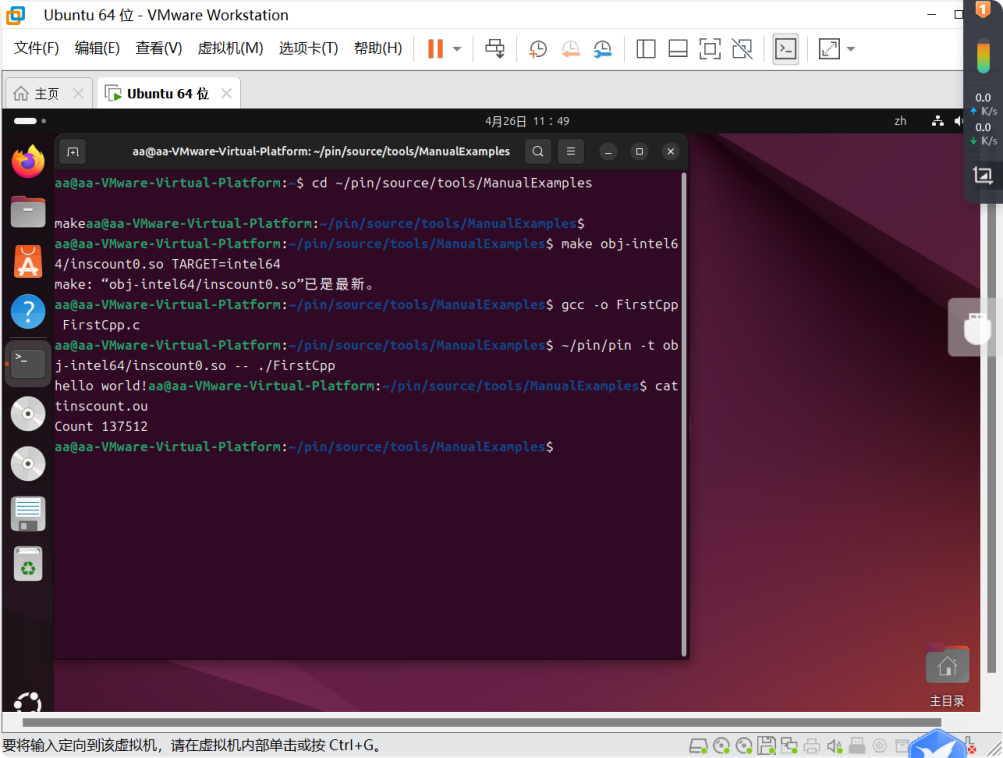
1）编译原始Inscount0

用cd ~/pin/source/tools/ManualExamples，make obj-intel64/inscount0.so TARGET=intel64命令编译原始Inscount0



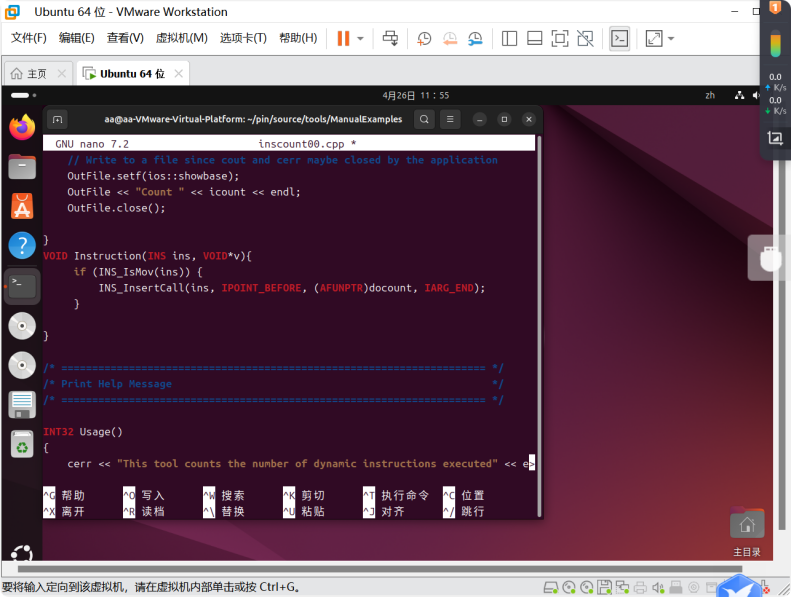
2）重新编译测试程序并运行inscount0

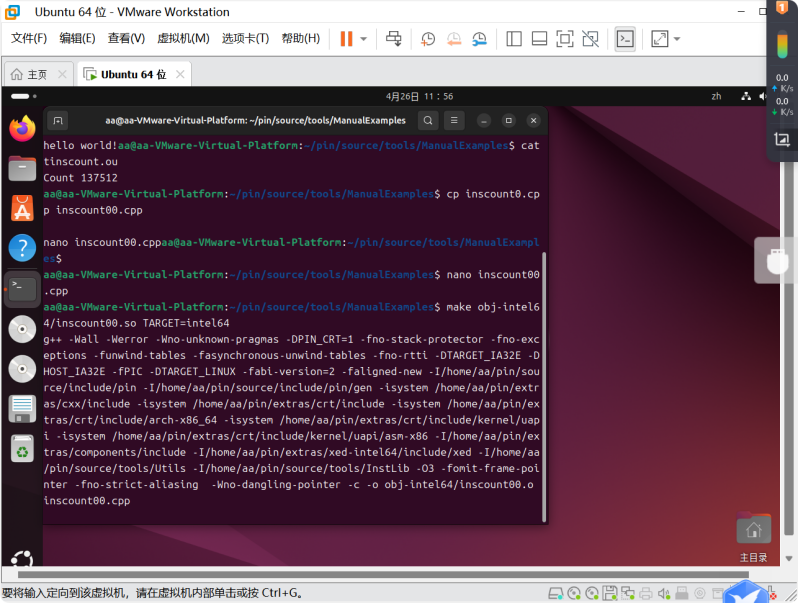
使用 gcc -o FirstCpp FirstCpp.c 指令编译成可执行文件 FirstCpp。然后用~/pin/pin -t obj-intel64/inscount0.so -- ./FirstCpp，cat inscount.out 命令使用生 成的 inscount0.so 执 行 可 执 行 文 件 FirstCpp ，得到插桩后的计数结果



1. 修改代码重新编译

使用cp inscount0.cpp inscount00.cpp，nano inscount00.cpp命令编辑新工具，在Instruction 函数中修改添加条件判断，仅统计mov指令。然后编译新工具，重新运行。





局部截取_20250426_120100

**心得体会：**

通过本次实验的实践操作，我对动态二进制插桩技术有了更为深刻的认识和体会。在复现malloctrace工具的过程中，当看到终端输出中清晰显示出内存分配的地址和大小信息时，那种将抽象理论转化为具体实践的感受尤为强烈。这种直观的数据呈现方式，让我对程序运行时的内存管理机制有了更立体的理解，也让我意识到插桩技术在内存泄漏检测和性能分析中的实际价值。

而在inscount0工具的复现和改造过程中，最令我印象深刻的是通过简单的代码修改就能实现对特定指令的精准统计。当第一次看到修改后的工具仅输出MOV指令的计数结果时，我切实感受到了二进制插桩技术的灵活性和强大之处。这个过程不仅锻炼了我的代码调试能力，更让我学会了如何根据实际需求来定制插桩工具，这种能力对于未来的研究和开发工作都大有裨益。

整个实验过程中遇到的各类问题，从环境配置到代码调试，都促使我不断查阅文档、思考解决方案。特别是在处理编译错误时，通过分析报错信息逐步定位问题的过程，让我对Pin工具的工作原理有了更深入的认识。这次实验经历让我明白，掌握一项技术不仅要理解其原理，更重要的是通过实践来积累经验，培养解决实际问题的能力。这些收获都将为我后续在程序分析、系统优化等领域的研究打下坚实基础。