题目：冷补丁实验

一、实验目的：

理解二进制文件的安全补丁概念、掌握二级制文件修复、编辑的基本原理

二、解题思路:

2.1 第一次补丁的应用

在应用第一次补丁之前，程序的行为是两个字符串都会输出。如图所示，我们可以看到具体的输出内容。**注意文件需要提权。**

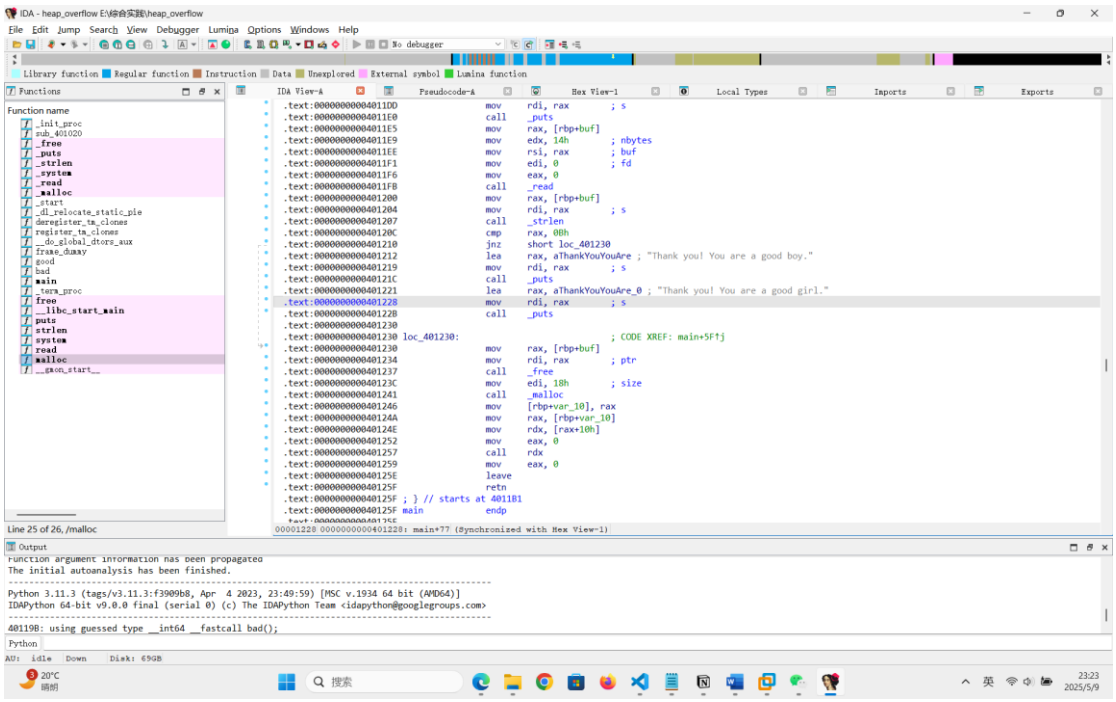

图 2-1 ida 相关汇编代码处截图

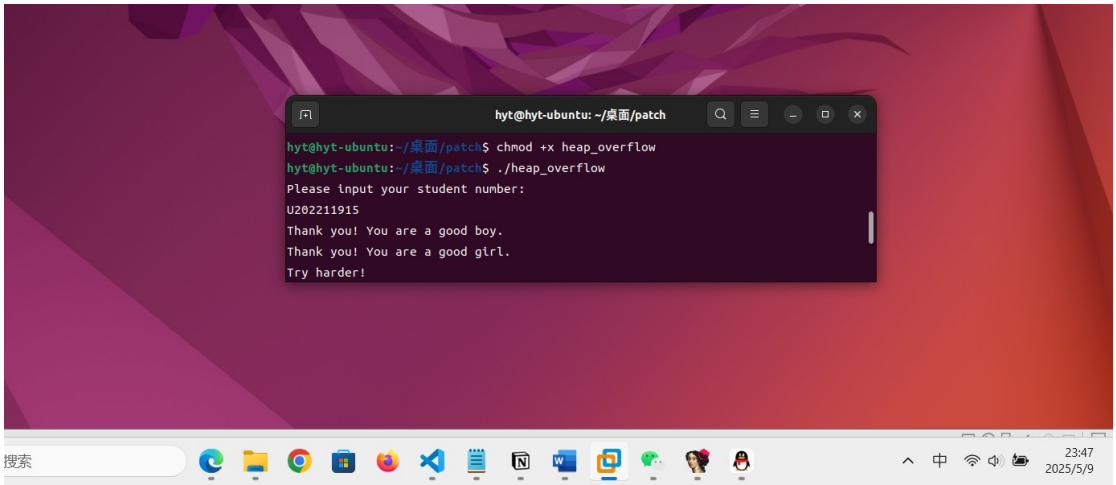

图 2-2 运行原始程序输出

我使用的补丁方法为，通过修改特定的程序指令为 NOP（无操作指令），我们可以取消特定功能的执行。使用补丁允许我们更细致地控制程序的输出。

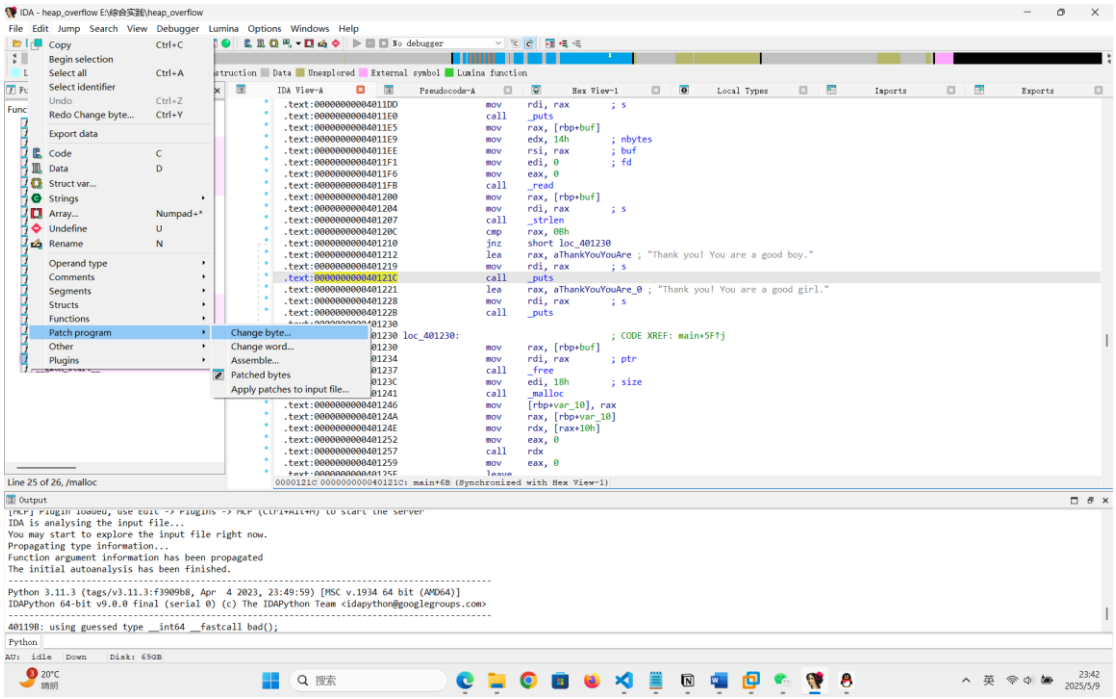

图 2-3 打补丁的具体步骤 1

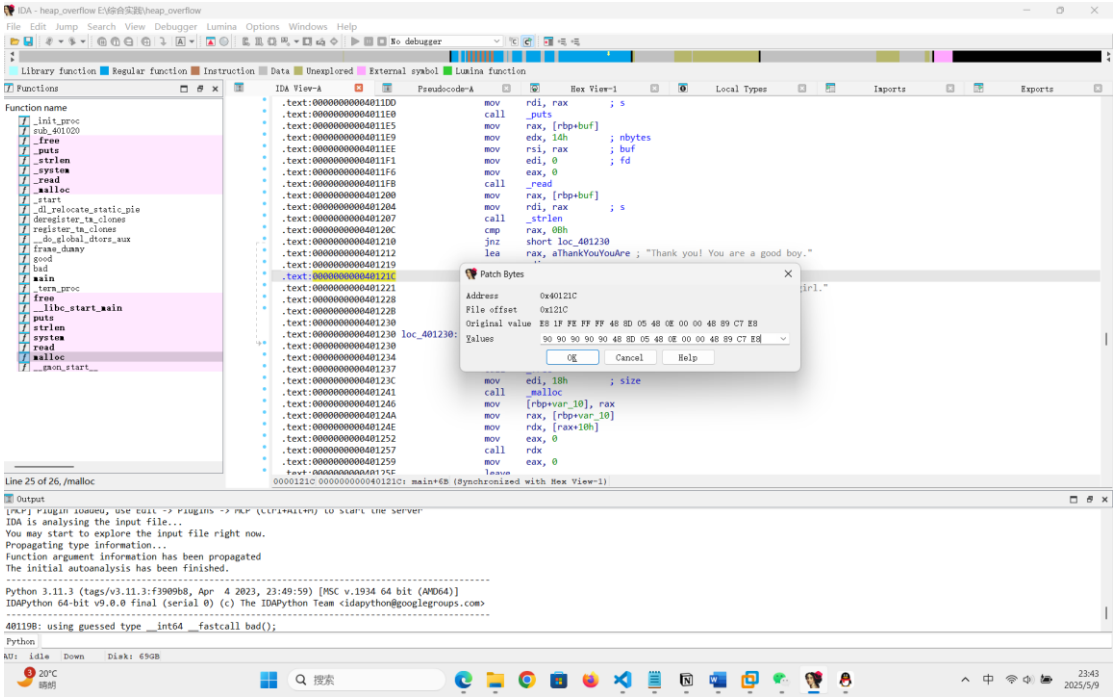

图 2-4 打补丁的具体步骤 2

如下图 2-5 所示，可以观察到部分指令段编程 nop。同时根据图 2-6 的步骤进行导出为 **heap_overflow_patch 1**。

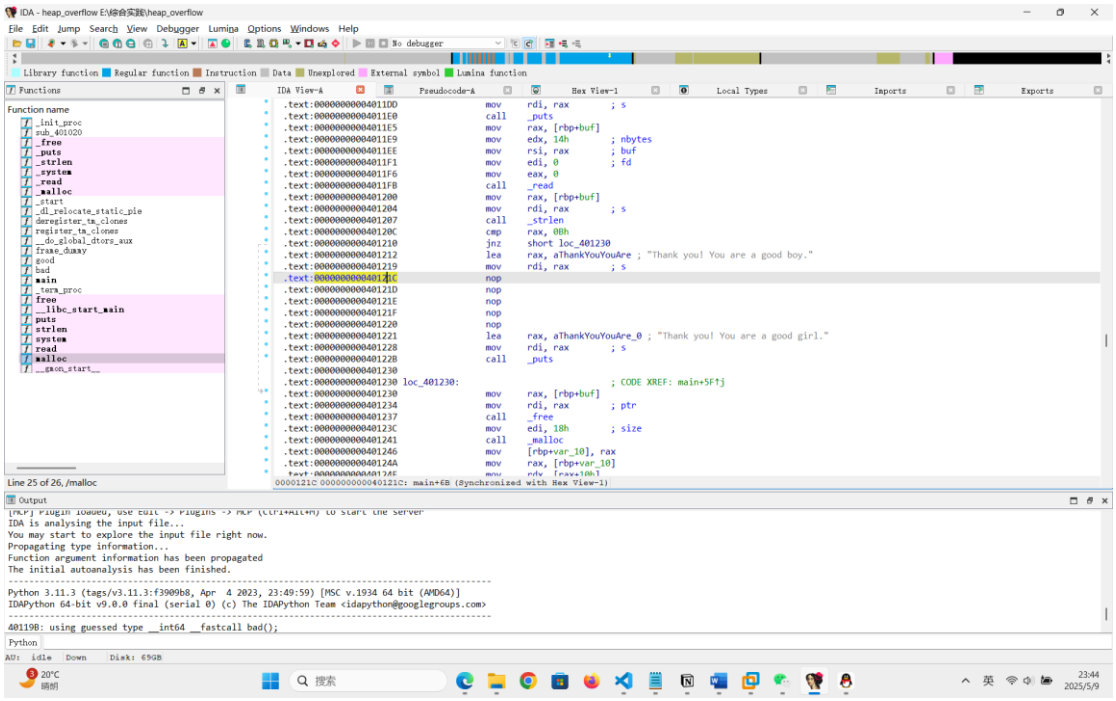

图 2-5 打补丁的具体步骤 3

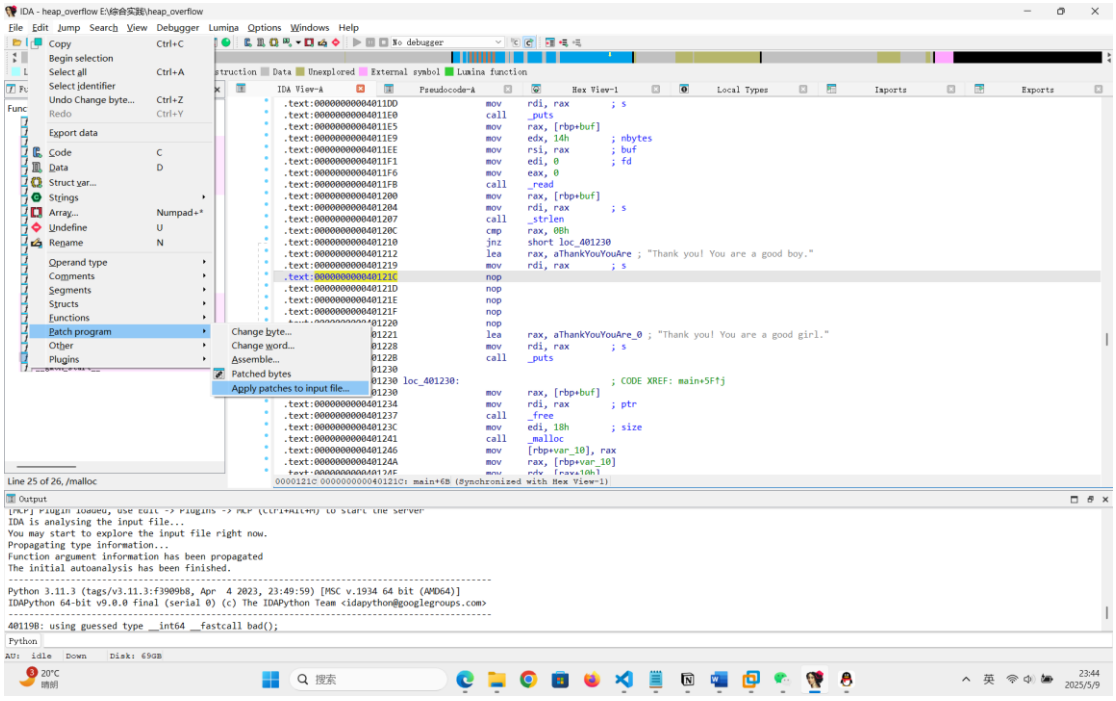

图 2-6 打补丁的具体步骤 4

按照上面的顺序和方法修改指令后，我们导出修改后的程序并在虚拟机中运行。执行结果显示，此时程序仅输出了其中一个性别相关的语句，实现了修改目标。

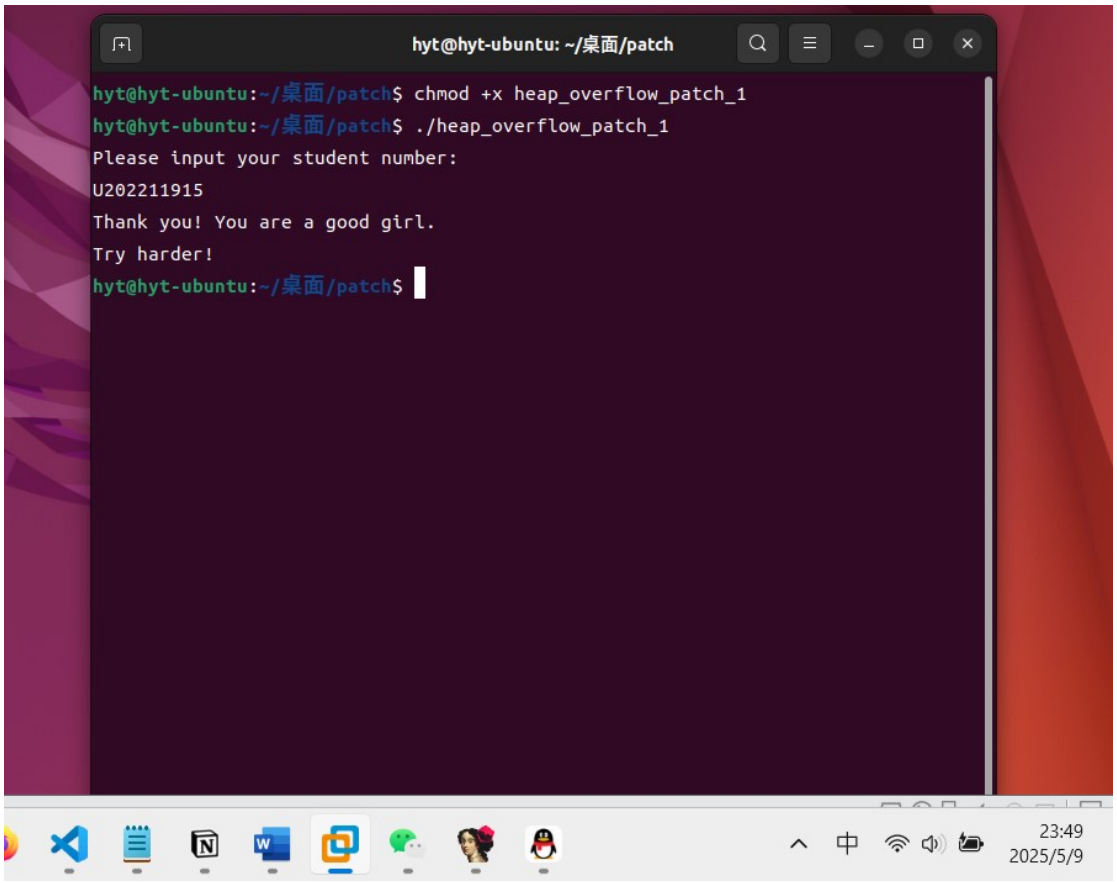

图 2-7 第一个补丁成功

2.2 第二次补丁的应用

第二次补丁存在栈溢出。原程序中的 EDX 寄存器被赋值为 0x14（即 20），这表示程序能够接受 20 个字符的输入。多出的字符将被截断，增加了溢出的风险。

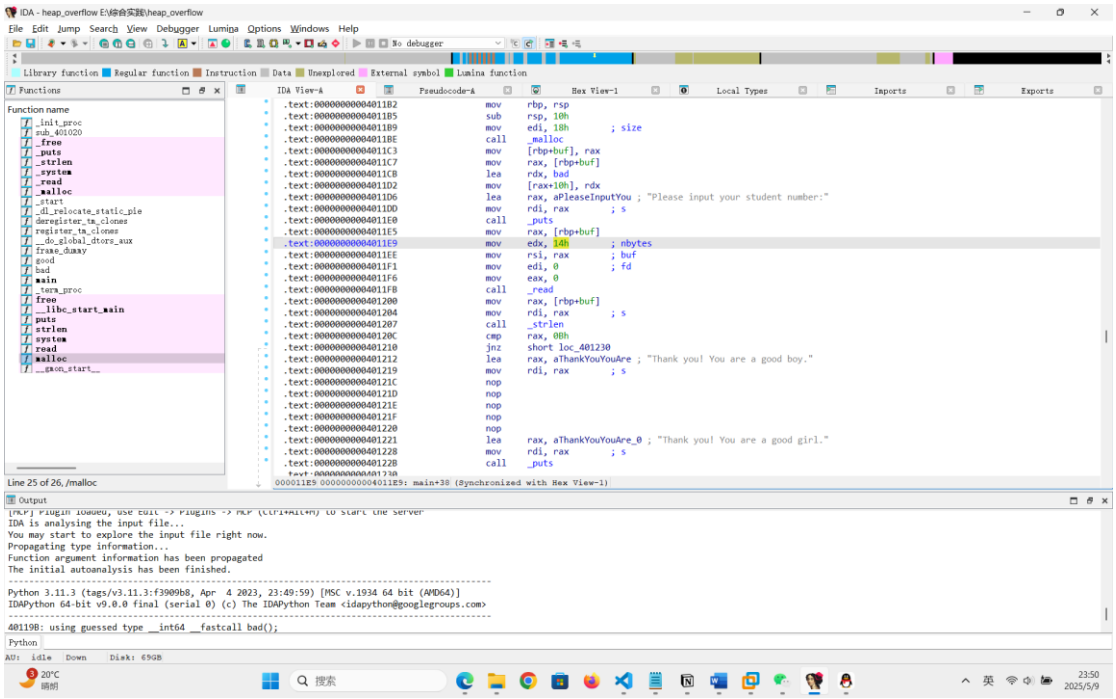

图 2-8 第二个补丁的汇编代码位置

我们运行程序可以看到下面的输出，表示多余的字符被截断了。

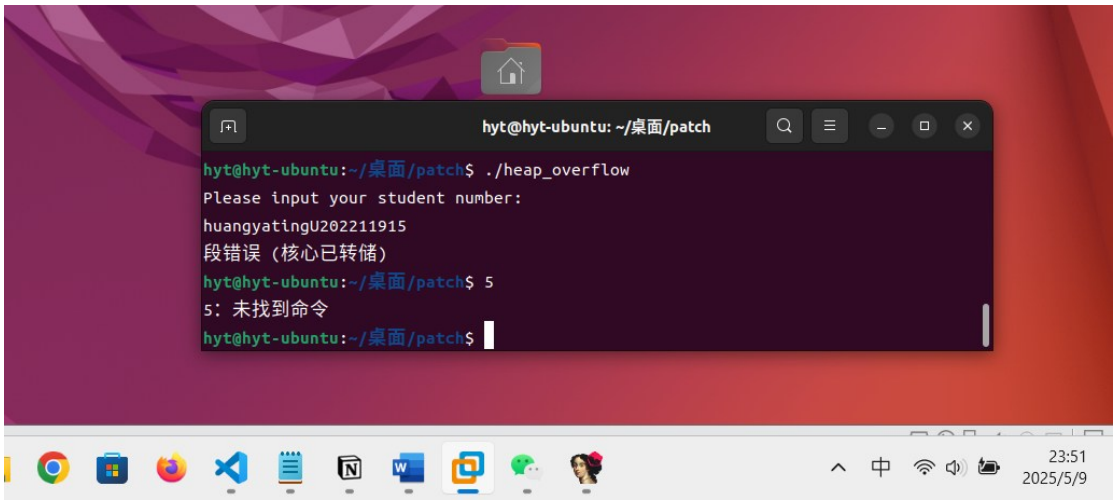

图 2-9 第二个补丁原始输出

这里可能导致漏洞，即栈溢出的漏洞。为了防止栈溢出漏洞，需要修改 EDX 寄存器的赋值。按照任务书的指示将 EDX 的值改为 0x10，如图 2-11 所示修补后的程序仅打印与自己性别相对应的话，且无论输入多长的字符串均不触发堆溢出漏洞，从而提高了程序的安全性。

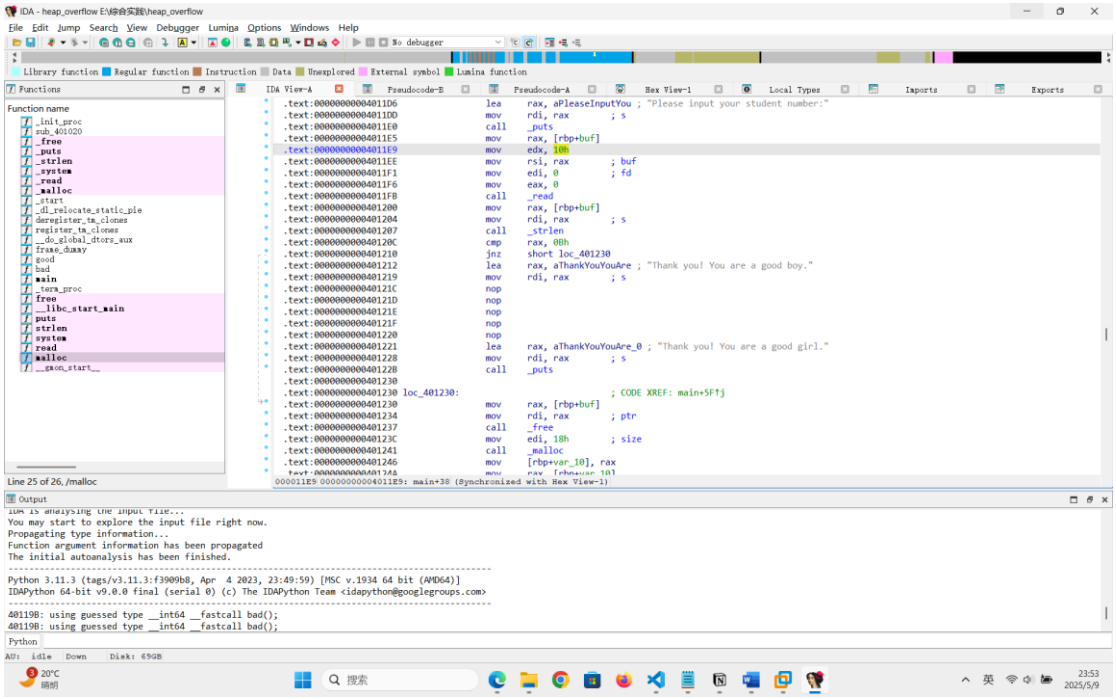

图 2-10 修改第二个补丁

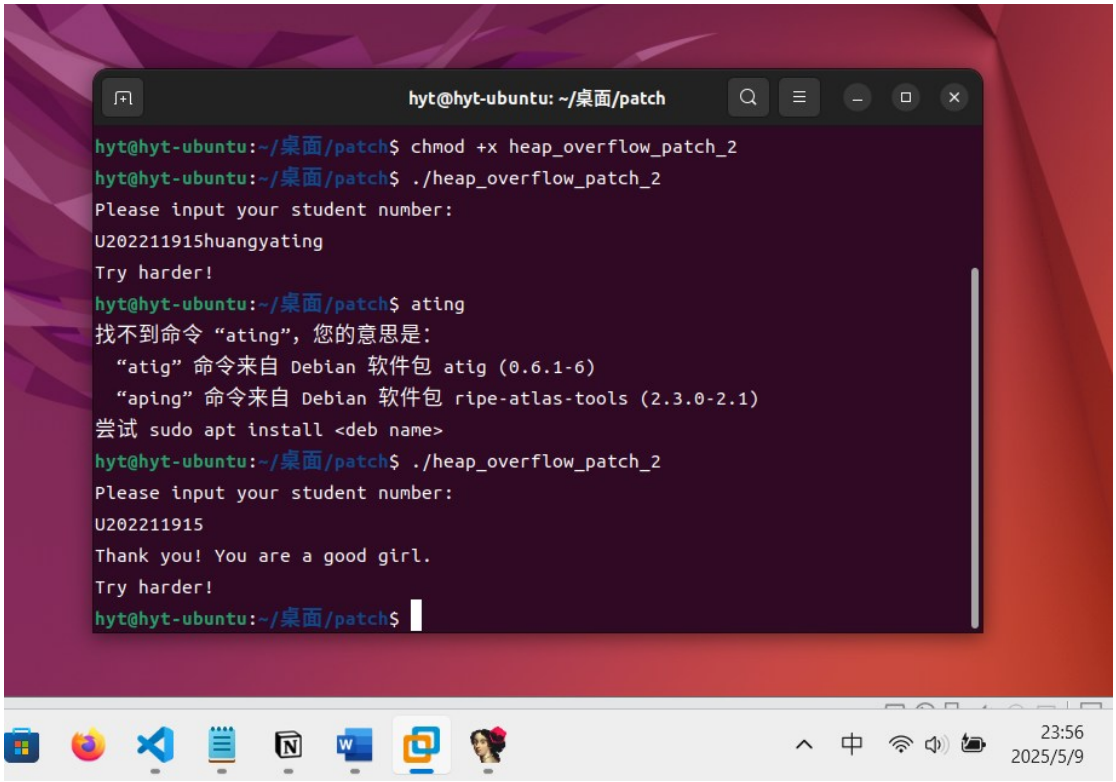

图 2-11 第二个补丁成功

2.3 附加任务分析

在附加任务中，我们考虑了通过一个栈溢出漏洞导航到 good 函数。good 函数的地址和功能在 IDA 中进行了解析，如下图 2-12 所示，得知它能执行命令。恢复 EDX 至 0x14，可使溢出达到 good 函数的位置。

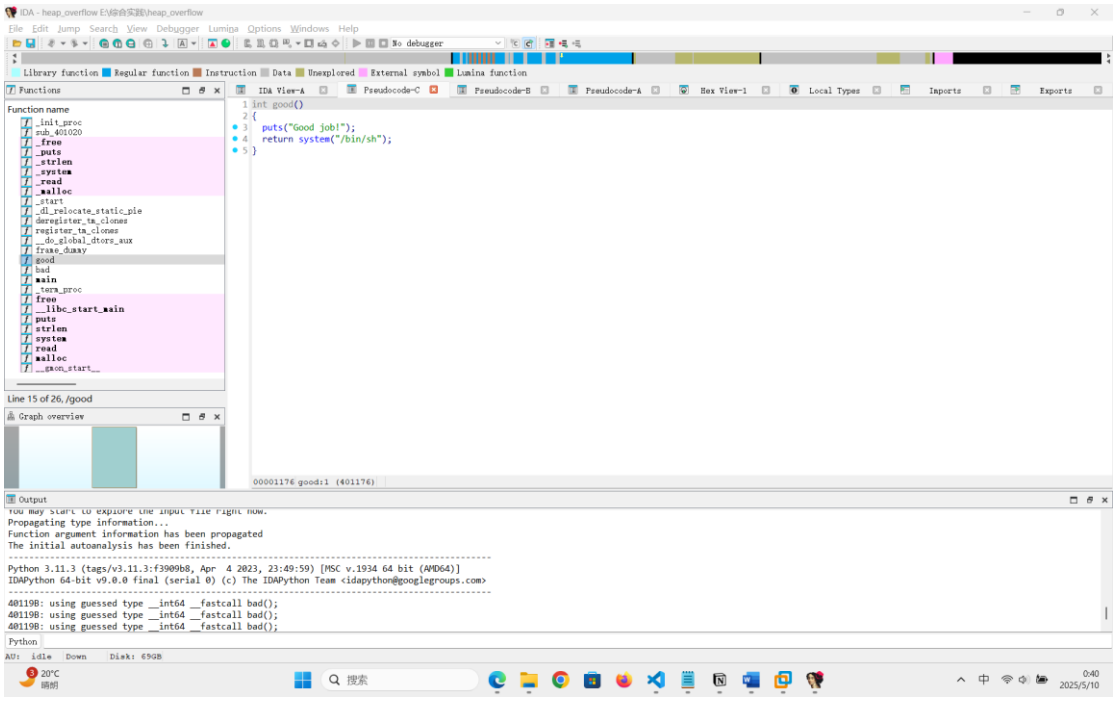

图 2-12 good 函数

我们利用已提供的 exp.py 脚本，并进行了轻微修改以用于调试，这能更直观地展示溢出的地址和效果。代码如下图所示：

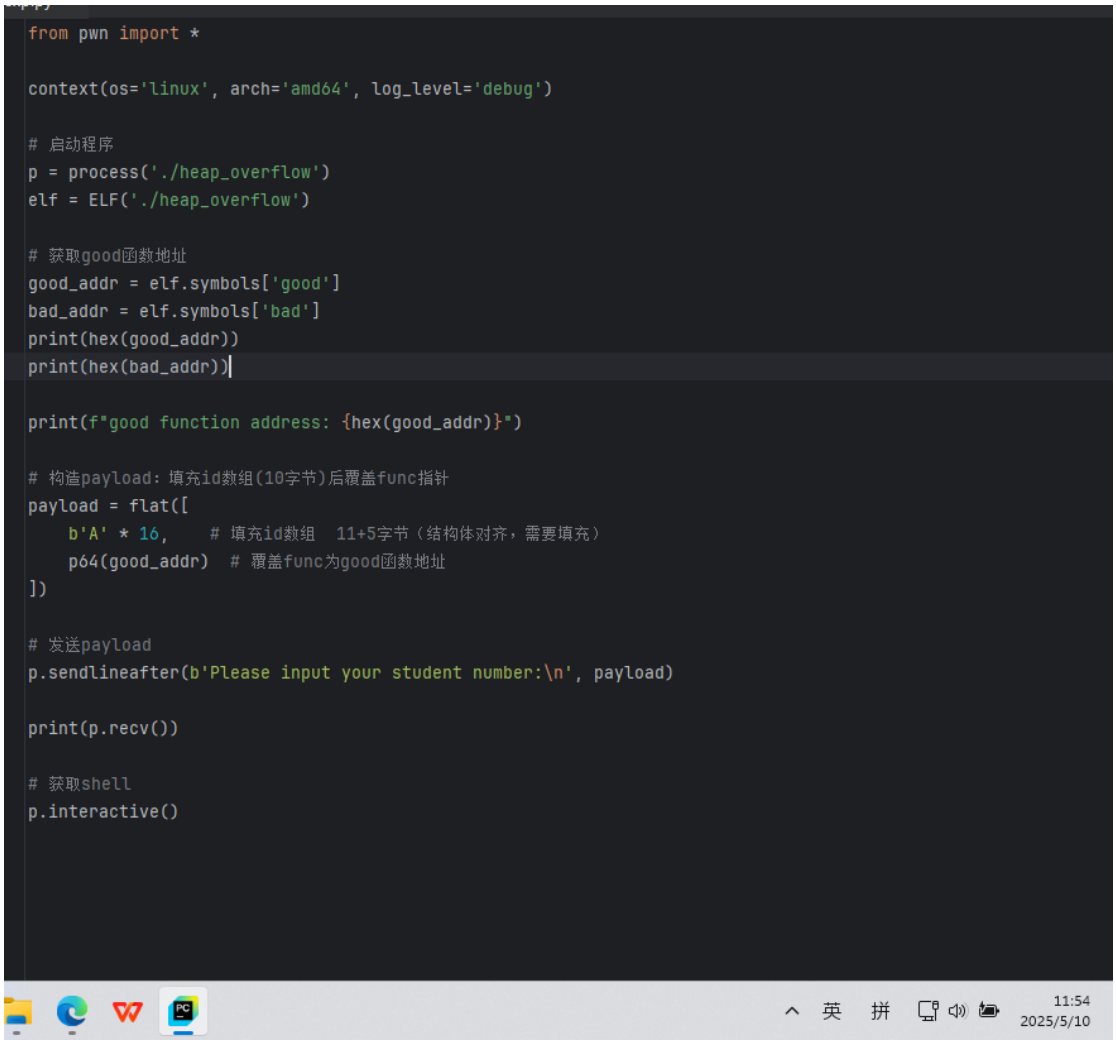

图 2-13 exp.py 脚本代码

执行改进后的脚本，可以看到程序成功执行了预期的命令，并输出了"good job"，表示破解成功。如下图所示。

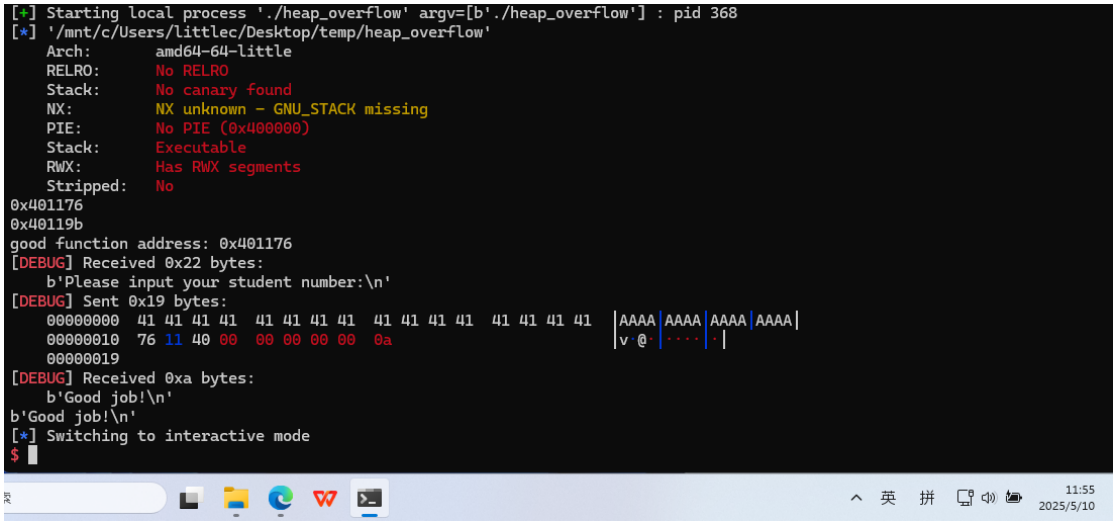

图 2-14 附加任务完成 1

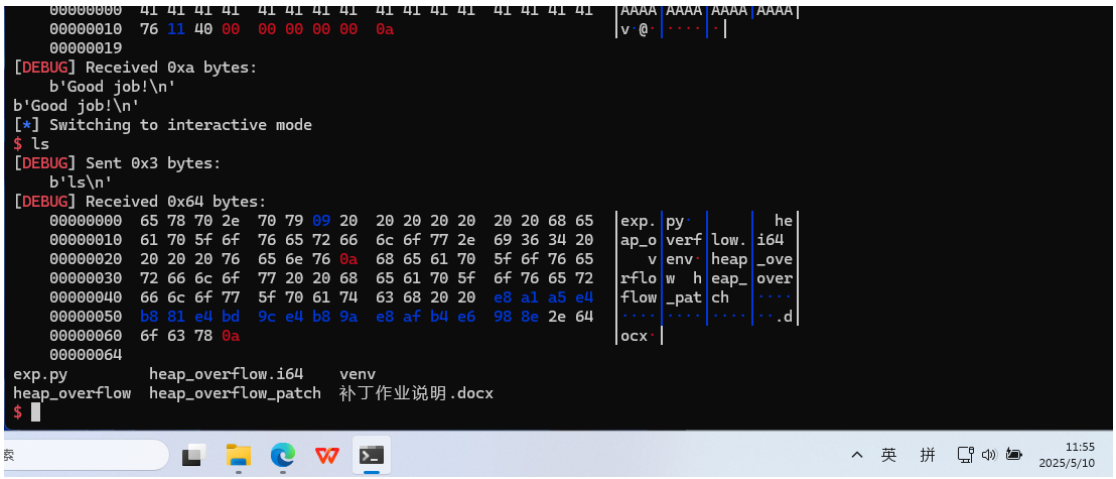

图 2-15 附加任务完成 2

综上已完成实验全部要求。通过本次实验，不仅对二进制文件中的安全补丁概念有了更深刻的理解，而且学会了如何实际应用这些补丁来修复和编辑二进制文件。本次实验整合了理论学习与实践操作，提升了实际操作能力和安全意识。