编号：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |

武汉大学国家网络安全学院

课程实验(设计)报告

题 目： 最小PE文件

专业(班)： 信安5班

学 号： 2022302181161

姓 名： 王亚鹏

课程名称： 软件安全

任课教师： 赵磊

2024年 11 月 20日

**目录**

[堆溢出实验 1](#_Toc183095534)

[1实验名称 1](#_Toc183095535)

[2实验原理 1](#_Toc183095536)

[3实验步骤 2](#_Toc183095537)

[4 实验结果 5](#_Toc183095538)

[5实验心得 6](#_Toc183095539)

## 堆溢出实验

### 1实验名称

**PE文件最小化**

### 2实验原理

**主要原理**：通过**精简**PE**文件中的冗余结构和优化对齐方式**，以实现文件最小化

**1. 对齐粒度的调整**

PE文件中，文件偏移和内存偏移的对齐通常会填充大量的零，增加了文件的大小。通过调整对齐粒度（例如从默认的512字节调整为更小的4字节），使文件偏移和内存偏移一致，从而减少对齐填充的冗余字节。

* **过程**：

修改内存和文件对齐参数。

手动更改入口地址及相关位置（确保逻辑地址映射正确）。

删除多余的填充部分。

**2. 去除无效部分**

PE文件中包含一些对程序运行无直接作用的部分，例如DOS STUB（旧版MS-DOS兼容代码）。实验中通过以下方式进一步压缩文件：

* **删除无用区域**：

将怀疑无用的区域先置为1，验证程序是否还能运行。

确定无用部分后直接删除。

删除后导致的地址偏移变化需要在文件头中进行调整。

**3. 利用重叠结构优化**

PE文件的结构中，有些区域可以通过**重叠布局**优化空间利用率：

* **DOS头和PE头的重叠**：

修改DOS头中指向PE头的地址（03Ch字段）。

重新调整可选头中的重要字段，如程序入口地址（RVA）和导入表位置。

* **插入其他数据到未使用区域**：

将DLL名称、导入表等重定位到PE头的空闲区域。

通过修改引入地址表和引入名字表中的指针，确保重叠后的结构仍然正确

**4. 重新调整节头与代码节**

PE文件中不同节的分布也可以进一步优化：

* **代码节头的合并**：

将代码节挪到靠近节头表的区域，避免中间出现空隙。

修改代码节和数据节的RVA和文件偏移地址，确保与PE头的映射一致。

* **导入表的优化**：

合并引入地址表（IAT）和引入名字表（INT），减少冗余空间。

调整user32.dll和kernel32.dll的位置，使其更加紧凑。

将重要函数（如MessageBoxA和ExitProcess）重新映射到未被使用的区域。

**5. 删除多余数据和恢复必要字符串**

在压缩文件的最后一步，对文件进行清理和调整：

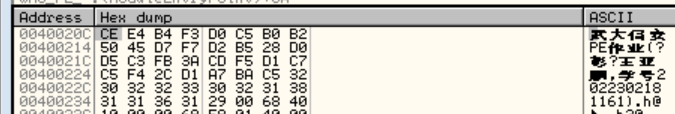
* **删除无效数据**：将之前置1且没有被使用的区域清理
* **恢复必要的字符串和数据**：将这些内容移动到代码节之后，同时确保所有相关指针地址都经过更新

### 3实验步骤

1. **填充自己的姓名学号**

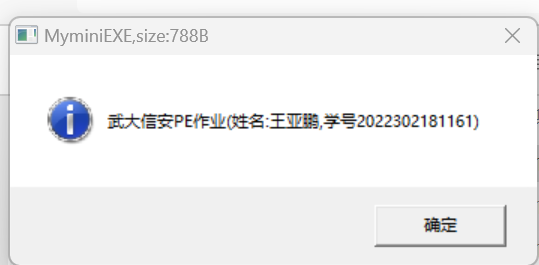
使用ollydbg打开文件

找到数据区，左键点击之后follow in dump，之后进行二进制编辑



如图所示，填充完成

保存之后运行结果如下：



由于其余部分都没有进行更改，这时候文件的大小没有被改变

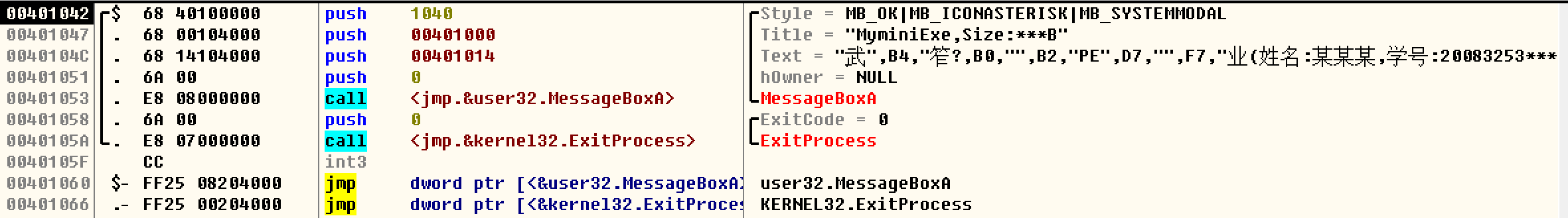
1. **修改对齐粒度**

由于对齐填充了很多0，因此可以将内存和文件的对齐粒度改为4，使内存偏移和文件偏移一致，因此有很多入口地址需要更改

可以使用PE\_View看到程序的入口处如下：



对应文件偏移是242，这是因为要减去那个1000h，这样就可以找到代码执行开始的位置，注意是16进制

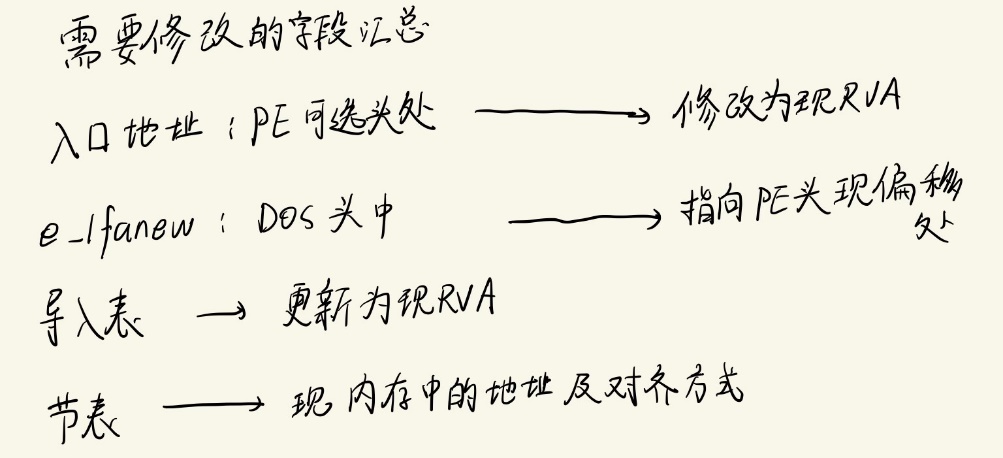


通过这个可以看到401000->title; 401014->text

402008->MessageBoxA; 402000->ExitProcess

修改对齐粒度之后，这些不能继续代表位置，需要我们手动算出这个时候的有效地址

需要知道的是此处文件偏移是100，内存偏移是2000，可以根据此以及新的对齐算出新的位置入口



1. **去除无用部分**

PE文件由以下几个主要部分组成：

1. DOS头 2.DOS存根 3.PE文件头

4.可选头 5.节头表 6.节数据

**无用部分**

**1. DOS存根**

这是为了兼容MS-DOS系统设计的，通常是一个提示信息，比如“程序不能在DOS模式下运行”。

直接删除该部分。此区域删除后，会导致后续所有结构的地址上移，因此需要重新调整e\_lfanew（DOS头中的字段，指向PE头位置）指向的新地址

**2. 对齐填充的无效数据**

PE文件对齐填充是为了满足内存和文件对齐的需求。例如默认对齐粒度为512字节，很多节和段之间会填充无效的0。

将文件对齐粒度调整为更小的值

删除节与节之间多余的0填充，减小文件大小

**3. 节表中未使用的节**

节表中可能存在未被实际加载或使用的节，例如：

数据段（.data）中未用到的变量部分。

资源段（.rsrc）中无效的资源数据。

**实验处理**：

删除这些无用的节，重新计算节表的大小和RVA偏移

**4. 导入表中的冗余部分**

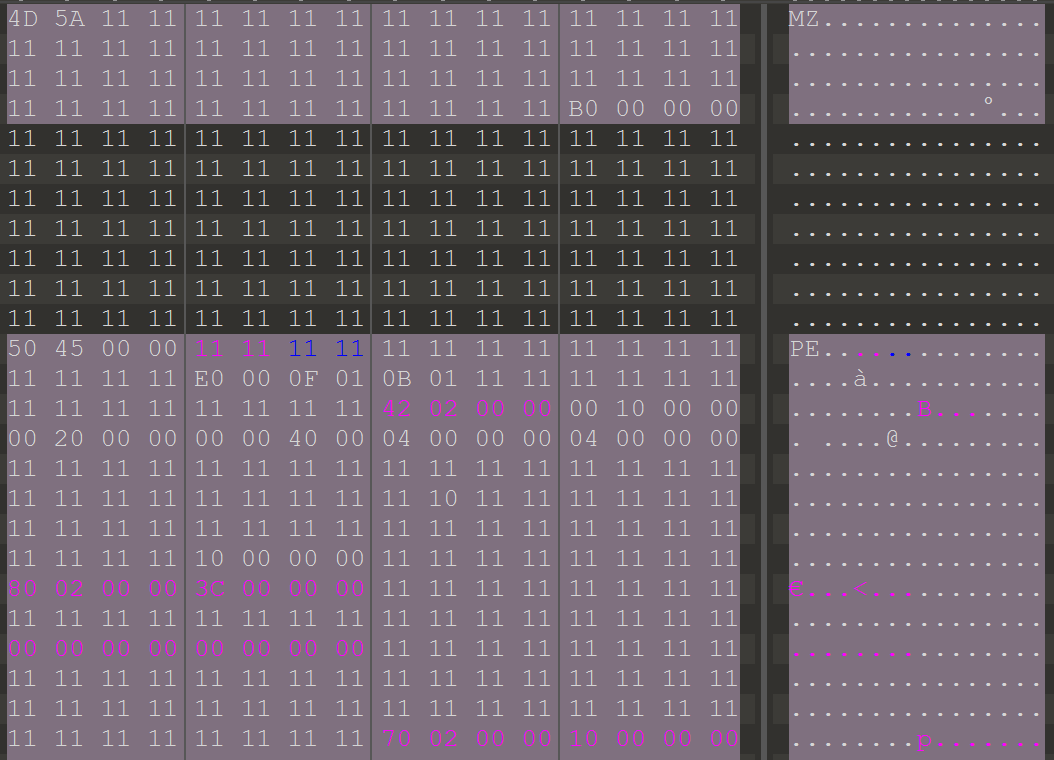
导入表列出了程序所需的外部库函数。有时导入表中包含了程序实际未用到的导入项

进行如下操作：

删除未使用的导入项

合并引入地址表（IAT）和引入名字表（INT），节省空间

调整导入表的大小和指针



这张图表示的是无用部分已经用1填充（保存后仍可以正常运行该PE文件，表明无误）

**（4）重叠结构的优化**

经过视频讲解，了解到PE头可以和DOS头重叠，从而少一个头结构，但是还能够正常运行

**DOS头**

**位置**：PE文件最开始的部分。

**大小**：固定为64字节。

**作用**：提供兼容性，主要用于老的MS-DOS环境，现代Windows系统通常忽略其大部分内容。

**关键字段**：

e\_magic：DOS头标志，值为MZ。

e\_lfanew：指向PE头的偏移地址，是连接DOS头与PE头的关键。

**PE头**

**位置**：通过e\_lfanew指针连接到DOS头。

**大小**：由节头表（Section Table）的数量决定，通常在200字节左右。

**作用**：描述PE文件的核心信息，包括节表的位置、入口点地址、对齐信息等。

通常情况下有：

DOS头在前，PE头通过DOS头的e\_lfanew字段指针定位。

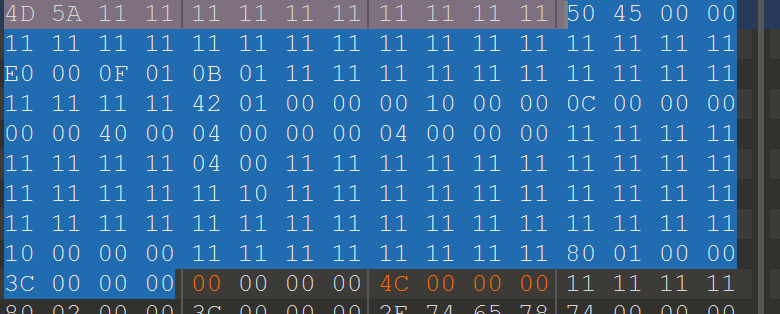
常规情况下，DOS头和PE头不重叠，DOS头独占64字节

但是DOS头的大部分字段对现代Windows环境下的PE文件加载来说是无用的，仅以下字段必须有效：

**e\_magic**：必须是MZ，用于标识PE文件。

**e\_lfanew**：指向PE头的偏移地址，是加载PE头的关键。

其余字段通常是DOS存根代码，可以被替换或删掉。因此，我们可以将PE头重叠到DOS头未使用的区域



步骤如下：

1. **识别无用的DOS头部分**：

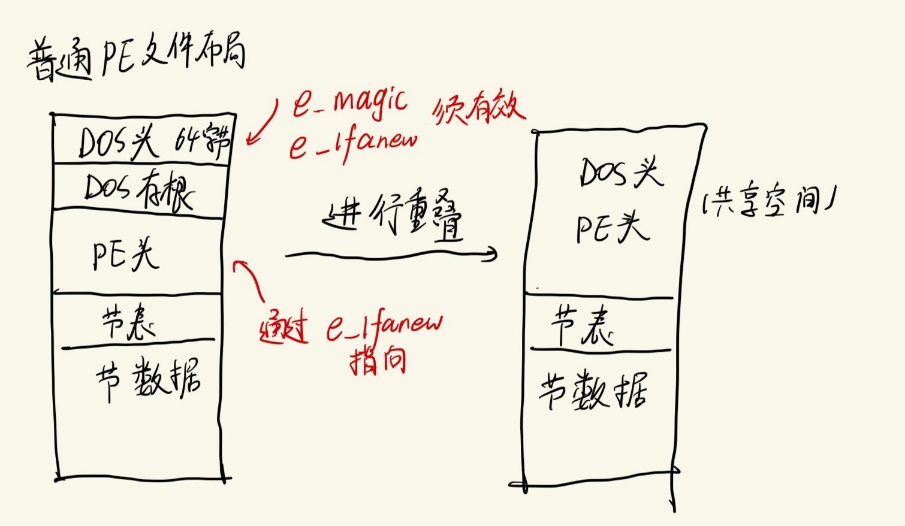
e\_magic（MZ标志）和e\_lfanew（指向PE头的偏移）必须保留。

其他字段，如DOS存根代码，可以被移除或重写。

1. **移动PE头到DOS头空间**：

将PE头的内容直接写入DOS头的未使用区域，避免浪费空间。

1. **更新e\_lfanew字段**：



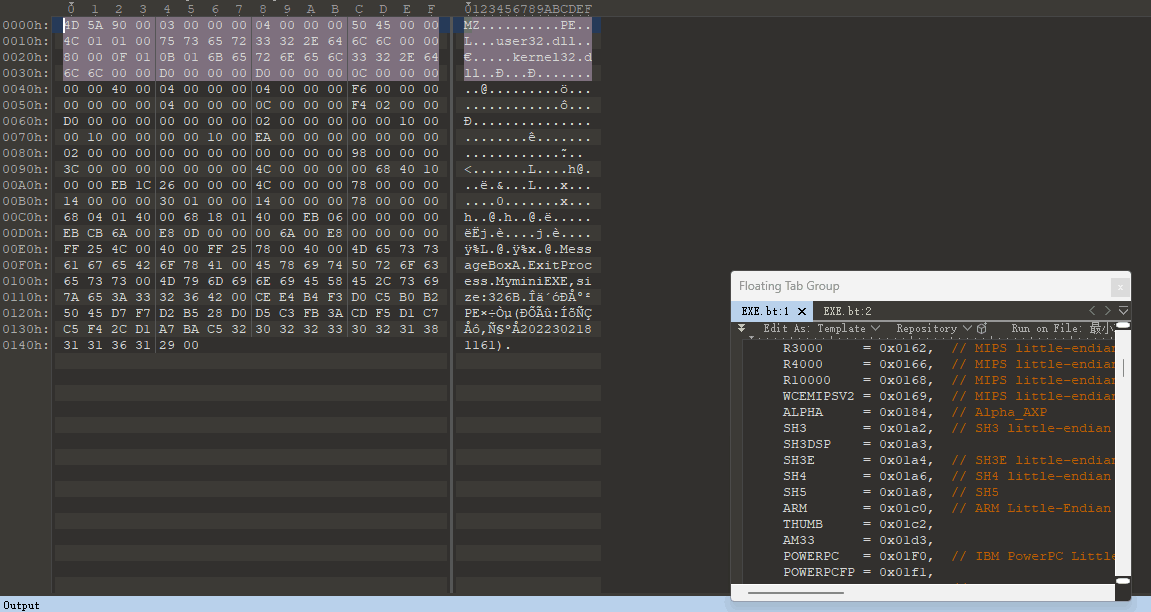
修改前后的结构

因为PE头现在与DOS头重叠，所以需要调整e\_lfanew字段，指向PE头的新位置。

通常为DOS头的起始地址（0x40），即e\_lfanew字段的值为固定的16进制地址

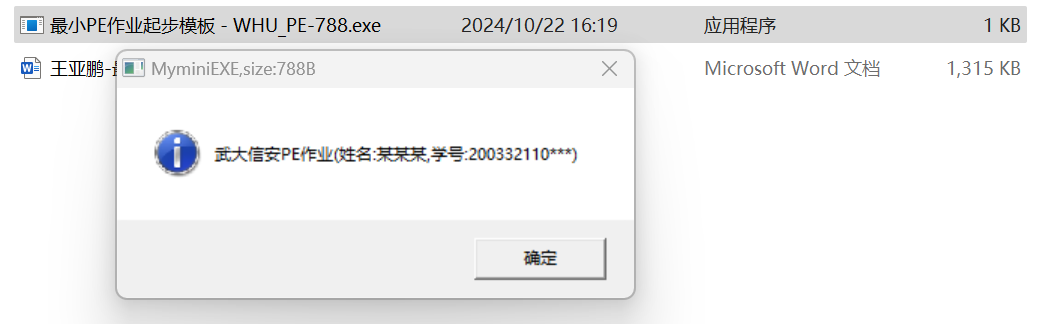
最后将不必要的部分删去，之前置1且没有被占据的部分恢复正常值，其余的字符串放到代码之后，并且调整相应的地址

最终经过修改后的PE文件如下：

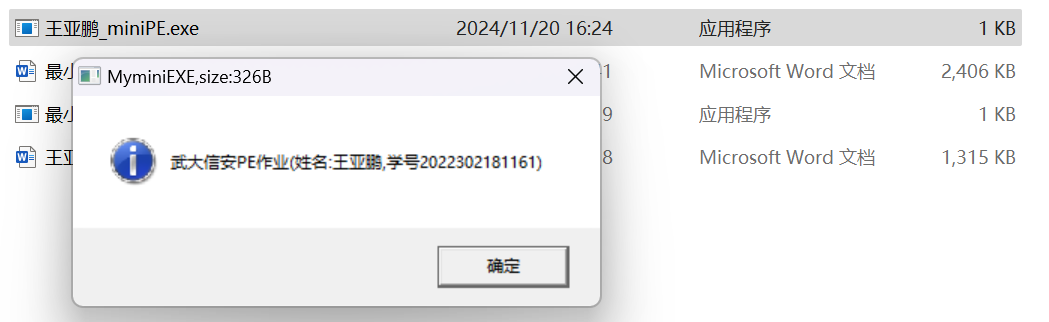


### 4 实验结果

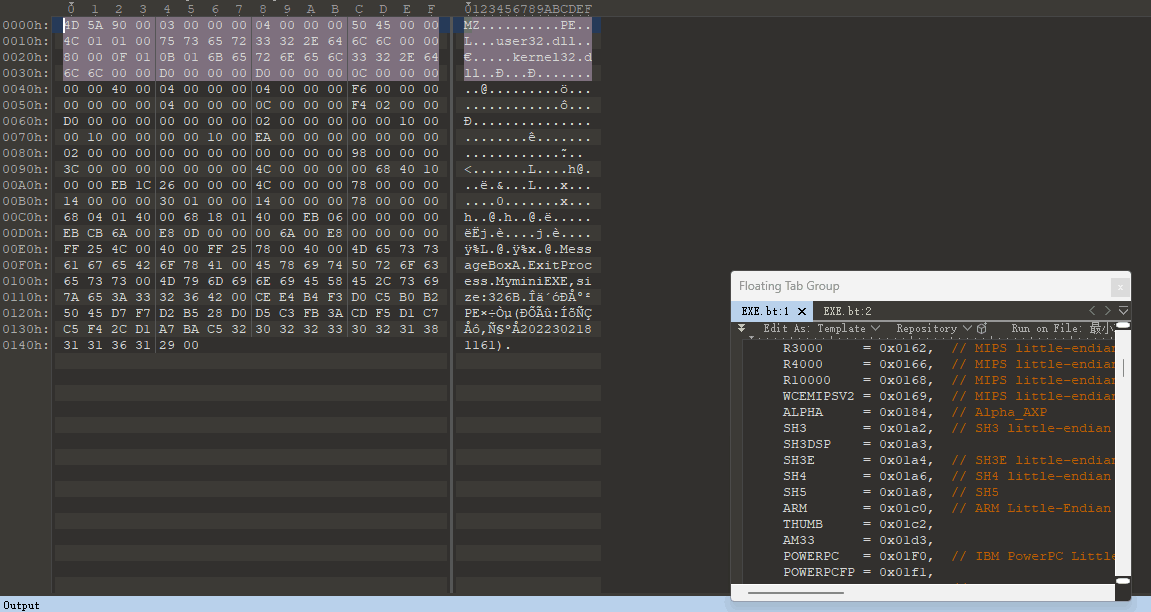
**（1）**这是没有经过任何更改的原始PE文件，可以看到大小是788B，姓名和学号还处于代填充状态



**（2）**经过更改之后的PE文件执行结果如下：姓名学号都已经填好，大小是326B，可以从010editor中看到



如下图所示，可以数出一共只有326B



16进制140，是256加上64，再加上最后的6个字节，总共是326B

### 5实验心得

**1. 对PE文件结构的理解**

在实验中，通过对PE文件的分析，更加清楚地认识到PE文件的核心结构，包括：

DOS头：虽然是为兼容MS-DOS设计的，但其大部分内容在现代操作系统中并非必需。

PE头和可选头：包含重要的元数据信息，例如入口点、对齐方式、导入表地址等。

节表和数据段：程序的实际代码和数据存储区。

**2. 优化与压缩方法的认识**

对文件优化有了更全面的认识，之前一直不理解为什么文件被压缩之后还能够正常运行、解压缩等

**对齐粒度的调整：**通过将默认的对齐粒度从512字节降低到4字节，大幅减少了对齐填充的冗余数据。这启发我在优化文件或程序结构时，可以从减少浪费入手。

**删除无效部分：**删除如DOS STUB这类没有实际作用的部分时，我意识到，很多历史遗留的结构在特定场景下是可以安全去除的，但必须确保程序的功能性不受影响。

**重叠结构的使用：**将DOS头与PE头等区域进行重叠，是一次非常有趣的尝试。通过这一方法，文件中的空闲区域得到了充分利用

**3. 对程序运行机制的进一步理解**

在实验中，我们对PE文件中各种地址进行了大量修改，例如：

（1）调整了程序入口地址（RVA）和导入表的位置。

（2）修改了内存和文件偏移的映射关系。

深刻理解了程序加载运行的机制：文件在磁盘上存储的形式与加载到内存中的布局并不一致，文件的各种指针和地址需要相互配合，才能保证程序正常运行

也认识到在修改二进制文件时，每一处地址的变化都可能对程序运行产生影响