typedef struct \_IMAGE\_SECTION\_HEADER {

BYTE Name[8]; *// 节区名称（如 ".idata"）*

DWORD VirtualSize; *// 内存中节区实际大小（可能未对齐）*

DWORD VirtualAddress; *// 内存中的 RVA（关键！用于计算）*

DWORD SizeOfRawData; *// 文件中节区大小（对齐后）*

DWORD PointerToRawData; *// 文件中的偏移（关键！用于计算）*

DWORD PointerToRelocations; *// 重定位表偏移（无用，除非是OBJ文件）*

DWORD PointerToLinenumbers; *// 调试信息（通常为0）*

WORD NumberOfRelocations; *// 重定位项数（无用）*

WORD NumberOfLinenumbers; *// 调试信息（通常为0）*

DWORD Characteristics; *// 节区属性（如可读/可写）*

} IMAGE\_SECTION\_HEADER;

节区头信息如上。每个节区头固定40字节，属于默认信息。而查看节区大小可以根据节区VirtualSize和SizeOfRawData确定。

VirtualSize表示实际大小，是该节区在内存中占用的实际大小，SizeOfRawData表示文件大小，是该节区在磁盘文件中占用的对齐后的大小。节区在内存中对齐会填充无效数据，使节区长度变长，例如在某文件中：VirtualSize = 0x1A44（6,724 字节），SizeOfRawData = 0x1C00（7,168 字节），说明该节区在文件中占用 7,168 字节（对齐后），但实际有效数据只有 6,724 字节，多出的 0x1C00 - 0x1A44 = 0x1BC（444 字节）可能是对齐填充（通常是 0x00）。

节区（磁盘）

节区（内存）

VirtualAddress和PointerToRawData规定节区起始位置。 VirtualAddress表示内存中的节区起始地址，是定义该节区加载到内存后的起始相对虚拟地址（RVA）。PointerToRawData表示文件中的节区起始偏移，是定义该节区在磁盘文件中的起始物理偏移。而起始位置还与NT头IMAGE\_OPTIONAL\_HEADER中的Section Alinment和File Alinment有一些关系。Section Alignment规定了节区在**内存**中的**起始地址**必须对齐的粒度，File Alinment规定了文件在**磁盘**上存储时每个节区**起始地址**的对齐粒度。换句话说，VirtualAddress和PointerToRawData必须分别要满足是Section Alinment和File Alinment的倍数关系，当需要手动修改PE结构时则需要考虑该值对VirtualAddress和PointerToRawData的影响。

另外一个与节区相关的量还有NT头IMAGE\_FILE\_HEADER中的Number of Sections，它记录的是一个PE文件中的节区数量。

扩展

在手动修改节区时，需要修改一个值：NT头IMAGE\_OPTIONAL\_HEADER 的 Size of Image。Size of Image是整个 PE 文件加载到内存占用的虚拟地址空间大小，所以这里将讨论在整个PE文件中节区是如何排列的。

Size of Image并不是每个节区单独对齐后相加的结果，而是最后一个节区结束后的总对齐。每个节区均需要按照Section Alinment的制约完成对齐，但唯独最后一个节区不同在末尾需要按照“下一个”节区的开头补齐0完成对齐。

计算方法：

Size of Image = 最后一个节区的结束地址（VirtualAddress + VirtualSize）向上对齐到 Section Alignment