内存、栈、堆

在理解计算机程序的内存管理时，关键概念包括栈、堆、可执行文件映像和保留区。

栈（Stack）

栈是由操作系统自动管理的内存区域，主要用于存储函数调用的上下文，包括：

- **函数的返回地址和参数**：控制程序流程。
- **临时变量**：主要包括函数内的非静态局部变量及编译器生成的临时变量。
- **保存上下文**：在函数调用过程中，需要保存和恢复某些寄存器的值，以保持执行状态。

栈的特点是先进后出（LIFO），且其操作速度一般比堆快，但大小有限且由系统预设。

堆（Heap）

堆是用于动态内存分配的区域，通常由程序员手动分配和释放（或由垃圾回收机制处理），特点包括：

- **动态内存分配**：适用于程序运行时才知道所需内存大小的情况。
- **分配算法**：包括空闲链表（Free List）、位图（Bitmap）和对象池等，这些方法各有优劣，应根据具体需求选择。

堆的大小不固定，可动态扩展，但其管理成本高于栈，且易产生内存碎片。

可执行文件映像

这是内存中的一个区域，用于存储可执行文件的内存映像。当程序启动时，操作系统的装载器会将可执行文件的内容读取或映射到这个区域。

保留区

保留区并不是指单一的内存区域，而是泛指被操作系统或硬件保护的、不能被普通程序访问的内存区域。例如，C语言中通常将无效指针赋值为NULL，而NULL通常对应的0地址是不可访问的。

常见错误：段错误（Segmentation Fault）

当程序尝试访问它没有权限访问的内存区域时，就会出现段错误。这通常是由非法指针引用导致的，例如：

- **空指针解引用**：未初始化的指针被错误地用于访问数据。
- **野指针**：指针指向了不合法的内存地址，如未初始化或已释放的内存。
- **栈溢出**：函数调用过深，超出了栈的容量限制。

编译链接

编译链接过程是程序从源代码转换为可执行文件的关键步骤，涉及多个平台和文件格式。

各平台文件格式

不同的操作系统支持不同的文件格式，下表总结了主流平台的主要文件类型：

平台	可执行文件	目标文件	动态库/共享对象	静态库
Windows	`.exe`	`.obj`	`.dll`	`.lib`
Unix/Linux	`ELF`、`out`	`.o`	`.so`	`.a`
Mac	`Mach-O`	`.o`	`.dylib`、`.tbd`、`.framework`	`.a`、`.framework`

编译链接过程

1. **预编译**：预处理器处理宏定义、文件包含等指令，生成 `.i` 或 `.ii` 文件。
2. **编译**：将预处理后的源代码转换为汇编语言，生成 `.s` 文件。
3. **汇编**：将汇编代码转换为机器码，生成 `.o`（目标文件）。
4. **链接**：将多个目标文件合并，解决引用问题，生成可执行文件（`.out`、`.exe` 等）。

> GCC 合并了预编译和编译步骤，包含预编译编译程序 `cc1`、汇编器 `as` 和连接器 `ld`。
> MSVC 环境中，包含编译器 `cl`、连接器 `link` 和可执行文件查看器 `dumpbin`。

目标文件

目标文件是编译后但未链接的文件，已转换为可执行文件格式但可能含未调整的符号或地址。

> 可执行文件（Windows 的 `.exe` 和 Linux 的 `ELF`）、动态链接库（Windows 的 `.dll` 和 Linux 的 `.so`）、静态链接库（Windows 的 `.lib` 和 Linux 的 `.a`）均采用特定的可执行文件格式（Windows 使用 PE-COFF，Linux 使用 ELF）。

目标文件格式

- **Windows**：PE（Portable Executable）或 PE-COFF，`.obj` 格式。
- **Linux**：ELF（Executable and Linkable Format），`.o` 格式。
- **其他格式**：Intel/Microsoft 的 OMF（Object Module Format），Unix 的 `a.out`，MS-DOS 的 `.COM`。

> PE 和 ELF 都是基于 COFF（Common File Format）的变种。

目标文件存储结构

目标文件通常包含以下主要段：

- **File Header**：描述文件属性，如可执行性、链接类型、入口地址等。
- **.text section**：包含编译后的机器代码。
- **.data section**：已初始化的全局和静态变量。
- **.bss section**：未初始化的全局和静态变量。
- **.rodata section**：只读数据，如常量和字符串。
- **.comment section**：编译器版本信息。
- **.note.GNU-stack section**：堆栈提示信息。

链接中的符号（Symbol）

链接过程主要涉及对符号的解析和地址分配，符号表（Symbol Table）记录了符号名和其对应地址。

Symbol（符号名）	Symbol Value （地址）
main	0x100
Add	0x123
…	…