目录

[1. 内存分区模型 3](#_Toc127620663)

[代码区 3](#_Toc127620664)

[全局区 3](#_Toc127620665)

[栈区 3](#_Toc127620666)

[堆区 4](#_Toc127620667)

1. 内存分区模型

不同区域存放的数据，赋予不同的生命周期给我们更大的灵活编程

在程序编译后，生成exe可执行程序，未执行该程序前分为两个区域，代码区和全局区

代码区

存放函数体的二进制代码，由操作系统进行管理

存放CPU执行的机器指令

代码区是共享的，共享的目的是对于频繁被执行的程序，只需要在内存中有一份代码即可

代码区是只读的，使其只读的原因是防止程序意外的修改了它的指令

全局区

存放全局变量和静态变量（static）以及常量

全局变量和静态变量存放在此

全局区还包含了常量区，字符常量和其它常量也存在此

该区域的数据在程序结束后由操作系统回收

|  |  |
| --- | --- |
| 不在全局区中 | 全局区 |
| 局部变量  const修饰的局部变量（局部常量） | 全局变量  静态变量（static关键字）  字符串常量  const修饰的全局变量（全局常量） |

栈区

由编译器自动分配释放，存放函数的参数值（形参），局部变量等

注意：不要返回局部变量的地址，会由编译器自动释放

堆区

由程序员分配和释放，若程序员不释放，程序结束时由操作系统回收

主要利用new在堆区开辟内存，释放内存使用delete

释放数组的时候，要加 [ ] （delete[] arr;）

|  |  |
| --- | --- |
| 栈区 | 堆区 |
| int \* func() {  int a = 10;  return &a;  } | int \* func() {  int \*p = new int(10);  int \*arr = new int[10];  return p;  }  int main() {  int \*p = func();  delete p;  } |
| 使用函数返回的地址时非法操作 | 返回堆区存放数据的地址  释放后再调用为非法操作 |