这种软件设计的思想，就是所谓的"分层"

Java的这种分层设计带来了什么好处?
1.跨平台性。
依赖于不同平台上的不同JVM Java应用实现了跨平台性

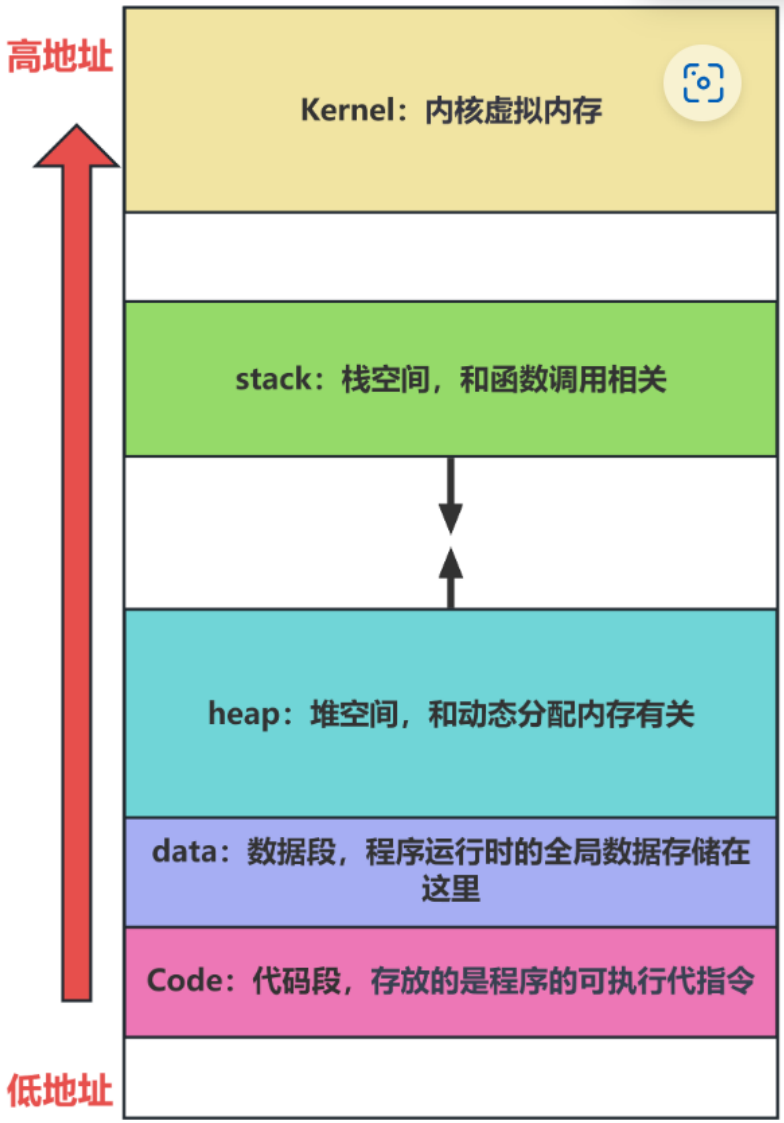
2.Java应用可以把很多原本是自己做的事情，交给JVM
比如最典型的就是 内存管理
所以Java应用的安全性和稳定性就得以提升

那么缺点是什么?

1.效率降低了。

2.功能降低了。

没有十全十美的编程语言



1. 代码段
它和程序员关系不大 用来存储程序的指令

2.数据段
用来存储C程序中的运行时全局数据
全局变量
静态变量（static修饰的局部变量和全局变量）
数据段的只读段用来存储C程序中的只读数据
比如字符串字面值，它是只读不可变的

3，堆空间
涉及到C程序的动态内存分配和管理
C程序员管理的内存就是堆内存

堆空间是可以生长的(扩容)的
而且是从低地址向高地址生长

4.栈空间
栈和C程序的函数调用关联十分紧密
栈空间用于存储C语言中的局部变量
栈空间实际上是模拟了数据结构中的栈的行为来设计的
它是"先进后出"的

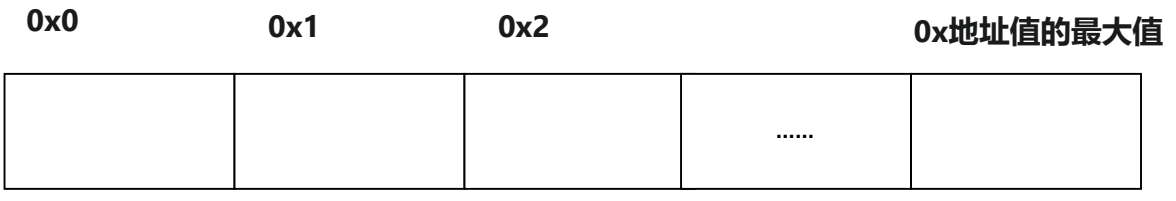
堆空间占据虚拟内存绝大多数空间，而且可以扩容
但是栈空间虽然也可以扩容，但容量一般非常少

5.内核区域
用户态 --> 内核态

操作系统寻址处理数据的最小单位是8位1个字节
所以内存地址指的是：
虚拟内存空间当中某1个字节的区域的唯一标识。

变量的地址：
1个int变量，它一般占用4个字节(虚拟内存空间的4个字节)
变量所占用内存空间的第一个字节的内存区域的唯一标识。

我们可以把虚拟内存想象成一个连续的数组，内存地址是唯一标识，可以看成是数组的下标



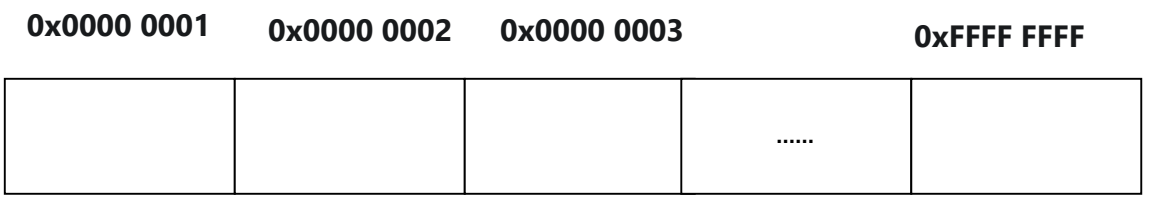
虚拟内存空间的地址(值)
在区间
[0, 最大地址值]

低地址： 靠近内存空间中地址最小的区域，就是低地址
高地址： 靠近内存空间中地址最大堆区域，就是高地址

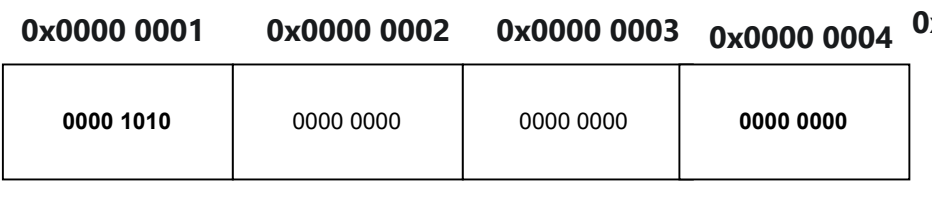
对于某个平台的虚拟内存空间来说，它的最大地址值是如何确定的?
要根据平台的架构确定
32位的平台
地址总线是32位，意味着虚拟内存空间的内存地址有2^32种可能性
于是该地址的最大值就是[0, 2^32-1]
我们一般都习惯于用16进制表示地址值
[0x0000 0000, 0xFFFF FFFF]
其中，区间范围内的每一个十六进制数都恰好代表一个字节的内存区域

64位的平台
可寻址空间更大，内存地址值更大

32位平台的虚拟内存空间模型



现在有1个int变量假如是：
int a = 10;
那么它在虚拟内存空间中是如何存储的呢?
在内存中，整数是以有符号数的补码形式存储的
正数的原反补码是一样的
补码：
0000 0000 0000 0000 0000 0000 0000 1010



小端存储法：
指的是在Intel/AMD等现代计算机架构中
存储某个数据时
选择将此数据的低有效位数，存储在虚拟内存空间的低地址上
所以10这个整数在虚拟内存空间中的存储是：
从低地址到高地址
0000 1010 0000000000000000...

大端存储法
将此数据的高有效位放在低地址上

大端存储法在网络传输过程中使用

变量: 在程序中具有改变可能性的一个数据项.
如何理解一个变量?
从内存的角度来说,变量就是内存中的一片存储区域,它有大小,它有能够存储的数据的类型...
从变量的使用角度理解(推荐):
1.一个变量是可以存储数据的,而且这个数据必须是明确的,这就引出了变量的数据类型的概念
2.变量的概念提出来,目的是为了方便程序员使用的.所以变量还需要一个名字,这就是变量名,它是一个标识符
3.最后还需要关注变量的具体取值是多少,也就是变量的取值,一个明确取值的变量才是可以使用的变量

这就是变量的三要素:
1.数据类型
2.变量名
3.取值

变量的数据类型是什么呢?
内存角度: 变量的存储空间,究竟长什么样子,占多么内存,存什么数据
具体使用的角度理解:
数据的集合 + 合法操作的集合

int类型: 表示整数,一般占用4个字节,默认有符号数,取值范围是[-2^31, 2^31-1] 大概是正负21亿多

float类型: 表示单精度浮点数,一般也占用4个字节的,遵循ieee754标准