C 代码优化（3） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/449/2021_2022_C___E4_BB_A3_E7_A0_81_E4_c97_449187.htm 结构体成员的布局 很多编译器有“ 使结构体字，双字或四字对齐 ”的选项。但是，还是需要改善结构体成员的对齐，有些编译器可能分配给结构体成员空间的顺序与他们声明的不同。但是，有些编译器并不提供这些功能，或者效果不好。所以，要在付出最少代价的情况下实现最好的结构体和结构体成员对齐，建议采取这些方法： 按类型长度排序 把结构体的成员按照它们的类型长度排序，声明成员时把长的类型放在短的前面。 把结构体填充成最长类型长度的整倍数 把结构体填充成最长类型长度的整倍数。照这样，如果结构体的第一个成员对齐了，所有整个结构体自然也就对齐了。下面的例子演示了如何对结构体成员进行重新排序： 不好的代码，普通顺序 推荐的代码，新的顺序并手动填充了几个字节 struct { char a[5]； long k； double x； } baz； struct { double x； long k； char a[5] ； char pad[7] ； } baz； 这个规则同样适用于类的成员的布局。 按数据类型的长度排序本地变量 当编译器分配给本地变量空间时，它们的顺序和它们在源代码中声明的顺序一样，和上一条规则一样，应该把长的变量放在短的变量前面。如果第一个变量对齐了，其它变量就会连续的存放，而且不用填充字节自然就会对齐。有些编译器在分配变量时不会自动改变变量顺序，有些编译器不能产生4字节对齐的栈，所以4字节可能不对齐。下面这个例子演示了本地变量声明的重新排序： 不好的代码，普通顺序 推荐的代码，改进的顺序 short ga, gu, gi； long

foo, bar； double x, y, z[3]； char a, b； float baz； double z[3]； double x, y； long foo, bar； float baz； short ga, gu, gi； 避免不必要的整数除法 整数除法是整数运算中最慢的，所以应该尽可能避免。一种可能减少整数除法的地方是连除，这里除法可以由乘法代替。这个替换的副作用是有可能在算乘积时会溢出，所以只能在一定范围的除法中使用。 不好的代码 推荐的代码 int i, j, k, m； m = i / j / k； int i, j, k, m； m = i / (j * k)；