一：线程堆栈：

1>不要在函数内部定义过大的局部变量，如过大的结构体变量、联合变量、多大的字符串、数组等；

2>函数调用的深度需要注意；

3>不要直接将大的结构变量通过函数参数传递，这样也会消耗堆栈空间，可以通过指针或者引用的方式传递；

4>如果在运行中 Coredump，并且通过gdb\dbx的 where 命令时看到刚进入某个函数就报错，连函数内的第一条调试语句都无法指向，则基本可以认为是堆栈空间不够导致的，可以尝试将堆栈空间配置大一点，如果问题不再出现，则可以确定问题。这时需要按照前面几点的要求修改代码，减少堆栈的使用。

参考：<http://blog.csdn.net/dog250/article/details/7704898>

1. 一个线程的开销包括：

内核模式下的开销（内核堆栈，对象管理所需内存）

用户模式下的开销（线程局部存储、线程环境块、堆栈、CRT、MFC、COM等等等等）

通常，线程数目的瓶颈在于线程自己的堆栈。Visual C++编译器默认设置是每个线程的堆栈大小是1MB。当然，如果你在创建线程时指定较小的堆栈大小，你应该可以创建较多的线程。

但是创建大量线程不是一个好的设计。每个线程创建和销毁的时候，Windows会调用已经加载的动态链接库的DLLMain，传递DLL\_THREAD\_ATTACH和DLL\_THREAD\_DETACH作为参数，除非动态库使用DisableThreadLibraryCalls禁用了这个通知。在创建大量线程的时候，这个开销是很大的。对于你这样的用后即弃的线程，你应该使用线程池。一个线程池示例可以在微软知识库找到。

参数和局部变量的函数都存储在线程的堆栈。 如果声明局部变量具有大型值, 堆栈快速耗尽。

二：写时复制（COW）

参考：<http://www.cnblogs.com/biyeymyhjob/archive/2012/07/20/2601655.html>