71 Safety in modern C++ and how to teach it

本节将讨论C++中“安全”意味着什么。

安全编程，就是在编程中，我们希望降低崩溃、内存泄漏、非法访问等问题。
随着C++11的到来，Cherno想说的是应该转向智能指针这样的概念，而不是原始指针。这主要是因为存在内存泄漏以及不知道实际分配或者释放了哪些内存的问题。本节也重点围绕指针和内存，而不是异常或者是其它与安全编程有关的比如错误检查之类的东西。

当我们开始倾向于智能指针之类的东西时，这一切都可以归结为我想要分配**堆内存**，智能指针和自动内存管理系统的存在使程序员的生活更容易，且更有力，这意味着你不再需要处理某些事情，就算忘记处理了它也会自动为你处理。

分配内存这件事很简单，你想在堆上分配一块内存，如果你分配成功会得到一个指向那块内存开始部分的有效的指针，它将一直存在，直到你明确地删除它，这就是整个基本概念了。

那问题就来自几方面了：

- 如果我***忘记释放***我的内存会发生什么问题，可能是无害的甚至注意不到，也有内存耗尽灾难性地导致程序崩溃。而“细心一点、做一个好程序员”显然不是一个真正的解决方案，你还是需要考虑更复杂的结构来删除由你自己明确分配的内存。
- 还有*ownership*（所有权问题），即谁会拥有分配的内存呢？如果我有一个原始指针，指向那块内存，我把它从一个函数传递给另一个函数，从一个类传递给另一个类，谁会**负责管理和清理这些内存**就是***所有权问题***。你不确定A、B这两个管理那个原始指针的函数哪个最后结束，但是要保证两个函数都能访问那个指针，除非你指明这两个函数运行完后再执行一个清理步骤，但这显然会极大复杂化整个程序，也是我们绝对想避免的。我想要重新分配数据，但我不想要显式地建立一些东西，比如管理所有权或者转义所有权，which会使事情变得非常复杂，你将不得不手动跟踪它。这是另一种所有权问题。

这两大问题就是我们需要自动删除内存的原因，当我们讨论C++的安全问题时，特别是智能指针时，我们只需要自动化一行简单的代码就搞定了内存删除与释放问题，所以你百分之百不应该拒绝使用智能指针，自己构建、修改智能指针也是正常的。

当然如果只是做一个一百来行的小型sandbox应用，可能用原始指针可读性更好，因为你不关心是否释放了内存，也不关心所有权，你只用写一个`*`就能让代码会更干净。

Cherno认为大家应该停止关于“Smart or Raw”的争论，在一个真正的框架环境、真正的应用中，生产代码应该使用智能指针，不这么做是非常愚蠢的举动，大部分典型的问题都可以通过这样解决（可能线程方面有点问题，因为`shared_ptr`不是线程安全的，使用智能指针还有很多其它约束，所以智能指针不是通用的内存解决方案）。更严肃的代码中完全应该使用智能指针，只是初学C++是需要了解原始指针和内存是如何工作的，因为智能指针只是原始指针上的包装，它们围绕原始指针做了额外的辅助代码，以便自动化所有事情，但本质上只是删除和释放了内存。你必须得知道这一切是如何工作的，这也是为什么Cherno有几课是讲编译器和链接是如何工作的(06 How the C++ Compiler Works、07 How the C++ Linker Works)