<http://www.cnblogs.com/Lifehacker/p/whats_the_difference_between_libevent_and_libev_chinese.html>

## [[译]libev和libevent的设计差异](https://www.cnblogs.com/Lifehacker/p/whats_the_difference_between_libevent_and_libev_chinese.html)

本文译自[what's the difference between libev and libevent?](http://stackoverflow.com/questions/9433864/whats-the-difference-between-libev-and-libevent) 作者是libev作者

[问]两个库都是为异步io调度而设计，在Linux上都是使用epoll机制，在FreeBSD上则都是kqueue，还有诸如此类的很多相通之处。

除了这些表面上的差别外，其实这两者根本的区别在哪里呢？比如架构上，或者设计哲学上。

[答]就设计哲学来说，libev的诞生，是为了修复libevent设计上的一些错误决策。例如，全局变量的使用，让libevent很难在多线程环境中使用。watcher结构体很大，因为它们包含了I/O，定时器和信号处理器。额外的组件如HTTP和DNS服务器，因为拙劣的实现品质和安全问题而备受折磨。定时器不精确，而且无法很好地处理时间跳变。

libev试图改进所有这些缺陷，例如避免使用全局变量，转而在所有函数中，使用上下文变量来代替。每个事件类型，使用单独的watcher类型（一个I/O watcher在64位机器上，只需要56字节。而libevent需要136字节）。允许额外的事件类型，例如基于挂钟的计时器，或者单调时间，线程内中断，准备并检查watchers来嵌入其他事件循环，或者被用于其他事件循环来嵌入。

额外组件的问题，是通过直接去掉额外组件来解决的，这样libev就可以小而美，快速高效了。但你也需要从其他地方寻找http库。因为libev没有带上。（例如，有一个库叫libeio，可以完成异步IO的工作，也可以和libev配合使用)。

总而言之，libev试图做好一件事而已（目标是成为POSIX的事件库），这是最高效的方法。libevent则尝试给你全套解决方案（事件库，非阻塞IO库，http库，DNS客户端）

一句话总结，libev尝试追随UNIX工具箱哲学，一次只干一件事，每次都做到最好。

注意，这是libev的设计哲学，我想我作为libev的设计者，有着足够的发言权。至于这些设计目标有没有实际达到，或者这些设计哲学是否坚实可靠，则交由你来评判。

[译者注]第一次注意到libev，是在gevent的开发者blog上的这篇[libev and libevent](http://blog.gevent.org/2011/04/28/libev-and-libevent/)，它简要说明了gevent从libevent切换到libev的决策过程。回顾gevent，它实际需要的只是一个负责事件循环的C库，在上面的HTTP库和DNS库，都可以交由标准库强大得不得了的python完成。因此，作者的选择还是非常明智的。

从Libevent 2.0来看，libevent团队已经意识到上述的问题，也提取出了event loop这个上下文context，但是在具体的DNS解析，HTTPS连接等等，还是有种力不从心的感觉。作为libevent的使用者，我们经历了libevent的试错阶段，发现HTTPS实现不行，再切换到libcurl去，与其这样，倒不如直接不提供该功能呢