1.先了解http协议

2.要做一个高并发的web服务器需要用到线程池

3.半同步半异步模式

4.有限状态机编程

碰见的问题：

1.解析请求消息行头体的时候出了很多bug，比如解析不完全，解析错误，解析接收条件有问题，后来参考了linux高性能服务器编程这本书，利用了很多C语言的高阶字符串函数，顺利解决了这个大问题。

2.由于我们需要发送一个响应消息行头体，和一个文件，所以我利用了高阶writev这个函数，writev将多个数据存储在一起，将驻留在两个或更多的不连接的缓冲区中的数据一次写出去。

在我测试简单的静态html页面时没发生什么问题，当我传一个小图片过去也没问题。然后我就把程序放到腾讯的云主机上去跑，再次测试就发现图片解析不完全，整个图片只有上半部分能解析出来，但是在本机测试可以显示，最开始还以为上传到公网之后，是网络传输不稳定造成的。之后我又把各个大小的图片都进行一个传输，发现本机发送15K一下的图片都能接受，云主机上5K一下的图片也能完整的接受。这时候我意识都并不是公网的原因，因为本地文件大了也不能接受，公网只是限制更大了而已。

后来实在检查不出问题，我就去问了老师，他虽然没有给我解决问题，但是他给了我建议，让我利用抓包工具慢慢排查错误。后来我就利用fiddler4这个工具去抓包，看http请求和响应的数据。最终发现我本来要发50K的数据，却发了51K，当文件小的时候没有问题，当文件大了就会多发数据，我想了好久为什么会多发？图片展示不完全应该是少发了啊。这时候我采用以前管用的debug手段（打印），从程序开始到程序结束，我觉得任何有可能出问题的地方都进行打印。最终我发现，由于writev函数返回-1之后，加上errno == EAGAIN所以我对socket再次进行了监听，监听写缓冲区是否能写，当可以写的时候我再次调用writev这个函数进行写，这样就造成了如果一次发不完，第二次就会把响应消息重复发送。所以文件大了之后，发送的内容会多。找到问题之后，我决定不用writev这个函数，我就用平常的write函数进行发送，在类成员中保存每次发送的index，之后再次发送的时候就从index开始发送，确保文件完整的发到浏览器中。

总结：一个好的工具和正确的debug手段，能很好的帮助我们解决问题，比一直盯着代码看好的多。当程序比较大加上是多线程的时候gdb调试往往没有打印效果好。

3.从原则上来讲，父进程只accept就好，其他的读，写都给子线程。《Linux高性能服务器编程》这本书上做的不好，在父进程里面去read和wirte，然后子线程去解析，这样做并不好，有可能你在读的时候又有其他事件来了，你wait不到，这样有些数据就不能及时处理(甚至永远处理不了)。

4.Listen的第二个参数就是“已完成队列”长度。你accept就是从这个队列里面拿数据，有可能在最后通知你队列里面有两个数据时，但是你只accept了一个，这就会造成另一个连接你无法对他进行read或write操作。《Linux高性能服务器编程》只accept了一次，我增加了一个循环，让她全部都accept