关于人的能力，经验，和……

2020-05-03

最近好像没什么特别的事情，工作内容比起以前变多了，占据了很多时间，工作之外的空隙时间，似乎没有太多精力去关注学习。

这个博客上的内容，大概就那么两三种，一种是想到和学习到的技术方面的东西，再一种是工作和生活方面的事情，遇到的或者想到的引起了一些感悟的事情，第三种也许就是流水账、幻想之类，单纯做记录用，写下发生了什么事情。设想中博客的内容频率大概两周到一个月左右，时间太短没什么可说的，时间太长就有点长了。如果连续很长一段时间没什么可写或者不想写，就意味了发生了危险的事情，要么是松懈于动脑子了，要么是心理产生了不正常的问题。所以博客也可以作为自己的照妖镜。

最近纠结于的一点事情，问题在于很难得出什么结论。

(1)

前段时间周末的时候一个读研的同学问了我一点比较基础的操作问题。一个C++的项目，从Github克隆下来，用VS2012打开后编译报错，说找不到文件。我很少接触C++，不过还是尽快了解了一下。项目叫 wxCharts，是一个图表的UI库，编译的时候几乎找不到所有的头文件。他的老师告诉他在常规里把外部依赖库加上，他就加啊加还是找不到头文件。

我说把找不到的文件随便挑一个，在磁盘里搜一下，搜不到……

后来我看了一下wxCharts的官网，知道了wxCharts是依赖于 wxWidgets 的。wxWidgets是一个GUI库，wxCharts只是这个GUI库的样式组件。然后我本地测试了一下，编译安装wxWidgets后wxCharts就能编译过了。

原因找到了，但是他说他已经安装过wxWidgets了。我当时不太确定window下vs的生成和linux的`make install`是不是一回事，就说有那么几种方案，一是把wxWidgets下面的wx文件夹复制到wxCharts下（wx文件夹是项目默认的依赖文件夹，类似Go项目的src目录），二是把wxCharts的项目源码放到wxWidgets项目里试试能不能编译过，三是重新安装一下wxWidget，一定要全局安装，环境变量里配置一下。

后来折腾半天总算是能找到头文件了，外部依赖那里配置的路径有问题，用了错误的环境变量作为路径的变量，还重复引用了错误的路径。但是编译还不通过，找不到什么什么文件的，又折腾了一下，就是除了头文件还要引入动态依赖库，而且编译了生产版本的目录位置不一样，后面带d的是debug版本……

其实整个过程面对的问题很简单，就是找不到项目依赖，如果Java项目必定轻车熟路了（话说其实有工作好几年的同事面对项目依赖这种低级问题有时候还难以搞定的）。这是我一个关系不错的同学，本科时我们天天坐在床上打王者荣耀。他的实力我还是稍微有一些了解，可能学术能力比较强，但计算机方面真是比小白还小白。但要说学术能力多强其实本科成绩还没我好，也就是考研的一年多确实实实在在的努力了。

稍有感触的一点是，说起来现在他算是一所还不错的211大学的研究生，我的学历依然是双非本科，学历上差了好几个段位。至于能力的话，只能说曾经有一个考研的机会放在我面前，我没有珍惜，直到失去了也没有一丝丝悔意，如果上天给我一个重新来过的机会，我一定会做出和现在相同的决定，如果非要给这个决定加一个期限，我希望是，有生之年。

(2)

有一个我以前也提到过的让我感觉充满矛盾的同事，实际的动手能力和对工作的负责程度真是让我感觉理解不了。但是后来遇到几次问题，发现她毕竟还是有丰富开发经验的，经验方面真的难以反驳的超过我。

当时面对比较简单的一个递归的问题。我之前也确实一直没搞明白，一个对象传到一个函数里

面，是值还是引用传递?

```
function a() {
    obj = { "a": 1 }
    b(obj)
    console.log(obj)
}
function b(obj) {
    // 方式1
    obj = 1
    // 方式2
    obj = { "a": 2 }
    // 方式3
    obj.a = 2
}
```

必须先搞清楚这个问题，才能使用递归解决问题。方式1直接给对象赋值，方式2给了参数一个新的对象，方式3改变了对象的属性。我当时是没有概念的，一心想着是不是能用原型链解决这个问题。我也知道她不太清楚这个问题，但是她疑疑惑惑的提了一句这个和堆栈是不是有关系，然后又很不确定的没再提了。

我以为没有关系，我以为她是随口说的。我错了。

对象是引用传递的，这一点没有问题，但对象的属性有没有改变，取决于改变的是对象的栈空间还是堆空间。给对象赋新值，不管是简单值还是复合值，都会将对象的变量指向新的栈地址，所以对象的属性并不会改变。如果直接用方式3的写法，改变的是对象指向的堆里面的内容，原对象的属性就会改变。

所以其实经验在一定程度上是有用并且难以轻易超越的，经验基本上无法通过捷径获得。

(3)

能力通过经验获得，强大的能力可以更快更有效的获取更高级的经验，然后这些经验会使能力更加强大。

也许能力和经验是相辅相成的，类比所谓的“经验和洞察力”。其实这里的“能力”确实稍微有点“洞察力”的意味。有时候在招聘的时候感觉招聘方一味强调多少多少年经验，好像用工作经验多的人就比用经验少的人占了便宜一样，但是吧，关于能力和经验……

这两天下了一个编曲的软件，简单尝试一下，感觉编曲这个事情，其实还是很难的，和编程虽然只有一字之差，技能要求却是截然不同而且更高的。现在各种培训班、在线课程，编程已经几乎没有门槛，当“全民编程”的时代来临后，我们改怎么面对这个世界呢。未来的编程也许不会把MVC、ORM什么的渗透到生活中，但一套物联网设备、自动化设备，或者iOS系统快捷指令的未来版本，如何更加适合人类个性化的生活习惯，可能多少也会需要点“编程”的逻辑。

PS:

iOS 13的快捷指令已经支持基本的语句逻辑