High一下!

酷 壳 – CoolShell

享受编程和技术所带来的快乐 – Coding Your Ambition
(https://coolshell.cn/)

Search …

程序员技术练级攻略

2011年07月18日 (Https://Coolshell.Cn/Articles/4990.Html) 陈皓 (Https://Coolshell.Cn/Articles/Author/Haoel) 9,707,046 人阅读

月光博客6月12日发表了《写给新手程序员的一封信 (http://www.williamlong.info/archives/2700.html)》，翻译自《An open letter to those who want to start programming (http://blog.akash.im/an-open-letter-to-those-who-want-to-start)》，我的朋友（他在本站的id是Mailper (https://coolshell.cn/?author=3)）告诉我，他希望在酷壳上看到一篇更具操作性的文章。因为他也是喜欢编程和技术的家伙，于是，我让他把他的一些学习Python和Web编程的一些点滴总结一下。于是他给我发来了一些他的心得和经历，我在把他的心得做了不多的增改，并根据我的经历增加了"进阶"一节。**这是一篇由新手和我这个老家伙根据我们的经历完成的文章。**

我的这个朋友把这篇文章取名叫Build Your Programming Technical Skills，我实在不知道用中文怎么翻译，但我在写的过程中，**我觉得这很像一个打网游做任务升级的一个过程，所以取名叫"技术练级攻略"，题目有点大，呵呵，这个标题纯粹是为了好玩。这里仅仅是在分享Mailper和我个人的学习经历。**（注：省去了我作为一个初学者曾经学习过的一些技术(今天明显过时了)，如：Delphi/Power builder，也省去了我学过的一些我觉得没意思的技术Lotus Notes/ActiveX/COM/ADO/ATL/.NET ……）

前言

你是否觉得自己从学校毕业的时候只做过小玩具一样的程序？走入职场后哪怕没有什么经验也可以把以下这些课外练习走一遍（朋友的抱怨：学校课程总是从理论出发，作业项目都看不出有什么实际作用，不如从工作中的需求出发）

建议：

- 不要乱买书，不要乱追新技术新名词，基础的东西经过很长时间积累而且还会在未来至少10年通用。
- 回顾一下历史，看看历史上时间线上技术的发展，你才能明白明天会是什么样。
- 一定要动手，例子不管多么简单，建议至少自己手敲一遍看看是否理解了里头的细枝末节。
- 一定要学会思考，思考为什么要这样，而不是那样。还要举一反三地思考。

注：你也许会很奇怪为什么下面的东西很偏Unix/Linux，这是因为我觉得Windows下的编程可能会在未来很没有前途，原因如下：

- 现在的用户界面几乎被两个东西主宰了，1）Web，2）移动设备iOS或Android。Windows的图形界面不吃香了。
- 越来越多的企业在用成本低性能高的Linux和各种开源技术来构架其系统，Windows的成本太高了。
- 微软的东西变得太快了，很不持久，他们完全是在玩弄程序员。详情参见《Windows编程革命史 (https://coolshell.cn/articles/3008.html)》

所以，我个人认为以后的趋势是前端是Web+移动，后端是Linux+开源。开发这边基本上没Windows什么事。

启蒙入门

1、 学习一门脚本语言，例如Python/Ruby

可以让你摆脱对底层语言的恐惧感，脚本语言可以让你很快开发出能用得上的小程序。实践项目:

- 处理文本文件，或者csv (关键词 python csv, python open, python sys) 读一个本地文件，逐行处理（例如 word count，或者处理log）
- 遍历本地文件系统 (sys, os, path)，例如写一个程序统计一个目录下所有文件大小并按各种条件排序并保存结果
- 跟数据库打交道 (python sqlite)，写一个小脚本统计数据库里条目数量
- 学会用各种print之类简单粗暴的方式进行调试
- 学会用Google (phrase, domain, use reader to follow tech blogs)

为什么要学脚本语言，因为他们实在是太方便了，很多时候我们需要写点小工具或是脚本来帮我们解决问题，你就会发现正规的编程语言太难用了。

2、 用熟一种程序员的编辑器(不是IDE) 和一些基本工具

- Vim / Emacs / Notepad++，学会如何配置代码补全，外观，外部命令等。

- Source Insight (或 ctag)

使用这些东西不是为了Cool，而是这些编辑器在查看、修改代码/配置文章/日志会更快更有效率。

3、 熟悉Unix/Linux Shell和常见的命令行

- 如果你用windows，至少学会用虚拟机里的linux， vmware player是免费的，装个Ubuntu吧
- 一定要少用少用图形界面。
- 学会使用man来查看帮助
- 文件系统结构和基本操作 ls/chmod/chown/rm/find/ln/cat/mount/mkdir/tar/gzip …
- 学会使用一些文本操作命令 sed/awk/grep/tail/less/more …
- 学会使用一些管理命令 ps/top/lsof/netstat/kill/tcpdump/iptables/dd…
- 了解/etc目录下的各种配置文章，学会查看/var/log下的系统日志，以及/proc下的系统运行信息
- 了解正则表达式，使用正则表达式来查找文件。

对于程序员来说Unix/Linux比Windows简单多了。（参看我四年前CSDN的博文《其实Unix很简单 (http://blog.csdn.net/haoel/article/details/1533720)》）学会使用Unix/Linux你会发现图形界面在某些时候实在是太难用了，相当地相当地降低工作效率。

4、 学习Web基础（HTML/CSS/JS)+服务器端技术 (LAMP)

未来必然是Web的世界，学习WEB基础的最佳网站是W3School (http://www.w3school.com.cn/)。

- 学习HTML基本语法
- 学习CSS如何选中HTML元素并应用一些基本样式（关键词：box model）
- 学会用 Firefox + Firebug 或 chrome 查看你觉得很炫的网页结构，并动态修改。
- 学习使用Javascript操纵HTML元件。理解DOM和动态网页（http://oreilly.com/catalog/9780596527402 (http://oreilly.com/catalog/9780596527402)) 网上有免费的章节，足够用了。或参看 DOM (http://www.w3school.com.cn/htmldom/index.asp) 。
- 学会用 Firefox + Firebug 或 chrome 调试Javascript代码（设置断点，查看变量，性能，控制台等）
- 在一台机器上配置Apache (www.apache.org)或 Nginx (nginx.net)
- 学习PHP (www.php.net)，让后台PHP和前台HTML进行数据交互，对服务器相应浏览器请求形成初步认识。实现一个表单提交和反显的功能。
- 把PHP连接本地或者远程数据库 MySQL（MySQL 和 SQL现学现用够了）
- 跟完一个名校的网络编程课程（例如：http://www.stanford.edu/~ouster/cgi-bin/cs142-fall10/index.php (http://www.stanford.edu/~ouster/cgi-bin/cs142-fall10/index.php)) 不要觉得需要多于一学期时间，大学生是全职一学期选3-5门课，你业余时间一定可以跟上
- 学习一个javascript库（例如jQuery 或 ExtJS）+ Ajax (异步读入一个服务器端图片或者数据库内容）+JSON数据格式。
- HTTP: The Definitive Guide 读完前4章你就明白你每天上网用浏览器的时候发生的事情了(proxy, gateway, browsers)
- 做个小网站（例如：一个小的留言板，支持用户登录，Cookie/Session，增、删、改、查，上传图片附件，分页显示）
- 买个域名，租个空间，做个自己的网站。

进阶加深

1、 C语言和操作系统调用

- 重新学C语言，理解指针和内存模型，用C语言实现一下各种经典的算法和数据结构。推荐《计算机程序设计艺术 (http://product.china-pub.com/197050)》、《算法导论 (http://product.china-pub.com/31701)》和《编程珠玑 (http://product.china-pub.com/209243)》。
- 学习（麻省理工免费课程）计算机科学和编程导论 (https://coolshell.cn/articles/3723.html)
- 学习（麻省理工免费课程）C语言内存管理 (https://coolshell.cn/articles/2474.html)
- 学习Unix/Linux系统调用（Unix高级环境编程 (http://product.china-pub.com/30181)），，了解系统层面的东西。
  - 用这些系统知识操作一下文件系统，用户（实现一个可以拷贝目录树的小程序）
  - 用fork/wait/waitpid写一个多进程的程序，用pthread写一个多线程带同步或互斥的程序。多进程多进程购票的程序。
  - 用signal/kill/raise/alarm/pause/sigprocmask实现一个多进程间的信号量通信的程序。
  - 学会使用gcc和gdb来编程和调试程序（参看我的《用gdb调试程序 (blog.csdn.net/haoel/article/details/2879)》）
  - 学会使用makefile来编译程序。（参看我的《跟我一起写makefile (blog.csdn.net/haoel/article/details/2886)》）
  - IPC和Socket的东西可以放到高级中来实践。
- 学习Windows SDK编程（Windows 程序设计 (http://product.china-pub.com/52880)，MFC程序设计 (http://product.china-pub.com/3804)）
  - 写一个窗口，了解WinMain/WinProcedure，以及Windows的消息机制。
  - 写一些程序来操作Windows SDK中的资源文件或是各种图形控件，以及作图的编程。
  - 学习如何使用MSDN查看相关的SDK函数，各种WM_消息以及一些例程。
  - 这本书中有很多例程，在实践中请不要照抄，试着自己写一个自己的例程。
  - 不用太多于精通这些东西，因为GUI正在被Web取代，主要是了解一下Windows 图形界面的编程。@virushuo (http://twitter.com/#!/virushuo) 说：“我觉得GUI确实不那么热门了，但充分理解GUI工作原理是很重要的。包括移动设备开发，如果没有基础知识仍然很吃力。或者说移动设备开发必须理解GUI工作，或者在win那边学，或者在mac/iOS上学”。

2、学习Java

- Java 的学习主要是看经典的Core Java 《Java 核心技术编程 (http://product.china-pub.com/208978)》和《Java编程思想 (http://product.china-pub.com/34838)》（有两卷，我仅链了第一卷，足够了，因为Java的图形界面了解就可以了）
- 学习JDK，学会查阅Java API Doc http://download.oracle.com/javase/6/docs/api/ (http://download.oracle.com/javase/6/docs/api/)
- 了解一下Java这种虚拟机语言和C和Python语言在编译和执行上的差别。从C、Java、Python思考一下“跨平台”这种技术。
- 学会使用IDE Eclipse，使用Eclipse 编译，调试和开发Java程序。
- 建一个Tomcat的网站，尝试一下JSP/Servlet/JDBC/MySQL的Web开发。把前面所说的那个PHP的小项目试着用JSP和Servlet实现一下。

3、Web的安全与架构

- 学习HTML5，网上有很多很多教程，以前酷壳 (https://coolshell.cn)也介绍过很多，我在这里就不罗列了。
- 学习Web开发的安全问题（参考新浪微博被攻击的这个事 (https://coolshell.cn/articles/4914.html)，以及Ruby的这篇文章 (http://guides.rubyonrails.org/security.html)）
- 学习HTTP Server的rewrite机制，Nginx的反向代理机制，fast-cgi (http://en.wikipedia.org/wiki/Fast_CGI)（如：PHP-FPM (http://php-fpm.org/) ）
- 学习Web的静态页面缓存技术。
- 学习Web的异步工作流处理，数据Cache，数据分区，负载均衡，水平扩展的构架。
- **实践任务：**
  - 使用HTML5的canvas 制作一些Web动画。
  - 尝试在前面开发过的那个Web应用中进行SQL注入，JS注入，以及XSS攻击。

- 把前面开发过的那个Web应用改成构造在Nginx + PHP-FPM + 静态页面缓存的网站

4、学习关系型数据库

- 你可以安装MSSQLServer或MySQL来学习数据库。
- 学习教科书里数据库设计的那几个范式，1NF，2NF，3NF，……
- 学习数据库的存过，触发器，视图，建索引，游标等。
- 学习SQL语句，明白表连接的各种概念（参看《SQL Join的图示 (https://coolshell.cn/articles/3463.html)》）
- 学习如何优化数据库查询（参看《MySQL的优化 (https://coolshell.cn/articles/1846.html)》）
- **实践任务**：设计一个论坛的数据库，至少满足3NF，使用SQL语句查询本周，本月的最新文章，评论最多的文章，最活跃用户。

5、一些开发工具

- 学会使用SVN或Git来管理程序版本。
- 学会使用JUnit来对Java进行单元测试。
- 学习C语言和Java语言的coding standard或coding guideline。（我N年前写过一篇关C语言非常简单的文章——《编程修养 (http://blog.csdn.net/haoel/article/category/9200/2)》，这样的东西你可以上网查一下，一大堆）。
- 推荐阅读《代码大全 (http://product.china-pub.com/28351)》《重构 (http://product.china-pub.com/196374)》《代码整洁之道 (http://product.china-pub.com/196266)》

高级深入

1、C++ / Java 和面向对象

我个人以为学好C++，Java也就是举手之劳。但是C++的学习曲线相当的陡。不过，我觉得C++是最需要学好的语言了。参看两篇趣文“C++学习信心图 (https://coolshell.cn/articles/2287.html)”和“21天学好C++ (https://coolshell.cn/articles/2250.html)”

- 学习（麻省理工免费课程）C++面向对象编程 (https://coolshell.cn/articles/2474.html)
- 读我的“如何学好C++ (https://coolshell.cn/articles/4119.html)”中所推荐的那些书至少两遍以上（如果你对C++的理解能够深入到像我所写的《C++虚函数表解析 (https://coolshell.cn/articles/12165.html)》或是《C++对象内存存局 (https://coolshell.cn/articles/12176.html)》，或是《C/C++返回内部静态成员的陷阱 (https://coolshell.cn/articles/12192.html)》那就非常不错了）
- 然后反思为什么C++要干成这样，Java则不是？你一定要学会对比C++和Java的不同。比如，Java中的初始化，垃圾回收，接口，异常，虚函数，等等。
- **实践任务：**
  - 用C++实现一个BigInt，支持128位的整形的加减乘除的操作。
  - 用C++封装一个数据结构的容量，比如hash table。
  - 用C++封装并实现一个智能指针（一定要使用模板）。
- 《设计模式 (http://product.china-pub.com/25961)》必需一读，两遍以上，思考一下，这23个模式的应用场景。主要是两点：1）钟爱组合而不是继承，2）钟爱接口而不是实现。（也推荐《深入浅出设计模式 (http://product.china-pub.com/27862)》）
- **实践任务：**
  - 使用工厂模式实现一个内存池。
  - 使用策略模式制做一个类其可以把文本文件进行左对齐，右对齐和中对齐。
  - 使用命令模式实现一个命令行计算器，并支持undo和redo。
  - 使用修饰模式实现一个酒店的房间价格订价策略——旺季，服务，VIP、旅行团、等影响价格的因素。
- 学习STL的用法和其设计概念 – 容器，算法，迭代器，函数子。如果可能，请读一下其源码。
- **实践任务：**尝试使用面向对象、STL，设计模式、和WindowsSDK图形编程的各种技能
  - 做一个贪吃蛇或是俄罗斯方块的游戏。支持不同的级别和难度。
  - 做一个文件浏览器，可以浏览目录下的文件，并可以对不同的文件有不同的操作，文本文件可以打开编辑，执行文件则执行之，mp3或avi文件可以播放，图片文件可以展示图片。
- 学习C++的一些类库的设计，如：MFC（看看候捷老师的《深入浅出MFC (http://product.china-pub.com/3565)》），Boost, ACE, CPPUnit，STL（STL可能会太难了，但是如果你能了解其中的设计模式和设计那就太好了，如果你能深入到我写的《STL string类的写时拷贝技术 (http://blog.csdn.net/haoel/article/details/24058)》那就非常不错了，ACE需要很强在的系统知识，参见后面的“加强对系统的了解”）
- Java是真正的面向对象的语言，Java的设计模式多得不能再多，也是用来学习面向对象的设计模式的最佳语言了（参看Java中的设计模式 (https://coolshell.cn/articles/3320.html)）。
- 推荐阅读《Effective Java (http://product.china-pub.com/195040)》 and 《Java解惑 (http://product.china-pub.com/197212)》
- 学习Java的框架，Java的框架也是多，如Spring, Hibernate，Struts 等等，主要是学习Java的设计，如IoC等。
- Java的技术也是烂多，重点学习J2EE架构以及JMS， RMI, 等消息传递和远程调用的技术。
- 学习使用Java做Web Service（官方教程在这里 (http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/docs/2.0/tutorial/doc/)）
- **实践任务：** 尝试在Spring或Hibernate框架下构建一个有网络的Web Service的远程调用程序，并可以在两个Service中通过JMS传递消息。

C++和Java都不是能在短时间内能学好的，C++玩是的深，Java玩的是广，我建议两者选一个。我个人的学习经历是：

- 深究C++（我深究C/C++了十来年了）
- 学习Java的各种设计模式。

2、加强系统了解

重要阅读下面的几本书：

- 《Unix编程艺术 (http://product.china-pub.com/197413)》了解Unix系统领域中的设计和开发哲学、思想文化体系、原则与经验。你一定会有一种醍醐灌顶的感觉。
- 《Unix网络编程卷1，套接字 (http://product.china-pub.com/196770)》这是一本看完你就明白网络编程的书。重要注意TCP、UDP，以及多路复用的系统调用select/poll/epoll的差别。
- 《TCP/IP详解 卷1:协议 (http://product.china-pub.com/35)》- 这是一本看完后你就可以当网络黑客的书。了解以太网的的运作原理，了解TCP/IP的协议，运作原理以及如何TCP的调优。
- **实践任务：**
  - 理解什么是阻塞（同步IO），非阻塞（异步IO），多路复用（select, poll, epoll）的IO技术。
  - 写一个网络聊天程序，有聊天服务器和多个聊天客户端（服务端用UDP对部分或所有的的聊天客户端进Multicast或Broadcast）。
  - 写一个简易的HTTP服务器。
- 《Unix网络编程卷2，进程间通信 (http://product.china-pub.com/196859)》信号量，管道，共享内存，消息等各种IPC……这些技术好像有点老掉牙了，不过还是值得了解。
- **实践任务：**
  - 主要实践各种IPC进程序通信的方法。

    - 尝试写一个管道程序，父子进程通过管道交换数据。
    - 尝试写一个共享内存的程序，两个进程通过共享内存交换一个C的结构体数组。
- 学习《Windows核心编程 (http://product.china-pub.com/209058)》一书。把CreateProcess，Windows线程、线程调度、线程同步（Event, 信号量，互斥量）、异步I/O，内存管理，DLL，这几大块搞精通。
- **实践任务：**使用CreateProcess启动一个记事本或IE，并监控该程序的运行。把前面写过的那个简易的HTTP服务用线程池实现一下。写一个DLL的钩子程序监控指定窗口的关闭事件，或是记录某个窗口的按键。
- 有了多线程、多进程通信，TCP/IP，套接字，C++和设计模式的基本，你可以研究一下ACE了。使用ACE重写上述的聊天程序和HTTP服务器（带线程池）
- **实践任务：**通过以上的所有知识，尝试
    - 写一个服务端给客户端传大文件，要求把100M的带宽用到80%以上。（注意，磁盘I/O和网络I/O可能会很有问题，想一想怎么解决，另外，请注意网络传输最大单元MTU）
    - 了解BT下载的工作原理，用多进程的方式模拟BT下载的原理。

3、系统架构

- 负载均衡。HASH式的，纯动态式的。（可以到Google学术里搜一些关于负载均衡的文章 (http://scholar.google.com.hk/scholar?q=%E8%B4%9F%E8%BD%BD%E5%9D%87%E8%A1%A1&hl=zh-CN&as_sdt=0&as_vis=1&oi=scholart)读读）
- 多层分布式系统 – 客户端服务结点层、计算结点层、数据cache层，数据层。J2EE是经典的多层结构。
- CDN系统 (http://en.wikipedia.org/wiki/Content_delivery_network) – 就近访问，内容边缘化。
- P2P式系统 (http://en.wikipedia.org/wiki/Peer-to-peer)，研究一下BT和电驴的算法。比如：DHT算法 (http://en.wikipedia.org/wiki/Distributed_hash_table)。
- 服务器备份，双机备份系统（Live-Standby和Live-Live系统），两台机器如何通过心跳监测对方？集群主结点备份。
- 虚拟化技术 (http://en.wikipedia.org/wiki/Virtualization)，使用这个技术，可以把操作系统当应用程序一下切换或重新配置和部署。
- 学习Thrift (http://thrift.apache.org/)，二进制的高性能的通讯中间件，支持数据(对象)序列化和多种类型的RPC服务。
- 学习Hadoop (http://hadoop.apache.org/)。Hadoop框架中最核心的设计就是：MapReduce和HDFS。MapReduce的思想是由Google的一篇论文所提及而被广为流传的，简单的一句话解释MapReduce就是“任务的分解与结果的汇总”。HDFS是Hadoop分布式文件系统（Hadoop Distributed File System）的缩写，为分布式计算存储提供了底层支持。
- 了解NoSQL数据库 (http://en.wikipedia.org/wiki/NoSQL)（有人说可能是一个过渡炒作的技术 (https://coolshell.cn/articles/3609.html)），不过因为超大规模以及高并发的纯动态型网站日渐成为主流，而SNS类网站在数据存取过程中有着实时性等刚性需求，这使得目前NoSQL数据库慢慢成了人们所关注的焦点，并大有成为取代关系型数据库而成为未来主流数据存储模式的趋势。当前NoSQL数据库很多，大部分都是开源的，其中比较知名的有：MemcacheDB、Redis、Tokyo Cabinet(升级版为Kyoto Cabinet)、Flare、MongoDB、CouchDB、Cassandra、Voldemort等。

写了那么多，回顾一下，觉得自己相当的有成就感。希望大家不要吓着，我自己这十来年也在不断地学习，今天我也在学习中，人生本来就是一个不断学习和练级的过程。**不过，一定有漏的，也有不对的，还希望大家补充和更正。（我会根据大家的反馈随时更新此文）**欢迎大家通过我的微博（@左耳朵耗子 (http://weibo.com/haoel)）和twitter（@haoel (http://twitter.com/haoel)）和我交流。

—— 更新 2011/07/19 ——

1）有朋友奇怪为什么我在这篇文章开头说了web+移动，却没有在后面提到iOS/Android的前端开发。因为我心里有一种感觉，移动设备上的UI最终也会被Javascript取代。大家可以用iPhone或Android看看google+，你就会明白了。

2）有朋友说我这里的东西太多了，不能为了学习而学习，我非常同意。我在文章的前面也说了要思考。另外，千万不要以为我说的这些东西是一些新的技术，这份攻略里95%以上的全是基础。而且都是久经考验的基础技术。即是可以让你一通百通的技术，也是可以让你找到一份不错工作的技术。

3）有朋友说学这些东西学完都40了，还不如想想怎么去挣钱。我想告诉大家，一是我今年还没有40岁，二是学无止境啊，三是我不觉得挣钱有多难，难的是怎么让你值那么多钱？无论是打工还是创业，是什么东西让你自己的价值，让你公司的价值更值钱？别的地方我不敢说，对于互联网或IT公司来说，技术实力绝对是其中之一。

4）有朋友说技术都是工具，不应该如此痴迷这句话没有错，有时候我们需要更多的是抬起头来看看技术以外的事情，或者是说我们在作技术的时候不去思考为什么会有这个技术，为什么不是别的，问题不在于技术，问题在于我们死读书，读死书，成了技术的书呆子。

5）对于NoSQL，最近比较火，但我对其有点保守，所以，我只是说了解就可以。对于Hadoop，我觉得其在分布式系统上有巨大的潜力，所以需要学习。 对于关系型数据库，的确是很重要的东西，这点是我的疏忽，在原文里补充。

（全文完，转载时请注明作者和出处）

关注CoolShell微信公众账号可以在手机端搜索文章