李丙洋 著

涂抹MySQL

跟着一思一步一步学MySQL

中国水利水电出版社
www.waterpub.com.cn

李丙洋 著

涂抹MySQL

跟着三思一步一步学MySQL

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书不是一本逐个介绍 MySQL 命令的书，不是一本用户帮助手册，也不是这个功能讲完讲那个功能的书。因为在写作之初我就设定了一条主线，不是依次讲特性，而要依据用户接触和学习 MySQL 的脉络去把握内容的安排。

本书主要侧重于 MySQL 数据库从无到有及其安装、配置、管理、优化的过程，其中穿插介绍数据导入导出，性能/状态监控，备份恢复和优化方面等内容，同时还会谈一谈 MySQL 数据库服务从单台到多台，从单实例到多实例集群的部署方案。

本书主要面向 Web 应用的一线开发人员和对 MySQL 数据库较有兴趣，希望使用或正在使用的读者。对于有志从事数据库管理员相关职业的读者，相信本书能够帮助他们快速找到入门的路径；本书中提到的一些技巧类应用和扩展方案，即使对于具有一定技术实力的有经验的 MySQL DBA，相信也会有一定的启发；此外本书也可以作为大中专院校相关专业师生的参考工具书和相关培训机构的培训教材。

本书部分源代码，读者可以到中国水利水电出版社网站及万水书苑免费下载，网址为 http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

涂抹MySQL：跟着三思一步一步学MySQL / 李丙洋著
. -- 北京：中国水利水电出版社，2014.4
ISBN 978-7-5170-1867-7

Ⅰ. ①涂… Ⅱ. ①李… Ⅲ. ①关系数据库系统 Ⅳ.
①TP311.138

中国版本图书馆CIP数据核字(2014)第067636号

策划编辑：周春元　　责任编辑：李　炎　　加工编辑：刘晶平

书　　名	涂抹 MySQL——跟着三思一步一步学 MySQL
作　　者	李丙洋　著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	184mm×240mm　16 开本　34.5 印张　740 千字
版　　次	2014 年 4 月第 1 版　2014 年 4 月第 1 次印刷
印　　数	0001—4000 册
定　　价	68.00 元

又见涂抹（推荐序）

去 IOE,闹着玩的？

“谁是谁的谁”，一首唯美中透着幽怨的歌，却会莫名地电到心中的累。学习太累，工作太累，生活太累，连歌名听着都那么累。

曾经的《涂抹 Oracle》，从策划，到上市，到畅销，到经典，实现了一个个不敢梦的梦想。正如一句歌词唱的：你看窗外花开那么美。

但不幸的是，武林盟主阿里系带头发起了“去 IOE”运动（IBM，Oracle，EMC）。为毛去 IOE？都是金钱惹的祸，用不起啊用不起。连盟主阿里都快用不起了，江湖又如何？

所以搞数据库的大牛小虾不免心有戚戚，一颗红心，两手准备吧，早做打算为妙。

谁来接班？

所以，你得清楚了解阿里去了 Oracle 想让谁接班。

没错，就是 MySQL！

但樱桃好吃树难栽，由 Oracle 转战 MySQl，不是开车变换一条车道那么简单。这个你懂，我懂，天下人都懂，当然，经历了从 Oracel 车道往 MySQL 车道变换的三思更懂。

扫地僧

提起三思，真是小孩没娘，说来话长，请允许我再说一遍吧，保证长话短说。

孔老二有个观点，“知之者不如好知者，好知者不如乐知者”，意即“乐知是学习的最高境界”。我虽对老二的多数观点深恶痛绝，但对这一点却是深以为然。

某年某月某一天。

本人在 ITPUB 闲逛，偶然看到某标有“专家”字样的数据库类技术博客，衰名“君三思”。他的签名也引用了孔老二的这句话，觉得是个同道中人，不免在人群中多看了他几眼，却又瞥见了一句说明：我就是那个无名的扫地僧——手里的扫把。正是这句话，勾搭的我闯进了他的博客。

随便点开一篇名为《小记 pub08 年会三两事之三、打扑克~~~》（链接见 http://space.itpub.net/

7607759/viewspace-151194）的博文。我承认，虽未见其人，但一个洒脱、机智、幽默、才华横溢或许还有些坏坏的君三思已经活脱脱浮现于脑海。我开始浏览他的其他博文，基本都是Oracle 学习笔记的系列博文，我带着些许的遗憾随便点阅了一篇，情不自禁的又点了一篇，又一篇……遗憾之感早已不翼而飞。他哪里只是一只扫把，其思想、其笔锋、其幽默、其技术、其功力，分明已至神光内敛返璞归真之境，他分明就是寺中的那个无名的扫地老僧。一种强烈的预感，这也许就是我要找的 Oracle 神灯吧。

Email，打电话，吃饭，约稿……

总之，我是死乞白赖地认识了君三思。也就有了曾经那本让读者大呼过瘾的《涂抹 Oracle》。

而不久以后，三思开始玩 MySQL 了，到底是不是因为去 IOE 的运动，我就不清楚了。

涂抹

精彩就像天边的彩虹，所以，有时候形容精彩不免想用“抹”。我审阅稿子时，喜欢记录书稿中每一抹精彩。

初看到三思给我的第一本 Oracle 书稿时，我震惊于连“Oracle 发展历史”这样的白开水内容，在三思的笔下都那么的精彩和酣畅（网上可见试读样章），更震惊于 Oracle 在 Windows 环境下和 Linux 环境下的安装与配置这样白开水般的内容在三思笔下竟然那么的具有深度，而三思对于 RMAN、DG、FLASHBACK、SQL*LOADER、加载、备份恢复、迁移、体系结构等相关知识的研究和理解是如此的深刻，其表达又能如此的简单、风趣而幽默。书中的精彩，记下一抹又一抹，右手麻木的时候，我发现原来我基本上是在抄书稿。所以不得不放弃记录。当时，我突然想到，既然精彩已不可数，索性就把书名定为“涂抹”吧，以表达我审阅此书的心情。

所以，还是涂抹

三思从 Oracle 转战 MySQL，转眼几年已经过去。

这本 MySQL 的稿子也已经交来三月有余。三思告诉我，这不是一本逐个介绍 MySQL 命令的书，不是讲完这个功能讲那个功能的书，更不是一本用户手册。叫个什么名好呢？

我一点都不担心为这本书起名字的问题，我最担心的，倒是三思的这本 MySQL 能否超越他的上一本 Oralce，或者至少可以一样精彩。

但阅读完这本 MySQL 的稿子，才发现，我的担心是多么的多余。但我却无法为此书起一个更恰当的名字，只好还是叫《涂抹 MySQL》吧，见谅，见谅。

周春元

轮扁斫轮（自序）

这些年一直坚持在博客上发表系列文章，也出版过技术方面的书，在业内积累了那么一点点知名度，就有很多朋友慕名而来跟我交流，要向我请教技术上的问题，咨询学习的技巧，让我推荐阅读的图书等。朋友们的热情让我感到很是忐忑，我虽然写过一些文章，但都是些通俗的东西，其实没什么学问，而且老实讲，有些朋友提出的问题真是不容易回答，比如有朋友上来就问我怎么优化数据库（这类问题恰恰是最多的），也有人问我看什么书能达到我现在的水平等等。

我知道这其中有些人确实是喜欢并且希望从事DBA这个职业的，他们当前只是没有经验才显得迷茫，其实我也是从这个阶段走过来的。多年以前我曾写过一篇文章《我想对初学 Oracle 的朋友说》（http://www.5ienet.com/note/html/stdstep/how-to-learn-in-oracle.shtml），其中的内容也是我有感而发。因为了解DBA成长道路上的艰辛，我不愿意随口敷衍他们几句，使他们走了弯路。可是因为自己的水平有限，我又难以简单几句话就说明白，有时候拉拉扯扯说了一堆，反倒是提问者自己不耐烦，甚至还有朋友直接了当地指责我不愿意分享（这个杀伤力太大，我的文章都白写了吗），整的我的心里也很受伤，对于这样的人，我当机立断大喝一声："你是猴子搬来的救兵吗？"。不过我知道，多数人还是希望通过不断学习获得提高，看到他们就像看到曾经的我，我真心想帮助这些朋友。

说起读书和学习，国内外不少前辈先贤都有论述的文章，先我们耳熟能详的："学而不思则罔，思而不学则殆；知之为知之，不知为不知……"等出自《论语》的至理名言就已流传千古，我一个刚及而立的后学晚辈，竟然也敢以《论》起谈，岂不是在自暴己丑，是要叫人笑掉大牙的嘛。但是我想，圣贤们的文章高瞻远瞩，高屋建瓴，高处不胜寒哪，以至于应者寥寥，我基于自身实力水平，也想借这个场合，站在稍低一层的层次，谈一谈这些年我学习过程中的一些感受。

先从一个典故开始吧。

春秋五霸之一的齐桓公，一次在堂上读书，堂下一名叫轮扁的技工师傅看到了，就放下手上的工作，走到堂上问桓公："请问桓公在看什么书？"

齐桓公回答他说："这是圣人的书。"

轮扁接着问："圣人在哪呢？"

齐桓公回答："圣人已经死了！"

轮扁说道："那桓公所读的，不过是古人留下来的糟粕罢了。"

齐桓公闻之怒道："寡人读书，岂是你这个做车轮的工匠可以议论的吗，今天你要是能说出道理还则罢了，要是说不出来，明年的今天就是你的忌日（好吧，我承认武侠片看多了，这台词管不住自己都往外蹦）。"

轮扁于是说道："我是通过我平常工作观察到的情况来理解的，给车做轱辘虽然不是高级岗位，但也是个技术工种。在做车轮的时候，如果轮孔弄的太宽，那么虽然车轮能做的很光滑但用起来并不牢固，如果轮孔弄的过紧，车轮又会很粗糙难以装配；因此只有得心应手，不紧不松才能做出高端大气上档次的车轮。可是要如何做到得心应手呢，我做轮子的时候知道存在这样一种境界，但用嘴又说不清楚，我甚至都没办法将这点技巧传授给我的儿子，我的儿子也没办法从我这儿学到这一点，所以虽然我都七十多了还得在这儿做轮子。因此我想说的是，古人和他们那些不能言传的东西想必也早一起都死去了，所以桓公所读的，不过是古人留下来的糟粕而已！"

提 示

这则典故出自《庄子外篇 · 天道十三》，叫做《轮扁斫轮》，原文如下："桓公读书于堂上，轮扁斫轮于堂下，释椎凿而上，问桓公曰："敢问：公之所读者，何言邪？"公曰："圣人之言也。"曰："圣人在乎？"公曰："已死矣。"曰："然则君之所读者，古人之糟粕已夫！"桓公曰："寡人读书，轮人安得议乎！有说则可，无说则死！"轮扁曰："臣也以臣之事观之。斫轮，徐则甘而不固，疾则苦而不入，不徐不疾，得之于手而应于心，口不能言，有数存焉于其间。臣不能以喻臣之子，臣之子亦不能受之于臣，是以行年七十而老斫轮。古之人与其不可传也死矣，然则君之所读者，古人之糟粕已夫！"

文中没有写明轮扁老师傅结局如何，但是想来老人家还是有极大的几率继续给桓公做车轮子的，轮扁（当然其实是庄子老人家借轮扁的口）讲的很有道理，找个熟练工不容易啊。

读到这里，我估摸着有些朋友已经在暗自嘀咕：看我这意思，似乎是在宣扬读书无用论了哟，出版社负责审校的同学，恐怕也已准备着把我这段序文删掉。都别着急，毛主席一直教导我们，要用辩证的眼光看待问题。古人说的不一定都对，我觉着问题的关键不在于看还是不看书，关键点首先是所阅读的图书质量，作者有没有把要表达的意思阐述清楚，其次是读者们有没有认真阅读，独立思考，真正领会作者想要表达的思想。

尽管时下写文字有种种限制，但是得益于近些年出版行业发达，现如今世面上讲经验、谈技巧、摆案例类的图书纷杂涌现，对于 IT 技术领域这类图书就更多了（因为 IT 行业本就是门实践性很强的技术）。在书中应对案例中出现的故障，作者们自己往往驾轻就熟，挥洒之间数千字，似乎也讲的透彻，但是初学者朋友可能看的云里雾里似懂非懂而不自知。仿佛学到了什么，但真正应对故障时却手忙脚乱，场景稍有变化甚至都不知道从何处着手处理。这种情况若对应到《轮扁斫轮》这则典故，说明操作者还没有达到"不徐不疾，得之于手而应于心"的境界。

读者朋友们认真看过书中的内容，可是实际工作中却不能很好的应用，这究竟是什么缘故呢，我想大概就是前面所说的两点关键因素，详细说来如下：

- 其一：可能作者没有（想或不想）把真正的精髓写出来。目前比较畅销的 IT 图书，其作者大都

是来自一线的工程师，随着这些年图书出版门槛的降低，很多人有机会能将自己工作学习过程中积累的经验写出来（我认为这也是国内 IT 图书中少见思想类图书的原因）。优秀的 IT 工程师都是出色的实践者，他们技术掌握的比较扎实，接触面广，经验丰富，当遇到问题时，处理的方法往往都是下意识的选择，没有为什么，就是要这么做。在写作的时候也是下意识就将过程写了出来，而没能把思路阐释清楚。

- 其二：并非书写的不好，可能由于读者自身层次的原因，没能正确理解作者表达的精髓。对于读者来说，找到一本好书难，读透一本好书更难。过去有一种说法叫做："书读百遍，其意自现"，我觉着这点在 IT 技术领域的局限是很大的，虽然任谁也不能否认阅读的作用是巨大的，可是，正如我前面谈到的，IT 行业是一门实践性非常强的技术，按照过去的老话讲，IT 工程师也是个手艺人。因为行业的特点，看的懂和做的到是两码事，尽管每读一遍都会有新的理解，但是这种理解必须要与实践相结合才能发挥最大的威力。

对此，庄子老人家其实也早已高度抽象地概括为：视而可见者，形与色也；听而可闻者，名与声也。世人以形色名声为足以得彼之情。夫形色名声，果不足以得彼之情，则知者不言，言者不知，而世岂识之哉！

要想学的好，首先所读的书必须拥有比较高的质量，其次学习也必须能学到书中的精华，所以您瞧，有多种因素可能会给学习的质量造成干扰。再举这样一个案例，DBA 管理的系统出现响应慢的情况，通过分析发现是由于之前执行的某项操作，正是该操作占用了过多的资源才导致系统响应变慢，针对这种情况怎么处理呢。一定有些资料中提到，要杀掉占用过多资源的进程，以释放资源，提高系统的响应效率，并且有实际的案例佐证此方案的有效。于是在这个场景中，DBA 为了缓解系统负载压力，利用之前看过的材料中提到的方法，手动杀掉了持有该操作的进程。

若仅把所执行的操作为独立个体来看，这当然是个很好的案例，有可能系统负载立刻就得到了明显下降，但问题有没有得到真正解决呢？深层次的根源究竟又是什么呢？如果没有弄清楚这些情况，那么所做的操作有可能不起效果（这就算好消息了），甚至有可能充满了风险。因为不是所有占用较多资源的进程都是不正常的，也不是所有进程都能随便中止，不管遇到的是什么问题，能找出造成问题的关键所在最重要。元芳，你怎么看！

不管要学习哪方面的知识，在学习过程中可参考的资料会有很多，在互联网时代更是可以用浩瀚来形容，这种现状换个角度看反倒更令读者们无所适从，不知道该选择看哪些资料好。若让我来选择，首要推荐的仍然是官方提供的技术文档，对于 Oralce 数据库可以到 tahiti.oracle.com 浏览，对于 MySQL 数据库可以到 dev.mysql.com/doc 浏览，官方文档始终都是内容最权威、最全面的学习资料，恒久远永流传。对于有一定经验的朋友，可能会认为官方文档的深度不够，案例也少，对于这部分朋友，可以去看一些专门的文章和图书，在选择图书时只有一个准则，就是要读有口碑的书。目前各大网络商城都有评分和评论系统，购买前先看一下其他用户的意见作为参考会很有帮助。

我读过的很多图书，文章开篇总要吹些牛皮，吸引读者的眼球，把读者的胃口调起来，希望大家能有兴趣接着往后看（欢迎对号入座）。像三思这种开篇不仅自我贬低，且一枪打击一大片的，怕着实不多见，我想这跟我的性格有关——耿直（好吧！我承认其实是情商低），也跟我所从事的职业有关——技术，来

不得半点儿虚假。

好了，讲到这里，是时候跟大家介绍下本书的内容了（分明是要开始王婆卖瓜自卖自夸），这并不是一本逐个介绍 MySQL 命令的书，不是一本用户帮助手册，不是这个功能讲完讲那个功能的书。在写作之初我就考虑要设定一条主线，不是依次讲特性，而是依据用户接触和学习 MySQL 的脉络去把握，介绍 MySQL 数据库从无到有，其安装、配置、管理、优化的过程，在这个过程中再穿插数据导入导出、性能/状态监控、备份恢复和优化方面的内容，最后再谈一谈 MySQL 数据库服务从单台到多台，从单实例到多实例集群的部署等稍显高阶的应用方案。

说起来，这其实是一本站在初学者的视角，描述他不断学习和提高的路径的图书，在这个过程中，我当然不可能面面俱到地讲到所有的技术特性，不过在介绍某些知识点时，会有意地忽略一些细节，是希望能让读者有思考的空间，既能看到优势同时也学会看到不足，找出更适合自己的解决方案，逐渐形成自己的操作思路，窥见“不徐不疾，得之于手而应于心”的境界。

最后，我想说的是，官方文档也好，技术图书也好，这些都是外在因素，最重要的因素仍然是自己，是否真正喜欢所要学习的技术，是否确实愿意花费时间和精力去深入研究，是否能够承受枯燥的应用和测试。只要打好了基础，看多了案例，精通了技能，学好了本领，明了方方面面前因后果，用不了多久，就可以成为大拿，升职加薪，当上技术总监，出任 CTO，迎娶白富美，登上人生顶峰！是不是想想都激动啊！小伙伴儿们，那就从现在开始吧，翻开第一页，MySQL 在向你招手。

目　录

第1章 开源运动与开源软件 MySQL

20 世纪 80 年代，传奇人物 Richard Stallman 发起 GNU 和 Free Software（自由软件）运动时，应该不会想到，他所开创的这种全新的软件开发、使用、传播模式，会在之后的 20 多年里，为整个 IT 行业的发展注入强大的活力，并且在可预见的未来数十年中，仍将继续影响甚至颠覆 IT 行业的方方面面。

1.1 开源软件的故事

在 PC（Personal Computer，个人电脑）这个概念刚刚出现的 20 世纪六七十年代，也就是 Bill Gates 憋在自己家里开发他的 MS-DOS，Steve Jobs 则跟他的朋友们在车库打磨苹果 I 号的年代，尚没有开源软件或商业软件的概念，计算机的操作用户都还是非常小众的群体，就更别提软件的开发者们了。计算机的平台几乎都不通用，各种特定硬件上运行的软件，其使用者多数情况下也正是其开发者，都是为了特定需求在特定平台上实现的特定功能。有时候呢，运气好，使用相同硬件的“科研”爱好者们会凑到一起，呃，虽然没有亲身经历，但我想应该跟时下 IT 技术圈子聚会没有什么本质差异，大家相互间分享些心得，保不齐还会把自己写的，虽不成熟但自我感觉良好的程序拿出来吹一吹。

当时的潮流是提供整套服务，买硬件附带送软件（貌似现如今的大型机平台仍是如此，并且通用产品领域也有朝此发展的迹象，比如说 Oracle 正大力推广的 Exadata，难道这就是传说中的返祖）。对于用户来说，在使用过程中可以对软件进行修改，并且可能的情况下也会将其修改版发布共享出来，甚至会相互合作开发功能。

总之，软件使用时自由度非常之高，很多一流的开发者也热衷于分享，代表人物中有位当时还名不见经传的程序员——Richard Stallman，在他就职于 MIT（Massachusetts Institute of Technology，麻省理工学院）的人工智能实验室时，就曾于 20 世纪 70 年代发起过一项代码共享运动。Stallman 当时提出的观点就是希望常用的代码能够在程序员之间共享，这样开发人员可以在相当大的范围内彼此合作。这一理念初期贯彻得很不错，在此期间 Stallman 本人也开发出多种影响深远的软件，其中最著名的就是 Emacs，这是一款功能强大的文本编辑器软件，对 Linux/UNIX 较熟悉的朋友应该听说过它，因为通常来说，文件编辑工具若不是使用 Vim，有极大概率就是使用 Emacs。

可是，潮流总是在变化，软件产业也是如此，它要发展、要前进，在过程中对于旧的

体系自然会产生两种影响，即好的影响或坏的影响。

直到 20 世纪 80 年代初，虽然能够支持多平台的商业软件仍不多见，但也已经开始崭露头角。对于商业软件来说，爱好者之间的这种共享和复制各种软件的行为，简直就是在抢人饭碗、断人财路，这当然是不可被接受的，于是出现了各种各样的反对声音。在这里不能不被提及的是微软公司的 Bill Gates 在一封信中提到，“大多数爱好者之间，通过相互共享和复制来使用各种软件，并不在乎软件开发者的利益，但是好的软件必须得到应有的收入来保证质量”，这一观点微软秉持了数十年。多年来，微软引领了软件商业化潮流并且颇为成功，给许多软件开发者和软件开发企业树立了强大的标杆效应。加上 PC 这种通用平台的流行，不出售源代码的商业软件开始大量涌现。就连 MIT 人工智能实验室的许多工程师，也因为各种原因转投到商业软件公司就职，甚至其中某些人还组建公司，加入到商业软件大潮中。这股浪潮对于 IT 行业的影响如何暂且不论，起码对于擅长编程的工程师们来说，属于他们的春天来了，自己开发出的东西居然能卖钱（“优秀软件不仅值钱，而且很值钱”Bill Gates 深有体会地说），编程在当时可是很吃得开，就算没心思自己创业，加盟商业软件公司，饭碗那是不愁的。

提 示

其实如今也一样，优秀的程序员不管何时都不必发愁生计。唯一的区别在于，当初国外管这群人叫 nerd，现如今国内管这部分群体叫屌丝（我主动对号入座）。尽管前后隔了几十年，语种也不相同，但意思都差不多，多少一线奋战的 IT 民工们闻听这个消息泪流满面，激动万分又欢欣雀跃，这个“荣耀”属于他们，活生生的事实说明，中国的软件产业终于也和世界接上轨了。

不是每个人都认同这股潮流，每个时代都有“唐吉诃德”（此处为褒义），像商业软件采取闭源的方式，禁止随意修改和传播，在他们眼中就是在禁锢思想、阻碍自由。这里不得不又提 Richard Stallman，尽管如今他在软件开发领域拥有宗师的地位，在当年他却很是孤独，甚至连身边 AI 实验室的同事，基本都被挖去开发商业软件了，但 Stallman 没有随波逐流，而是与现状积极抗争。一开始他只是小打小闹，比如说某家公司说不开放源码了，Stallman 就会为它的竞争对手写程序，帮助他们加入新功能来打击那家公司，这种操作方式真的……很 nerd。

不过很快他自己也意识到这种方式不妥，Richard Stallman 是有坚定理想的，他想要改变这股潮流，他想要编写一套完全开放的操作系统和运行环境，让所有人都可以享受到软件自由，于是有了著名的 GNU 宣言。有些文章中提到这一节时，用了更辉煌的词汇来形容，说他是“出于对自由的崇尚和对软件商业化的痛恨，期望重现当年软件界合作互助的团结精神”。这个听起来是理想主义的调调，但是 Stallman 全身心地为之投入，此后他又发起成立了非盈利组织——自由软件基金会（Free Software Foundation，FSF）。

在开源软件的历史长河里，不能忽略以下这几个关键名词，通过对它们的解释和述说，就能折射出开源软件史上一个个闪光点，让我们更好地熟悉和理解开源软件的发展进程。

1.1.1 GNU说，我代表着一个梦想

GNU的全称是Gnu's Not UNIX，看名称就知道与UNIX脱不开关系，事实也确实如此，GNU不管是形式上还是实际上都与UNIX有很大的关联。

话说甭管什么样的设备，如大型机、中型机或小型机，对于终端用户来说，要想操作它都离不开操作系统。而且在当时，操作系统并不仅仅是内核，还包括编译器、电子邮件、各类常用工具等，那会儿UNIX占据着主流地位，可是当商业软件的浪潮袭来，UNIX开始要收费了（而在之前则是免费发布甚至开放源码的，由此还产生众多UNIX发行版，比如著名的BSD系统）。

源码不再开放，大家不能再随意改写，使用也要受诸多许可限制。于是Stallman立下宏愿，要推出一套Free的系统（这里提到的Free，意义可能跟大家的理解不同，后面会阐述这方面的背景）。

开发一套完整的操作系统是项极其宏大的工作，单单实现它就已经很不容易了，开发出来还要有人愿意用，这就更有难度了。君不见微软那么大势力，投入这么多资源搞了Windows Phone，各大手机厂商还不是该用安卓用安卓，用iOS的是iPhone。考虑到在当时UNIX系统占据的主流地位，为了能够让UNIX用户使用GNU时快速上手和平滑过渡，新系统不仅要Free，还得尽可能地兼容UNIX。当然这个兼容只是两者看起来像，实质还是有差异的，就如同WPS为了占领市场，也兼容微软Office的文件格式，但它跟Microsoft Office完全是两套系统一个道理。

Richard Stallman是个理想主义者，但他同时也是个实干派，即使那会儿没有“总理”对他谆谆教诲，他也知道仰望星空，脚踏实地。为了发展这个类UNIX的操作系统，Richard Stallman以及由他发起的FSF，开始收集及开发组成系统的各种必备软件，包括库、编译器、调试工具、编辑器等，准备工作做了很多，输出的各类软件也不少，但是系统内核的开发较为缓慢。最初，他们开发出了一套内核，名叫GNU HURD，但是并不太理想，直到一位芬兰的大学生——Linus Torvalds，开发出Linux（最初名叫Freax），他们才算具备一个基本可用的底层环境，等到1.0版本的Linux正式发布时，时间的指针已经指向了1994年。这期间波折很多，细节这里不详细阐述，推荐两部影片——代码和操作系统革命，从中可以看到一些有意思的故事。

那么Linux和GNU有什么关系呢？简单来讲，Linux是操作系统内核（并不是一个完整的系统哟），上面运行着众多GNU程序（GNU程序如今不仅运行在Linux环境上，UNIX/Windows平台上也都有），它们共同组合出一套可用的系统，因此FSF（尤其是Richard Stallman本人）将这套组合定义为GNU/Linux。只是这个命名存在争议，有一部分Linux发行版，比如Debian采用了“GNU/Linux”的说法，但更多企业以及开发人员还是直接称其为Linux。

关于名称，我个人觉着GNU项目发展到现在，无需再用什么来为其正名，它的存在