Dennis与Ken爷爷的UNIX/C世界

沉寂了很久了，时间在不断地逝去,转眼又到了新的一年，2013的发生了太多，Beta版本、辞职、职位转换、ARM、Driver、初级厨艺、Dx11、GPU、CPU、登山、GNU/Linux、Cross-Platfrom Tool Chain、GLES、Android。。。难以计算，日子依然忙碌，孑然一身，这个世界于我依然有太多未知，想法却是越来越少了，也不知到底是好是坏，愿始终做一块海绵，继续努力行走下去！

最近正被Linux折腾的头疼，顺道转一篇关于UNIX/C世界的前世今生，
默默地向Dennis与Ken等伟大的先驱者们致敬以及自勉。

---

http://bbs.weiphone.com/read-htm-tid-6209622.html
首先说明本帖是转的，旨在让大家了解科技界的一些故事，每一种科技的进步都伴随开发者们不懈的努力！就是想让大家明白一个道理不管unix、linux、ios还是android都是开发者辛勤汗水的积累。大家可以看附件里面的UNIX家族谱，若你没有惊叹唏嘘的。。。

Unix是目前还在存活的操作系统的元老了，走过了40年的历程。由它引发的思想变革，对当今计算机文化造成的深远影响。这是一段所有从事计算机行业人员尤其是软件开发人员需要了解的历史。Unix的传奇历史是整个计算机世界文化最具代表性的，它对整个计算机世界文化的影响也是最巨大，最深远的。他给人带来的不单单的对过去的回味，更为我们带来了计算机世界的新思潮。

Unix 起源

回顾Unix历史，我们就要说一下一个叫MULTICS的项目。上世纪六十年代时，大部份计算机都是采用批处理的方式（也就是说，当作业积累一定数量的时候，计算机才会进行处理）。那时，我们熟知的美国电话及电报公司（AT&T）、通用电器公司（G。E。）及麻省理工学院（MIT）计划合作开发一个多用途、分时及多用户的操作系统，也就是这个MULTICS，其被设计运行在GE-645大型主机上。不过，这个项目由于太过复杂，整个目标过于庞大，糅合了太多的特性，进展太慢，几年下来都没有任何成果，而且性能都很低。于是到了1969年2月，贝尔实验室决定退出这个项目。

熟悉这段历史的人都知道，贝尔实验室中的有个叫Ken Thompson的人，他为MULTICS这个操作系统写游戏了个叫“Space Travel”的游戏，在MULTICS上经过实际运行后，他发现游戏速度很慢而且耗费昂贵 —— 每次运行会花费75美元。退出这个项目以后。他为了让这个游戏能玩，所以他找来Dennis Ritchie为这个游戏开发一个极其简单的操作系统。这就是后来的Unix。（值得一提的是，当时他们本想在DEC-10上写，后来没有申请到，只好在实验室的墙角边找了一台被人遗弃的Digital PDP-7的迷你计算机进行他们的计划，这台计算机上连个操作系统都没有，于是他们用汇编语言仅一个月的时间就开发了一个操作系统的原型）他们的同事Brian Kernighan非常不喜欢这个系统，嘲笑Ken Thompson说：“你写的系统好真差劲，干脆叫Unics算了。”Unics的名字就是相对于MULTICS的一种戏称，后业改成了Unix。于是，Unix就在这样被游戏和玩笑创造了，当时是1969年8月。也就是这一年，Linux之父Linus Torvalds在芬兰出生了。

1971年，Ken Thompson写了充分长篇的申请报告，申请到了一台PDP-11/24的机器。于是Unix第一版出来了。在一台PDP-11/24的机器上完成。这台电脑只有24KB的物理内存和500K磁盘空间。Unix占用了12KB的内存，剩下的一半内存可以支持两用户进行Space Travel的游戏。而著名的fork（）系统调用也就是在这时出现的。

到了1973年的时候，Ken Thompson 与Dennis Ritchie感到用汇编语言做移植太过于头痛，他们想用高级语言来完成第三版，对于当时完全以汇编语言来开发程序的年代，他们的想法算是相当的疯狂。一开始他们想尝试用Fortran，可是失败了。后来他们用一个叫BCPL的语言开发，他们整合了BCPL形成B语言，后来Dennis Ritchie觉得B语言还是不能满足要求，就是就改良了B语言，这就是今天的大名鼎鼎的C语言。于是，Ken Thompson 与Dennis Ritchie成功地用C语言重写了Unix的第三版内核。至此，Unix这个操作系统修改、移植相当便利，为Unix日后的普及打下了坚实的基础。而Unix和C完美地结合成为一个统一体，C与Unix很快成为世界的主导。

Unix的第一篇文章 “The UNIX Time Sharing System”由Ken Thompson和Dennis Ritchie于1974年7月的 the Communications of the ACM发表。这是UNIX与外界的首次接触。结果引起了学术界的广泛兴趣

公告

昵称： Zephyroal
园龄： 14年9个月
粉丝： 108
关注： 60
-取消关注

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

积分与排名

积分 - 148749
排名 - 8409

随笔分类

C++Basic(12)
Mobile—Android/iOS/WP(2)
Ogre-Fight(19)
Others(20)
ShaderNewTech(7)
Shader之旅：一天干掉一只Monkey计划(6)
ThinkCareer(40)
Unreal3_Way(25)
翻译(2)

随笔档案

2019年3月(1)
2014年12月(1)
2014年9月(1)
2014年1月(1)
2013年9月(1)
2013年7月(3)
2013年4月(1)
2013年3月(2)
2013年2月(1)
2013年1月(1)
2012年11月(3)
2012年8月(1)
2012年7月(1)
2012年6月(4)
2012年5月(8)
更多

阅读排行榜

1. 世界十大绝美花海(19225)
2. Lua脚本在C++下的舞步(入门指引)（…
3. 十分淫霸的Mooege&MadCow，你懂…
4. Cryengine渲染引擎剖析（转）(11130)

并对其源码索取，所以，Unix第五版就以“仅用于教育目的”的协议，提供给各大学作为教学之用，成为当时操作系统课程中的范例教材。各大学公司开始通过Unix源码对Unix进行了各种各样的改进和扩展。于是，Unix开始广泛流行。

Unix分裂

1978年，对 Unix而言是革命性的一年；因为学术界的老大柏克利大学，推出了一份以第六版为基础，加上一些改进和新功能而成的 Unix。这就是著名的“1 BSD（1st Berkeley Software Distribution）”，开创了Unix的另一个分支：BSD 系列。 同时期，AT&T成立USG，将 Unix变成商业化的产品。从此，BSD的 Unix 便和AT&T 的Unix 分庭抗礼，Unix就分为System IV和4。x BSD这两大主流，各自蓬勃发展。

1979年发布的Unix 第七版被称为是“最后一个真正的Unix”，这个版本的Unix内核只有40K bytes。后来这个版本被移植到VAX机上（我在大学时学习C语言时用过这个VAX机，我还记得那时上VAX机最大的爱好就是使用talk命令和别人聊天，呵呵）。20世纪80年代相继发布的8、9、10版本只授权给了少数大学。

1982年，AT&T基于版本7开发了UNIX System III的第一个版本，这是一个商业版本仅供出售。为了解决混乱的UNIX版本情况，AT&T综合了其他大学和公司开发的各种UNIX，开发了UNIX System V Release 1。这个新的UNIX商业发布版本不再包含源代码，所以加州大学Berkeley分校继续开发BSD UNIX，作为UNIX System III和V的替代选择。BSD对UNIX最重要的贡献之一是TCP/IP。BSD 有8个主要的发行版中包含了TCP/IP：4。1c、4。2、4。3、4。3-Tahoe、4。3-Reno、Net2、4。4以及 4。4-lite。这些发布版中的TCP/IP代码几乎是现在所有系统中TCP/IP实现的前辈，包括AT&T System V UNIX 和Microsoft Windows中的TCP/IP都参照了BSD的源码。

同时，其他一些公司也开始为其自己的小型机或工作站提供商业版本的UNIX系统，有些选择System V作为基础版本，有些则选择了BSD。BSD的一名主要开发者，Bill Joy，在BSD基础上开发了SunOS，并最终创办了Sun Microsystems。

1991年，一群BSD开发者（Donn Seeley、Mike Karels、Bill Jolitz 和 Trent Hein）离开了加州大学，创办了Berkeley Software Design， Inc （BSDI）。BSDI是第一家在便宜常见的Intel平台上提供全功能商业BSD UNIX的厂商。后来Bill Jolitz 离开了BSDI，开始了386BSD的工作。386BSD被认为是FreeBSD、OpenBSD 和 NetBSD、DragonFlyBSD的先辈。

这是一个AT&T妄图私有化的Unix的时代。为了私有化Unix，1986年IEEE指定了一个委员会制定了一个一个开放作业系统的标准，称为 POSIX （Portable Operating Systems Interface）。最后加上个X，不知道是为了好听，还是因为这本质上是UNIX的标准。当然，AT&T的Unix取得了这个标准制订战争的胜利，还取得了Unix这个注册商标。此时BSD的拥护者自喻为冷酷无情的公司帝国的反抗军。就销售量来说，AT&T UNIX始终赶不上BSD/Sun。到1990年，AT&T与BSD版本已难明显区分，因为彼此都有采用对方的新发明。

这段时期，从实验室出来的被全世界所分享的Unix，正处于被私有化的关键时期。

Unix的法律纠纷

BSDI很快就与AT&T的UNIX Systems Laboratories（USL）附属公司产生了法律纠纷，USL是AT&T注册的公司。AT&T为了拥有System V版权，以及Unix商标，为了垄断Unix，1992年，USL正式对BSDI提起诉讼，说BSD剽窃他的源码。而最终了结了好评如潮的BSD系统。

由于最后判决悬而未决，这桩法律诉讼将BSD后裔的开发，特别是自由软件，延迟了两年，这导致没有法律问题的Linux内核获得了极大的支持。Linux跟386BSD的开发几乎同时起步，Linus说，当时如果有自由的基于386的Unix-like操作系统，他就可能不会创造Linux。尽管无法预料这给以后的软件业究竟造成了什么样的影响（如果没有这个法律纠纷，很有可能没有今天的革命性的Linux），但有一点可以肯定，Linux更加丰富了这块土壤。

这场官司一直打到 AT&T将自己的Unix系统实验室卖掉，新接手的Novell公司采取了一种比较开明的做法，允许BSDI自由发布自己的BSD，但是前提是必须将来自于AT&T的代码完全删除，于是诞生了4。4 BSD Lite版，由于这个版本不存在法律问题，4。4BSD Lite成为了现代BSD系统的基础版本。

这桩诉讼最终在1994年1月了结，更多地满足了BSDI的利益。伯克利套件的18，000个文件中，只有3个文件要求删除，另有70个文件要求修改，并显示USL的版权说明。这项调解另外要求，USL不得对4。4BSD提起诉讼，不管是用户还是BSDI代码的分发者。于是，BSD Unix走上了复兴的道路。BSD的开发也走向了几个不同的方向，并最终导致了FreeBSD、OpenBSD和NetBSD的出现。

5. 全面介绍Windows内存管理机制及C++…

评论排行榜

1. 翻一下老黄历，大学毕设写的基于Ogr…
2. UE3代码阅读需知(12)
3. 一天干掉一只Monkey计划（四）——…
4. MyGUI中文完美解决方案(7)
5. MMO之禅（一）开论(5)

推荐排行榜

1. Lua脚本在C++下的舞步(入门指引)（…
2. 谈谈时间管理--陶哲轩(3)
3. 程序员的十层楼(2)
4. 24小时极限挑战WPF：LOLVoiceExtra…
5. Dennis与Ken爷爷的UNIX/C世界(1)

最新评论

1. Re:一天干掉一只Monkey计划（四）—…
博主尝试过Markdown吗，有没有时间重新整理一下
--本本熊
2. Re:UDK中Kismet处理输入及脚本输入…
很有用，非常感谢博主的分享！
--昨夜晨风
3. Re:一天干掉一只Monkey计划（四）—…
楼主有相关的rendermoney 工程文件参考下啊。。。有的话，麻烦发我份，邮箱 nevermorewish@163.com
--妹子你要么
4. Re:UE3代码阅读需知
@ gandalfzyq引用@Zephyroalvtune确实是好，但是没有源代码吧？……最理想的情况下是Unity里面的profiler那种（可以直接实时显示出来不同的函数的cpu占用比例），但是感...
--Zephyroal
5. Re:UE3代码阅读需知
@ Zephyroalvtune确实是好，但是没有源代码吧？……最理想的情况下是Unity里面的profiler那种（可以直接实时显示出来不同的函数的cpu占用比例），但是感觉实现起来似乎有点难度，（...
--gandalfzyq

从AT&T意识到了Unix的商业价值，不再将Unix源码授权给学术机构以来，到以后的几十年，Unix仍在不断变化，其版权所有者不断变更，授权者的数量也在增加。Unix的版权曾经为AT&T所有，之后Novell拥有了Unix，再之后Novell又将版权出售给了SCO（这一事实双方尚存在争议，这里是最新进展）。有很多大公司在取得了Unix的授权之后，开发了自己的Unix产品。（几年前，据传闻微软为了限制Linux，微软让SCO到法院告Linux剽窃其源码）

由于Unix是由C语言写的，所以修改和移植都很容易，因此，很多商业公司及学术机构均加入这个操作系统的研发，各个不同版本的Unix也开始蓬勃发展。这才产生了今天这么多的各式各样的Unix衍生产品。如AIX、Solaris、HP-UX、IRIX、OSF、Ultrix等等。（这些商业化的Unix基本上都是源于AT&T授权的Unix System V）

Unix开源组织

AT&T的这种商业态度，让当时许许多的Unix的爱好者和软件开发者们感到相当的痛心和忧虑，他们认为商业化的种种限制并不利于产生的发展，相反还能导制产品出现诸多的问题。随着商业化Unix的版本的种种限制和诸多问题，引起了大众的不满和反对。于是，大家开始有组织地结成“反叛联盟”以此对抗欺行罢市的AT&T等商业化行为。

另一方面，关于“大教堂”（集权、封闭、受控、保密）和“集市”（分权、公开、精细的同僚复审）两种开发模式的对比成为了新思潮的中心思想。这个新思潮对IT业产生了非常深远影响。为整个计算机世界带来了革命性的价值观。

此时，一个名叫Richard Stallman的领袖出现了，他认为Unix是一个相当好的操作系统，如果大家都能够将自己所学贡献出来，那么这个系统将会更加的优异！他倡导的Open Source的概念，就是针对Unix这一事实反对实验室里的产品商业化私有化。尽管Stallman既不是、也从来没有成为一个Unix程序员，但在后1980的大环境下，实现一个仿Unix操作系统成了他追求的明确战略目标。Richard Stallman早期的捐助者大都是新踏入Unix土地的老牌ARPANET黑客，他们对代码共享的使命感甚至比那些有更多Unix背景的人强烈。

为了这个理想，Richard Stallman于1984年创业了GNU，计划开发一套与Unix相互兼容的的软件。1985 年 Richard Stallman 又创立了自由软件基金会（Free Software Foundation）来为 GNU 计划提供技术、法律以及财政支持。尽管 GNU 计划大部分时候是由个人自愿无偿贡献，但 FSF 有时还是会聘请程序员帮助编写。当 GNU 计划开始逐渐获得成功时，一些商业公司开始介入开发和技术支持。当中最著名的就是之后被 Red Hat 兼并的 Cygnus Solutions。

GNU组织的建立，延续了当年Unix刚出现时的情形，并为这种情形建立了可靠的法律和财务保障。GNU 工程十几年以来， 已经成为一个对软件开发主要的影响力量， 创造了无数的重要的工具。例如：强健的编译器，有力的文本编辑器，甚至一个全功能的操作系统。从那时开始，许多程序员聚集起来开始开发一个自由的、高质量、易理解的软件，让这使得Unix社区生机勃勃，一派繁荣景象。

自90年代发起这个计划以来，GNU 开始大量的产生或收集各种系统所必备的组件，像是——函数库、编译器、调式工具、文本编辑器、网站服务器，以及一个Unix的使用者接口（Unix shell）等等，等等。但由于种种原因，GNU一直没有开发操作系统的kernel。正当Richard Stallman在为操作系统内核伤脑筋的时候，Linux出现了。

Linux横空出世

1990年，Linus Torvalds还是芬兰赫尔辛基大学的一名学生，最初是用汇编语言写了一个在80386保护模式下处理多任务切换的程序，后来从Minix（Andy Tanenbaum教授所写的很小 的Unix操作系统，主要用于操作系统教学）得到灵感，进一步产生了自认为狂妄的想法——写一个比Minix更好的Minix，于是开始写了一些硬件的设备驱动程序，一个小的文件系统。这样0。0。1版本的Linux就出来了，但是它只具有操作系统内核的勉强的雏形，甚至不能运行，你必须在有Minix的机器上编译以后才能玩。这时候Linus已经完全着迷而不想停止，决定踢开Minix，于是在1991年10 月5号发布Linux 0。0。2版本，在这个版本中已经可以运行bash 和gcc。

从一开始，Linus就决定自由扩散Linux，包括原代码，随即Linux引起黑客们（hacker）的注意，通过计算机网络加入了Linux的内核开发。Linux倾向于成为一个黑客的系统——直到今天，在Linux社区里内核的开发被认为是真正的编程。由于一批高水平黑客的加入，使Linux 发展迅猛，几乎一两个礼拜就有新版或修正版的出现，到1993年底94年初，Linux 1。0终于诞生了！Linux 1。0已经是一个功能完备的操作系统，而且内核写得紧凑高效，可以充分发挥硬件的性能，在4M内存的80386机器上也表现得非常好，至今人们还在津津乐道。时至今日，kernel的版本已经出到2。6。Linux的发展不像传统的软件工程，它完全是透过网络，集合世界各地的高手而成的一套操作系统，在这里我们也可以见识到网络快速传播的威力。Linux初次让整个世界感觉到了开源力量和网络力量的如此强大。（Linux 的标志和吉

祥物是一只名字叫做 Tux 的 企鹅，标志的由来是因为Linus在澳洲时曾被一只动物园里的企鹅咬了一口，便选择了企鹅作为Linux的标志。）

Linux 的历史是和GNU紧密联系在一起的。从1983年开始的GNU计划致力于开发一个自由并且完整的类Unix操作系统，包括软件开发工具和各种应用程序。到1991年 Linux 内核发布的时候，GNU已经几乎完成了除了系统内核之外的各种必备软件的开发。在 Linus Torvalds 和其它开发人员的努力下，GNU组件可以运行于Linux内核之上。整个内核是基于 GNU 通用公共许可，也就是GPL（GNU General Public License，GNU通用公共许可证）的，但是Linux内核并不是GNU 计划的一部分。1994年3月，Linux1。0版正式发布，Marc Ewing成立了 Red Hat 软件公司，成为最著名的 Linux 分销商之一。

严格来讲，Linux这个词本身只表示Linux内核，但在实际上人们已经习惯了用Linux来形容整个基于Linux内核，并且使用GNU 工程各种工具和应用程序的操作系统（也被称为GNU/Linux）。基于这些组件的Linux软件被称为Linux发行版。一般来讲，一个Linux发行套件包含大量的软件，比如软件开发工具，数据库，Web服务器（例如Apache），X Window，桌面环境（比如GNOME和KDE），办公套件（比如OpenOffice。org），等等。

1991至1995年间，Linux从概念型的0。1版本内核原型，发展成为能够在性能和特性上均堪媲美专有Unix的操作系统，并且在连续正常工作时间等重要统计数据上打败了这些Unix中的绝大部分。1995年，Linux找到了自己的杀手级应用——开源的web服务器Apache。就像Linux，Apache出众地稳定和高效。很快，运行Apache的Linux机器成了全球ISP平台的首选。约60%的网站选用Apache，轻松击败了另两个主要的专有型竞争对手。今天的LAMP（Linux ， Apache， MySQL， PHP）已经成为了架构Web服务器的主要首选。

现如今的Linux不但可以装在几乎所有的主流服务器上，当然也包括桌面的X86系统中。其还常常被用于嵌入式系统，机顶盒、手机、交换机、游戏机、PDA、网络交换机、路由器、等等，都是因为Linux那精彩的内核。

Linux的出现，不仅仅给世界带来了一个免费的操作系统，也不仅仅是对Unix自由、共享的文化的延续，它的出现带给了计算机世界自Unix、GNU以来更为成熟的思想和文化。

Linux今天的领袖

Linux和GNU关系是比较微妙的。那时，自由软件基金会编写的用户软件工具包铺平了一条摆脱高成本专有软件开发工具的前进道路。意识服从经济，而不是领导：一些新手加入了RMS的革命运动，高举GPL大旗，另一些人则更认同整体意义上的Unix传统，加入了反对GPL的阵营，但其他大部分人置身事外，一心编码。

Linus Torvalds巧妙地跨越了GPL和反GPL的派别之争。他利用GNU工具包搭起了自创的Linux内核，用GPL的传染性质保护它，但拒绝认同Richard Stallman的许可协议反映的思想体系计划。Linus Torvalds明确表示他认为自由软件一般情况下更好，但他偶尔也用专有软件。即使在他自己的事业中，他也拒绝成为狂热分子。这一点极大地吸引了大多数黑客，他们虽然早就反感Richard Stallman的言辞，但他们的怀疑论一直缺个有影响力或者令人信服的代言人。而Linus Torvalds正好充当了这一角色。

Linus Torvalds令人愉快的实用主义及灵活而低调的行事风格，促使黑客文化在1993至1997年间取得了一连串令人惊奇的胜利，不仅仅在技术上的成功，还让围绕Linux操作系统的发行、服务和支持产业有了坚实的开端。结果，他的名望和影响也一飞冲天。Torvalds成为了互联网时代的英雄；到1995年为止，他只用了四年时间就在整个黑客文化界声名显赫，而Richard Stallman为此花了十五年，而且他还远远超过了Stallman向外界贩卖“自由软件”的记录。与Torvalds相比，Richard Stallman的言辞渐渐显得既刺耳又无力。（参看《Linus Torvalds 语录 Top 10》）

今天，我们也说不清楚是GNU Linux还是Linux GNU。Linux既不排斥开源，也不排斥商业化，Linus认为好的软件是需要免费和商业化共同推进的。正是这种革命性的想法，造就了今天的Linux火红的局面（参看《谁写了Linux》、《Linux基金会的广告》、《Linux Distribution Timeline》）。Linux就像一股清泉流入了所有人的心中，引发了很多的启迪和思考。

Unix与黑客文化

黑客的文化和Unix的商业化存在着必然的联系。自从Unix出现，黑客文化就与之而来。

1993初，一个悲观的观察家撰文指出，已经有理由认为Unix的传奇故事连同他带有黑客文明将一同破产。许多人预测，从那时起Unix将在六月内死亡。他们很清楚，十年的Unix商业化，使自由跨平台的Unix梦以失败告终。Unix允诺的跨平台可移植性，在一打大公司专有的Unix版本之间不停地斗嘴中丢失，一个完美的操作系统最终沦为多种版本的一团乱麻，这应该说是人类文明史上的一个重大悲剧。

在专有软件社会中，只有像微软一样的“集权制，大教堂”生产方式才能成功。那个时代的人悲观地相信，技术世界的个人英雄主义时代已经结束，软件工业和发展中的互联网络将逐渐地由像微软一样的巨型企业支配，再也没有“佐罗”，世界是恺撒大帝的世界，计算机文明将进入黑暗的帝国时代。黑客已经死了，自由不付存在。

自从Unix出现以来，第一代的Unix黑客似乎垂垂老矣，衣食不饱（ Berkeley计算机科学研究组在1994丢失了自己基金）。这是一个抑压的时代。专有的商业Unix的结果证明那么沉重、那么盲目、那么不适当，以致微软能够用那次等技术的Windows抢走他们生存的空间，拿走他们的干粮。黑客世界的残余力量被逼到了世界上的角落里，苟延残喘。

就在黑客文化日渐衰落之时，美国新闻周刊的资深记者Steven Levy完成了著名的《黑客列传》一书，书中着力介绍了一个人物：Richard M。 Stallman的故事，他是麻省理工学院（MIT）人工智能实验室领袖人物，坚决反对实验室的研究成果商业化。他是商业软件社会中坚强的一员，决不随波逐流，建立了全新的黑客文化。

Richard M。 Stallman（他的登陆名RMS更为人熟知）早在1970年代晚期就已经证明他是当时最有能力的程序员之一。Emacs编辑器就是他众多发明中的一项。RMS的目标是将后1980的松散黑客社群变成一台有组织的社会化机器以达到一个单纯的革命目标。也许他未意识到，他的言行与当年卡尔·马克思号召产业无产阶级反抗工作的努力如出一辙。RMS宣言引发的争论至今仍存于黑客文化中。他的纲要远不止于维护一个代码库，已经暗含了废除软件知识产权主张的精髓。RMS通过“自由软件（free software）”让黑客文化更加有自我意识。当然，这个充满魅力又具争议的人物本身已经成为了一个黑客文化英雄。

只有痴迷的“黑客”和具有创造力的怪人结成的反叛联盟才能把我们从愚蠢中拯救出来——他们接着教导我们，真正的专业和奉献精神，正是我们在屈服于世俗观念的“合理商业做法”之前的所作所为。 ——《The Art of Unix Programming》

RMS让世界上所有的人都知道，入侵电脑系统只是低级不入流的黑客干的事，真正的黑客，是为了自由，为了软件的自由，为了挑战计算机世界中的霸权主义而斗争。他们不是街头小混混，他们更像是绿林好汉，更像是罗宾汉，更像是佐罗。就像渴望民主的人民同专制的政府斗争一样。RMS领导着许多的黑客通过互联网向专有软件发出宣战。

X Windows是首批由服务于全球各地不同组织的许多个人以团队形式开发的大规模开源项目之一。电子邮件使创意得以在这个群体中快速传播，问题由此得以快速解决，而开发者可以人尽其才。软件更新可以在数小时之内发送到位，使得每个节点在整个开发过程中步调一致。网络改变了软件的开发模式。

另一方面，RMS的理论体系有许多东西非常有争议，他的GPL被认为是一种“病毒式”的协议，BSD的fans和老牌Unix黑客们认为，他们编写Unix的年头都比GPL声明要长得多，GPL依然有太多的限制，而BSD协议则比GPL更加的自由。另一方面，RMS走向了另一个极端，他是完全反版权的，反商业化的。把软件产品从强制收费推向了强制免费、共享和开源，这也为他带来了许多许多的争议。

在RMS组织黑客闹革命的年代里，没有多少黑客认同于RMS的理论体系，更多的他们参与GNU只是为了体现那种在互联网上协同工作，令人激动的工作模式。自从GNU设立以来，争议不断，而黑客文化却从未有统一在他的理想体系之下。

自从Linux出现以后，一个新的黑客领袖出现了，Linus Torvalds的中庸态度网聚了世界上顶尖的黑客，其绕过了GPL和反GPL的派系之争，他使用GNU的工具从而以GPL的“传染性”保护了Linux，但他同时也不承认RMS的理论思想体系，他即开源，又支持商业化。虽然，他没有带给黑客们什么重要的思想体系或统一的价值观，但他整合了全世界黑客的阵营，让所有的黑客的行为都围绕着Linux这一事物进行。他以“用自由软件是因为它运行得更好”轻而易举地盖过了“用自由软件是因为所有软件都该是自由的”。

1998年初，这种新思潮促使网景公司（Netscape Communications）公布了其Mozilla浏览器的源码。媒体对此事件的关注促成了Linux在华尔街的上市，推动了1999－2001年间科技股的繁荣。事实证明，此事无论对黑客文化的历史还是对Unix的历史都是一个转折点。

Unix的历史教训

下面的文字出自《The Art of Unix Programming》（Unix编程艺术）。令今天我们所有人所反思。

在Unix历史中，最大的规律就是：（看看《谁写了Linux》你就会知道这一规律）

距开源越近就越繁荣。任何将Unix专有化的企图，只能陷入停滞和衰败。

回顾过去，我们早该认识到这一点。1984年至今，我们浪费了十年时间才学到这个教训。如果我们日后不思悔改，可能还得大吃苦头。

虽然我们在软件设计这个重要但狭窄的领域比其他人聪明，但这不能使我们摆脱对技术与经济相互作用影响的茫然，而这些就发生在我们的眼皮底下。即使Unix社区中最具洞察力、最具远见卓识的思想家，他们的眼光终究有限。对今后的教训就是：过度依赖任何一种技术或者商业模式都是错误的——相反，保持软件及其设计传统的的灵活性才是长存之道。

另一个教训是：别和低价而灵活的方案较劲。或者，换句话说，低档的硬件只要数量足够，就能爬上性能曲线而最终获胜。经济学家Clayton Christensen称之为“破坏性技术”，他在《创新者窘境》（The Innovator’s Dilemma）[Christensen]一书中以磁盘驱动器、蒸汽挖土机和摩托车为例阐明了这种现象的发生。当小型机取代大型机、工作站和服务器取代小型机以及日用Intel机器又取代工作站和服务器时，我们也看到了这种现象。开源运动获得成功正是由于软件的大众化。Unix要繁荣，就必须继续采用吸纳低价而灵活的方案的诀窍，而不是去反对它们。

最后，旧学派的Unix社区因采用了传统的公司组织、财务和市场等命令机制而最终未能实现“职业化”。只有痴迷的“黑客”和具有创造力的怪人结成的反叛联盟才能把我们从愚蠢中拯救出来——他们接着教导我们，真正的专业和奉献精神，正是我们在屈服于世俗观念的“合理商业做法”之前的所作所为。

Unix族谱

Unix的故事仍旧延续着……，许多网站也为这段历史留下记录。一个详细记录Unix历史的网站（http：//www.levenez.com/unix/），这个网站忠实记载着1969～2005 年Unix发展的大事，而且还有PDF 档案可供下载，上面有一个庞大的UNIX家族版本树，让人叹为观止。网站的首页陈列每个时期Unix的历史，也代表着无数工程师的心血与努力。

Unix的特点

现在的文献中提到Unix基本上是说，由Ken Thompson和Dennis Ritchie共同开发的。而通过历史我们也能发现，Unix的主要是由Ken Thompson写下的。但在学术界，Dennis Ritchie的名字往往被排在了Ken Thompson前面的。这就是因为，Dennis Ritchie不但发明了C语言，而且当时他设计Unix操作系统的设计思想，影响了整个世界，直到今天。

当时，他们开发UNIX，没有正式立项，是Ken Thompson和Dennis Ritchie等少数几个人偷偷干的，如果一切都要从头从新设计，那几乎是不可能的。所以，Unix吸取与借鉴了Multics的经验，如内核，进程，层次式目录，面向流的I/O，把设备当作文件，……等等。但是Unix在继承中又有创新，比如Unix采用一种无格式的文件结构，文件由字节串加组成。这带来两大好处：一是在说明文件时不必加进许多无关的“填充物”，二是任何程序的输出可直接用作其他任何程序的输入，不必经过转换。后面这一点叫做“管道”（piping），这就是Unix首创的。此外，像把设备当作文件，从而简化了设备管理这一操作系统设计中的难题，虽然不是UNIX的发明，但是实现上它采用了一些新方法，比Multics更高明一些。

下面是Unix的特点：（30多年过去了，这些东西早已变成经典）

◦Everything （including hardware） is a file

◦所有的事物（甚至硬件本身）都是一个的文件。

◦Configuration data stored in text

◦以文本形式储存配置数据。

◦Small， single-purpose program

◦程序尽量朝向小而单一的目标设计

◦Avoid captive user interfaces

◦尽量避免令人困惑的用户接口

◦Ability to chain program together to perform complex tasks

◦将几个程序连结起来，处理大而复杂的工作。

Unix的影响和哲学

Unix是第三次工业革命中计算机软件领域最具代表性的产物。在这近40年中，由Unix造成的影响是最有深远意义的。就我看来，Unix为软件领域带来了至少以下有积极的东西，由这些东西所引发的直接或间接的事物更是举不胜数。

软件开发的若干哲学和思想。

全民参与推动软件，代码共享的模式。

开启了黑客文化和开源项目。

免费和商业的完美结合的Linux。

C语言，而后发展的C++，Java等等类C的语言和脚本。（参看《C语言的演变史》）

TCP/IP，其的Socket编程已成为今天通用的网络编程主流。（参看《到处都是Unix的胎记》）

不能不说，AT&T虽然发展了Unix，但今天Unix的混乱的局面也和AT&T 有着直接原因。但反过来说，如果没有AT&T的反面教材，今天的GNU/Linux很有可能也不会出现。AT&T究竟是限制了Unix的发展，还是以反面示例促进了Unix社区，已不好评说。今天，软件是商业化好还是开源好的争论还在继续，纵观这几十年来Unix的历史，Linux的划时代地出现。相信你会得出自己的结论。不管怎么样，Unix的经历对计算机领域贡献的不单单是技术，他给我们提供了丰富而生动的教材。特别是Unix引发的哲学，让今天的我们依然受益不浅。

说到Unix为我们所带来的软件开发的哲学，我必需要说一说。Unix遵循的原则是KISS（Keep it simple， stupid）。在http：//en。wikipedia。org/wiki/Unix_philosophy 上有很多的基本上大同小异的Unix哲学，都是很经典的。

Doug McIlroy 是认为UNIX的哲学是这样的：三条哲学，简明扼要，就是这三条哲学贯穿着整个Unix世界。尤其是第一条“do one thing and do it well”真是相当精彩！

Write programs that do one thing and do it well。

Write programs to work together。

Write programs to handle text streams， because that is a universal interface。

只要是Unix的程序员，他们会比别的程序员在任何时候都会不停地强调着这三条哲学。

而《The Art of Unix Programming》总结了下面这些哲学，都是至理名言啊。

Rule of Modularity： Write simple parts connected by clean interfaces。

Rule of Clarity： Clarity is better than cleverness。

Rule of Composition： Design programs to be connected to other programs。

Rule of Separation： Separate policy from mechanism； separate interfaces from engines。

Rule of Simplicity： Design for simplicity； add complexity only where you must。

Rule of Parsimony： Write a big program only when it is clear by demonstration that nothing else will do。

Rule of Transparency： Design for visibility to make inspection and debugging easier。

Rule of Robustness： Robustness is the child of transparency and simplicity。

Rule of Representation： Fold knowledge into data so program logic can be stupid and robust。

Rule of Least Surprise： In interface design， always do the least surprising thing。

Rule of Silence： When a program has nothing surprising to say， it should say nothing。

Rule of Repair： When you must fail， fail noisily and as soon as possible。

Rule of Economy： Programmer time is expensive； conserve it in preference to machine time。

Rule of Generation： Avoid hand-hacking； write programs to write programs when you can。

Rule of Optimization： Prototype before polishing。 Get it working before you optimize it。

Rule of Diversity： Distrust all claims for “one true way”。

Rule of Extensibility： Design for the future， because it will be here sooner than you think。

X Windows 的设计者 Mike Gancarz 给出了下面九条哲学思想

Small is beautiful。

Make each program do one thing well。

Build a prototype as soon as possible。

Choose portability over efficiency。

Store data in flat text files。

Use software leverage to your advantage。

Use shell scripts to increase leverage and portability。

Avoid captive user interfaces。

Make every program a filter。

在今天，这种思想依然被传承着，在影响着世界上各个角落的每一个程序员。

Unix痛恨者手册

这里还需要值得一提的是一本叫《The Unix-Haters Handbook》，中文译做《Unix痛恨者手册》。可以在这里下载：http：//research。microsoft。com/~daniel/uhh-download。html。其中以调侃的语气声讨了Unix的种种不是。虽然这是十年前的一本书了，但还是值得一读。这本书指出了许多Unix的设计错误，指出了种种看起来很合理的设计走向了荒谬，还这样调侃了C语言——“如果说C语言给足了让你上吊的绳子，那么，C++在给了你足够的绳子把你的邻居全部捆起来之后，还给了你足够的绳子让你为一艘小帆船装上帆，最后你还有足够的绳子把自己吊死在帆船的桅杆上”。呵呵，相当的尖酸刻薄吧。里面有一句对操作系统的评价是这样的：“The fundamental difference between Unix and the Macintosh operating system is that Unix was designed to please programmers， whereas the Mac was designed

to please users。（Windows， on the other hand， was designed to please accountants。”（Windows设计给会计人员？！连计算机用户都不是了，呵呵）

不过，我可以感觉得到这本书的作者在书中对Unix的感情是比较复杂的，爱恨交加，在书的最后有这样一句话“would anyone have spent this much time and effort writing about how much they hated Unix if they didn’t secretly love it？ I’ll leave that to the readers to judge， but in the end， it really doesn’t matter： If this book doesn’t kill Unix， nothing will”。是的，如果Unix能够存活这么长的时间，那么，不会有什么东西可以把他消灭了。

从《Unix痛恨者手册》这本书，再加上Unix的历史，我们可以感到Unix的经历的风风雨雨，在Unix上面出现有种种教训，近40年的历程，Unix历经磨难，几近夭折，一路走来的确很不容易，让人由衷感叹。今天的Unix，今天的软件工业和以前相比已是不可同日而语。很大程度上，这些都要归功于这个充满苍桑的Unix。

后记

在中国我们开始学习计算机的时候，我们被Microsoft所创造的文化所笼罩里。就在Unix出现革命性的转变，在Unix影响计算机世界文化的那几年里，科班出生专业开发人员学习的是MS-DOS和微软的文化，我们犹如一个井底之蛙一样，对外面的翻天覆地的变化无动于衷。微软创造的文化在我们这里尤其地根深蒂固，我们几乎忘记了另外一边的Unix（参看《Unix 40年：Unix年鉴》、《Unix 40年：昨天，今天和明天》）。

在那充满激情的Unix的岁月里，大伙为了科研目的或个人兴趣在Unix上进行各种开发，并且不计较金钱利益，将这些源码公开，互相共享。在那里，开发和自由成为主题，正因为如此，当今的世界才如此丰富多采。在40年Unix文化和技术积淀的里面，蕴涵着比较纯正的计算机文化和思想。

纵观整个Unix的历史过程中，许许多多的程序员、工程师前辈们在Unix中所摸爬滚打，他们的辛勤地、他们呕心沥血地跟随Unix，努力建立一个繁荣的计算机世界的文明。Unix不是一个简简单单的操作系统。有人说，Unix是程序员设计给程序员的，一点没错。Unix的近40年历史造就了它的博大精深，它给程序员们带来的绝不仅仅只是技术上的知识。它的失误，它的无奈，它的精神，它的荣耀，它从技术和思想上都启迪着我们。对于程序员来说，学习Unix就等同于向前辈程序学习。无论你是什么样的程序员，你都应该了解Unix，这是开发人员的根，前面的开发者造就了它，而它又在引领后面的开发人员，它是前辈程序员们交给我们的一份礼物，一个接力棒，它是开发人员赖以生存的土壤，是上一辈程序员留给我们这一代程序员开启未来的钥匙。Unix就像一个程序员教父一样，理当受到我们的尊敬和崇拜。

（很长，对吧？没事，just enjoy it,再附一篇，祭奠驾鹤西去的Dennis）

丹尼斯•里奇对人类的贡献，被大大低估。此时此刻，全世界几十亿人正在使用的几百亿件电子产品，里面的软件最终都可以追溯到他的智力贡献。

==========

保持简单（Keep it simple）----纪念丹尼斯•里奇（Dennis Ritchie）

作者：阮一峰

1954年，电气工程师阿利斯泰尔•里奇（Alistair E. Ritchie），决定举家从纽约州的布朗克斯维尔（Bronxville），搬到几十公里以外的新泽西。这样可以离他的工作单位"贝尔实验室"更近一些。

13岁的丹尼斯•里奇（Dennis Ritchie），就这样随着父亲一起来到新泽西。那时，谁也没有想到，这个文静的少年将在这里待上一辈子，并且创造出改变世界的发明。

中学毕业后，丹尼斯•里奇听从父亲的建议，进入哈佛大学学习应用数学，直到拿到博士学位。他的第一份工作，是为军方研制核武器，这并不符合他的志趣。1967年，通过父亲介绍，26岁的他进入贝尔实验室，从事刚刚兴起的计算机研究。从此，他在新泽西老家一住就是44年，直到去世。

他在贝尔实验室的第一个任务，是参与大项目Multics，即开发一个前所未有的、可以多人使用的、同时运行多个程序的操作系统。该项目由贝尔实验室、麻省理工学院和通用电气公司三方联合研制，但是由于设计过于复杂，迟迟拿不出成果，1969年贝尔实验室宣布退出。

第一个任务这样无果而终，丹尼斯•里奇很不甘，但也无能为力。谁知过完了夏天，比他小两岁的同事肯•汤普森（Ken Thompson）找上门，说借鉴Multics的设计思路，做了一个个人项目Unix，问他有没有兴趣一起参与。丹尼斯•里奇立刻表示同意，于是两人一起投入Unix的开发。肯•汤普森的专业是电子工程，丹尼斯•里奇专业是应用数学，正好互补。经过日以继夜的工作，1969年圣诞节前，Unix已经可以初步运行了。

吸取了Multics设计复杂而导致失败的教训，丹尼斯•里奇将Unix的设计原则定为"保持简单和直接"（Keep it simple stupid），也就是后来著名的KISS原则。为了做到这一点，Unix由许多小程序组成，每个小程序只能完成一个功能，任何复杂的操作都必须分解成一些基本步骤，由这些小程序逐一完成，再组合起来得到最终结果。

表面上看，这样的设计很低效：为了取得结果，你必须运行一连串小程序，那么为什么不用一个大程序一次运行就得到结果呢？但是事实证明，由于小程序之间可以像积木一样自由组合，所以非常灵活，能够轻易完成大量意想不到的任务。而且，计算机硬件的升级速度非常快，所以性能也不是一个问题。另

一方面，开发单一目的的小程序，要比开发大型程序容易得多，所以Unix才有可能在短短几个月内问世。

Unix迅速在程序员中流传，到了80年代，已经成为主流操作系统，演变成整个软件工业的基础，当代最主要的操作系统----Windows、MacOS和Linux----都与Unix有关。由此可见，丹尼斯•里奇的"保持简单"原则，对计算机时代的影响有多大。

丹尼斯•里奇对世界的贡献还不止于此。Unix最早是用不通用的机器语言编写的，如果换一个型号的计算机，就必须重新编写一遍。为了提高通用性和开发效率，丹尼斯•里奇决定发明一种新的计算机语言----C语言。

C语言也贯彻了"保持简单"的原则，语法非常简洁，对使用者的限制很少。丹尼斯•里奇编写的教材《C编程语言》总共只有100多页，薄得难以置信。很多人都被它的简洁性吸引，学习并使用C语言。直到今天，C语言依然是世界上最重要的编程语言之一，"保持简单"原则显示了强大的生命力。

发明Unix和C语言，给丹尼斯•里奇带来巨大的荣誉，他得到了1983年的图灵奖、1990年的汉明奖、1999年的美国国家技术奖章。尽管功成名就，但是就像他的工程设计思想，丹尼斯•里奇在个人生活上也尽量"保持简单"。他依然住在新泽西，低调地生活，不太在媒体上曝光，终身没有结婚。

他也始终没有跳槽，一直在贝尔实验室工作，尽管后者多次分拆，到了最后已经名存实亡。2001年，接手贝尔实验室的朗讯公司，决定关闭大多数实验室，许多研究人员纷纷离开，包括Unix发明者之一的肯•汤普森都去了Google，但是丹尼斯•里奇哪里也没去，还是留了下来。2006年12月1日，贝尔实验室被整体卖给了法国阿尔卡特公司，第二年他就选择退休了。

退休以后，他过上了隐居生活，外界几乎忘了他的存在。2011年10月12日，共事20年的同事Rob Pike从加州到新泽西去拜访他，才发现他已经去世了。由于是独居，无法知道准确的死亡时间。据他的兄弟透露，这几年丹尼斯•里奇的健康状况一直不好，他患有前列腺癌和心脏病。

Rob Pike在Google Plus发了一条简短的消息，"据我所知，Unix和C语言发明人丹尼斯•里奇已经去世"，外界才知道这件事，引发了纪念浪潮。虽然这个过程有点令人伤感，但是必须说，这很符合他的"保持简单"的原则。

分类: ThinkCareer

Zephyroal
粉丝 - 108 关注 - 60

2 0

我在关注他 取消关注

您已推荐过，取消

« 上一篇： 宫崎骏用动漫教给我们的人生哲理，每一句都能说到心里[转]
» 下一篇： Android NDK中的C++调试踩坑标记

posted @ 2014-01-03 19:58 Zephyroal 阅读(1439) 评论(0) 编辑 收藏 举报

【推荐】行行AI人才直播第14期：《土豆利用GPT成功融资两次的提示词和故事》
【推荐】阿里云-云服务器省钱攻略： 五种权益，限时发放，不容错过

编辑推荐：
· 写给软件编程新手的建议
· golang技术降本增效的手段
· 你不知道的 HTTP Referer
· 记录一次线上服务 CPU 飙高问题
· CPU摸鱼被抓，上了一个新技术！

阅读排行：
· 一位大咖写给软件编程新手的建议 - 经验谈
· 简单了解一下国产操作系统
· 跟进 .NET 8 Blazor 之 ReuseTabs 支持 Query 属性绑定
· NativeBuferring，一种零分配的数据类型[上篇]
· 园子的商业化努力-阿里云开发者社区合作：RDS MySQL Serverless 免费试用活动